



中华人民共和国国家标准

GB/T XXXXX—XXXX/ISO/PAS 22720-2005

试验测试开放数据服务

Test open data services(TODS)

(ISO/PAS 22720-2005, MOD)

(征求意见稿)

2019-4-6

XXXX—XX—XX 发布

XXXX—XX—XX 实施

中华人民共和国国家质量监督检验检疫总局
中国国家标准化管理委员会 发布

目 次

前言.....	XVII
引言.....	XIX
1 范围.....	1
2 规范性引用文件.....	1
3 术语和定义.....	1
4 概述.....	11
4.1 目的	11
4.1.1 定位	11
4.1.2 现有系统的缺点	12
4.1.3 益处	13
4.2 技术方法	14
4.2.1 数据模型	14
4.2.2 接口	16
4.2.3 物理存储	17
4.2.4 传输格式 (ATF)	18
4.2.5 应用模型	18
4.3 对产品的影响	18
4.4 实施的技术水平	19
4.5 ODS 数据模型.....	19
4.5.1 基础模型	20
4.5.2 应用模型	22
4.5.3 实例	24
4.6 应用程序编程接口 (API)	25
4.6.1 面向对象 API 接口 (OO-API)	27
4.6.1.1 入口点	28
4.6.1.2 存取基础模型	28
4.6.1.3 存取应用模型	29
4.6.1.4 存取一般实例	29
4.6.1.5 存取结果数据	30
4.6.1.6 查询	30
4.6.2 程序接口 (RPC-API)	31
4.6.3 TODS API 接口 (TODS-API)	31
4.7 物理数据存储	31
4.8 传输格式	34
4.8.1 ATF/CLA	35
4.8.2 ATF/XML	36

4.9 应用模型.....	36
5 体系结构	37
5.1 简介.....	37
5.2 基础模型的元素.....	38
5.2.1 概述.....	38
5.2.2 环境.....	38
5.2.3 维度和单位.....	39
5.2.4 数据管理.....	39
5.2.5 测量.....	39
5.2.6 说明性数据.....	39
5.2.7 安全.....	39
5.2.8 其他数据.....	40
5.3 应用模型规则.....	40
5.3.1 应用元素命名规则.....	40
5.3.2 构建应用模型.....	40
5.3.3 映射基础模型和应用模型.....	42
5.3.4 存储测量数据.....	43
5.3.5 应用元素之间的关系.....	48
5.4 模型中的属性和关系.....	49
5.4.1 属性规则.....	49
5.4.2 所有基础元素的共同属性.....	49
5.4.3 应用属性.....	49
5.4.4 实例属性.....	50
5.4.5 关系规则.....	51
5.4.6 应用关系.....	51
5.5 数据类型的用法.....	52
5.5.1 ASAM ODS 中的数据类型的一般说明.....	52
5.5.2 基础模型和 APIs 的数据类型.....	54
5.5.3 标准化 ASAM 数据类型.....	59
5.6 应用接口.....	59
5.6.1 基本信息.....	59
5.6.2 APIs 工作.....	63
5.7 继承的概念.....	67
5.7.1 目的.....	67
5.7.2 具有继承机制的数据建模.....	67
5.7.3 继承和派生.....	69
5.8 ATF 在 ODS 体系架构中的应用.....	69
5.9 ASAM ODS 的安全概念.....	69
5.9.1 存取 ASAM ODS 信息的全球规则.....	70
5.9.1.1 用户端登录和服务器的身份验证.....	70
5.9.1.2 超级用户.....	70
5.9.1.3 一般存取限制.....	70

5.9.1.4	自动程序	71
5.9.2	基于 ACL 的安全概念概述	71
5.9.3	基于 ACL 的安全等级	72
5.9.3.1	元素安全：保护一个应用元素的所有实例	72
5.9.3.2	实例安全：保护单个实例	73
5.9.4	基本存取权限	76
5.9.5	客户存取现有数据对象	78
5.9.5.1	一般自由存取	78
5.9.5.2	存取控制模式	81
5.9.6	模板	82
5.9.7	确定用于新数据对象的 ACL 条目	83
5.9.8	继承概念安全	86
5.9.9	基于 ACL 的安全概念备注	87
5.10	其它因素	88
5.10.1	绝对和相对日期与时间	88
5.10.2	参数和参数集概念	89
5.10.2.1	AoParameter 的说明	90
5.10.2.2	AoParameterSet 的说明	90
5.10.2.3	基础模型中的关系	90
5.10.2.4	Parameters 在应用模型中的用法	90
5.10.3	量和单位的用法	92
5.10.3.1	量和它们的名称的概述	93
5.10.3.2	ASAM ODS 内的量的层次结构	93
5.10.3.3	ASAM ODS 内的量的用法	94
5.10.3.4	单位转换	94
5.10.4	别名用法	96
5.10.5	ASAM ODS 中的枚举	99
5.10.6	在一台 ASAM ODS 服务器存取文件	100
5.11	与其他 ASAM 标准的兼容性	112
5.11.1	经协调一致的 ASAM 数据类型	112
5.11.2	VSIM 车辆安全测试标准	112
5.11.3	MDF 存储 ECU 标定系统的测量数据	112
6	物理存储	115
6.1	概述	115
6.1.1	简介	115
6.1.2	常用的数据类型	115
6.2	SVC 表的描述	116
6.2.1	SVCENT	116
6.2.2	SVCATTR	118
6.2.2.1	应用属性的标记	120
6.2.2.2	ADTYPE 列的数据类型编码	120
6.2.3	SVCREF	122

6.2.3.1	实例之间保存 N: M 关系的表	123
6.2.4	SVCVAL	123
6.2.4.1	显式和隐式局部列及生成算法	126
6.2.5	SVCINST	128
6.2.6	SVCENUM	129
6.2.6.1	ASAMODS 枚举定义	129
6.2.6.2	早期版本的物理存储问题	130
6.2.7	SVCACLI	130
6.2.8	SVCACLA	131
6.2.9	SVCTPLI	132
6.2.10	SVCVAL_SPS	133
6.2.10.1	显式局部列	135
6.2.10.2	隐式局部列及其生成算法	140
6.3	属性值的存储	145
6.3.1	单值的存储	145
6.3.2	字节流值的存储	145
6.3.3	BLOB 值的存储	145
6.3.4	日期和时间的存储	146
6.3.5	属性阵列值和对象值的存储	146
6.3.5.1	通过 RPC-API 访问	147
6.3.6	枚举类型属性的存储	148
6.3.7	继承概念和物理存储	148
6.3.7.1	应用属性和关系的存储	148
6.3.7.2	实例属性的存储	151
6.4	混合模式的存储	151
6.4.1	BLOB 类型值的存储	153
6.4.2	字符串类型值的存储	154
6.4.3	字节流类型值的存储	155
6.4.4	数值型值的存储	157
6.4.5	标记的存储	161
6.4.6	示例	162
6.4.7	与 ATF 组件的联系	165
6.4.7.1	AoExternalComponent 属性到 ATF/CLA 元素的映射	165
6.4.8	混合模式服务器	166
6.4.8.1	混合模式服务器操作的用例	167
6.4.8.2	混合模式服务器实施注意事项	168
6.4.8.3	定义每个片段文件大小的参数	168
6.4.8.4	混合模式服务器的模式转换	168
6.4.8.5	对 ODS 安全性的影响	168
7	基础模型	168
7.1	基础模型介绍	168
7.1.1	概况	168

7.1.2	派生原则	169
7.1.3	元素属性关系	170
7.1.3.1	基础元素、属性和关系的名称	170
7.1.3.2	应用原素的名称, 属性及关系	170
7.1.3.3	可选属性和必选属性	170
7.1.3.4	SET 或 LIST 型的属性和关系	171
7.1.4	ASAM ODS 基础模型和基于 ACL 的安全概念	171
7.2	环境基础元素的描述	173
7.2.1	AoEnvironment (BID=1)	173
7.2.2	AoNameMap (BID=46)	175
7.2.3	AoAttributeMap (BID=47)	176
7.3	基础元素尺寸和单位的描述	178
7.3.1	AoQuantity (BID=11)	178
7.3.2	AoUnit (BID=13)	180
7.3.3	AoPhysicalDimension (BID=15)	182
7.3.4	AoQuantityGROUP (BID=12)	184
7.3.5	AoUnitGROUP (BID=14)	185
7.4	测量基础元素的描述	186
7.4.1	AoMeasurement (BID=3)	186
7.4.2	AoMeasurementQuantity (BID=4)	188
7.4.3	AoSubMatric (BID=38)	190
7.4.4	AoLocalColumn (BID=39)	191
7.4.5	AoExternalComponent (BID=40)	197
7.5	执行基础元素描述	200
7.5.1	AoTest (BID=36)	200
7.5.2	AoSubTest (BID=2)	201
7.6	描述性数据基础元素的描述	202
7.6.1	AoUnitUnderTest (BID=21)	203
7.6.2	AoUnitUnderTest (BID=22)	203
7.6.3	AoTestSequence (BID=25)	205
7.6.4	AoTestSequencePart (BID=26)	206
7.6.5	AoTestEquipment (BID=23)	207
7.6.6	AoTestEquipmentPart (BID=24)	208
7.6.7	AoTestDevice (BID=37)	209
7.7	安全性的基础元素描述	210
7.7.1	AoUser (BID=34)	210
7.7.2	AoUserGroup (BID=35)	211
7.8	其它数据基础元素的描述	212
7.8.1	AoAny	212
7.8.2	AoLog	213
7.8.3	AoParameter (BID=44)	214
7.8.4	AoParameterSet (BID=45)	215
7.9	ASAM ODS 基础模型形式描述	216

7.9.1	为什么采用标准化的表述	216
7.9.2	用 ISO 10303 描述元模型	216
7.9.3	用 EXPRESS 表达的基础模型	218
7.9.4	用 EXPRESS-G 图表达的基础模型	253
8	TODS 标准模型	265
8.1	TODS 标准模型介绍	266
8.1.1	扩展规则	266
8.2	环境元素的描述	266
8.2.1	TdmEnvironment	267
8.2.2	TdmNameMap	267
8.2.3	TdmAttributeMap	268
8.2.4	TdmFile	268
8.3	管理元素的描述	270
8.3.1	TdmTest	270
8.3.2	TdmSubTest	271
8.3.3	TdmTestTask	272
8.4	测量及测量数据元素的描述	275
8.4.1	TdmMeasurementQuantity	275
8.4.2	TdmMeasurement	276
8.4.3	TdmVedioFile	278
8.4.4	TdmPictureFile	279
8.4.5	TdmAffixFile	279
8.4.6	TdmSubmatrix	280
8.4.7	TdmLocalColumn	282
8.5	描述数据的描述	283
8.5.1	TdmTestUserGroup	283
8.5.2	TdmTestUser	284
8.5.3	TdmUnitUnderTest	284
8.5.4	TdmUUTrouble	285
8.5.5	TdmUUTRecord	285
8.5.6	TdmTestEquipment	286
8.5.7	TdmEquipmentTrouble	287
8.5.8	TdmEquipmentRecord	288
8.6	量与单位的元素描述	289
8.6.1	TUnitGroup	290
8.6.2	TUnit	290
8.6.3	TPhysicalDemension	291
8.6.4	TQuantityGroup	293
8.6.5	TQuantity	294
8.7	试验件管理的描述	295
8.7.1	TdmProductModel	295
8.7.2	TdmProductPhase	296

8.7.3	TdmProductNode	298
8.7.4	TdmUUTObjGroup	299
8.7.5	TdmUUTObj	299
8.7.6	TdmUUTObjComponent	301
8.7.7	TdmUUTObjParameter	301
8.8	设备管理的描述	302
8.8.1	TdmEquipmentType	302
8.8.2	TdmEquipment	303
8.8.3	TdmEquipmentComponent	305
8.9	知识元素的描述	306
8.9.1	TdmWsParamLibrary	306
8.9.2	TdmTestParamLibrary	307
8.9.3	TdmUUTParamLibrary	308
8.9.4	TdmStateTpye	308
8.9.5	算法库	308
8.10	质量体系元素的描述	309
8.10.1	质量体系文件	309
8.10.2	质量标准表单	310
8.11	安全元素的描述	310
8.11.1	TdmUserGroup	310
8.11.2	TdmUser	311
8.11.3	TdmSecurityGrade	312
8.12	其它的描述	313
8.12.1	TdmDataRelation	313
8.13	TODS OO-API (面向对象应用编程接口)	314
8.13.1	IMapleTRFactory (工厂)	314
8.13.1.1	获取当前系统版本信息	314
8.13.1.2	获取基础模型版本信息	314
8.13.1.3	获取应用模型版本信息	315
8.13.1.4	创建会话	315
8.13.2	IMapleTRSession (会话)	315
8.13.2.1	关闭会话	315
8.13.2.2	获取 IMapleTRModelAccess	315
8.13.2.3	获取 IMapleTRDBAccess	316
8.13.2.4	获取 IMapleTRFileAccess	316
8.13.3	IMapleTRModelAccess (数据模型)	316
8.13.3.1	获取所有基础模型清单	316
8.13.3.2	获取所有基础模型信息	316
8.13.3.3	获取制定模型信息	317
8.13.3.4	获取关联模型信息	317
8.13.3.5	获取指定关联模型信息	317
8.13.3.6	获取所有应用模型信息	318
8.13.3.7	新建应用模型	318

8.13.3.8 编辑应用模型信息	318
8.13.3.9 删除应用模型信息	318
8.13.4 IMapleTRDBAccess (DB 数据访问)	319
8.13.4.1 获取模型实例数据	319
8.13.4.2 获取模型实例数据集合	319
8.13.4.3 获取关联模型的实例数据	319
8.13.4.4 新建模型实例数据	319
8.13.4.5 编辑模型数据	320
8.13.4.6 删除模型实例数据	320
9 标准的传输格式 (ATF/CLA)	321
9.1 引言	321
9.2 ATF/CLA 文件	321
9.2.1 ATF/CLA 文件的作用	321
9.2.2 ATF/CLA 文件的结构	322
9.3 ATF/CLA 的语法描述	322
9.3.1 概述	322
9.3.2 本章所用字符集的说明	323
9.4 本章使用的符号的说明	323
9.5 ATF/CLA 保留的关键词	324
9.6 ATF/CLA 保留关键词的简要描述	325
9.6.1 ATF/CLA 保留的关键词的结构	326
9.7 ATF/CLA 语法图	326
9.7.1 空格的结构 (空格 WHITESPACE)	326
9.7.2 分隔符的结构 (分隔符 SEPARATOR)	327
9.7.3 注释的结构 (注释 COMMENT)	327
9.7.4 转义序列的结构 (转义 ESCAPE)	328
9.7.5 整数的结构 (整数 INTEGER)	330
9.7.6 浮点数结构 (REAL)	331
9.7.7 复数的结构 (COMPLEX)	335
9.7.8 字符常量的结构 (CHARACTER)	335
9.7.9 字符串的结构 (STRING)	335
9.7.10 字节流的结构 (BYTESTREAM)	336
9.7.11 标识符的结构 (IDENTIFIER)	336
9.7.12 文件名的结构 (FILENAME)	337
9.7.13 预定义的数据类型 (DATATYPE)	338
9.7.14 枚举的结构	339
9.7.15 ATF/CLA 版本标识符的结构 (VERSION)	340
9.7.16 文件块的结构 (FILES)	341
9.7.17 指令 INCLUDE 的结构 (INCLUDE)	342
9.7.18 参考的单位 (PHYS_UNIT)	342
9.7.19 数据属性值 (DATAVALUE)	343
9.7.20 应用元素的结构 (APPLELEM)	343

9.7.21	实例元素的结构 (INSTELEM)	345
9.7.22	组件的结构 (COMPONENT)	349
9.7.23	结束文件的结构 (ENDFILE)	356
9.8	ATF/CLA 文件的结构概述 (ATF_OVERVIEW)	356
9.9	ATF/CLA 文件的安全信息	356
9.9.1	ATF/CLA 安全信息的使用	356
9.9.2	ATF/CLA 安全信息的表达	357
9.10	继承的概念和 ATF/CLA	358
9.11	应用示例	360
10	传输格式 (ATF/XML)	386
10.1	ATF/XML 的概述	386
10.1.1	介绍	386
10.1.2	ATF/XML 和 ATF/CLA 的比较	386
10.1.3	ATF/XML 中数据类型的使用	389
10.1.4	基础元素, 属性, 关系和枚举的名称	389
10.1.5	命名 ATF/XML 文件	390
10.2	ATF/XML 文件	390
10.2.1	ATF/XML 文件的整体结构	390
10.2.2	逻辑和物理 ATF/XML 文件	391
10.2.3	ATF/XML 文件的 XML 声明	394
10.2.4	根元素<ATFX_FILE>	394
10.2.5	元素<DOCUMENTATION>	395
10.2.6	元素<EXPORTED_BY>	395
10.2.7	元素<EXPORTER>	396
10.2.8	元素<EXPORT_DATE_TIME>	396
10.2.9	元素<EXPORTER_VERSION>	396
10.2.10	元素<SHORT_DESCRIPTION>	396
10.2.11	元素<LONG_DESCRIPTION>	396
10.2.12	元素<ESCAPE_SPECIAL_CHARS>	396
10.2.13	元素<BASE_MODEL_VERSION>	397
10.2.14	元素<FILES>和<COMPONENT>	397
10.2.15	元素<IDENTIFIER>	397
10.2.16	元素<FILENAME>	398
10.2.17	元素<APPLICATION_MODEL>	399
10.2.18	元素<APPLICATION_ENUMERATION>	399
10.2.19	元素<ITEM>	400
10.2.20	元素<VALUE>	400
10.2.21	元素<APPLICATION_ELEMENT>	401
10.2.22	元素<NAME>	401
10.2.23	元素<BASETYPE>	401
10.2.24	元素<INHERITS_FROM>	401
10.2.25	元素<APPLICATION_ATTRIBUTE>	401

10.2.26	元素<BASE_ATTRIBUTE>	403
10.2.27	元素<DATATYPE>	403
10.2.28	元素<ENUMERATION_TYPE>	403
10.2.29	元素<AUTOGENERATE>	404
10.2.30	元素<OBLIGATORY>	404
10.2.31	元素<UNIQUE>	404
10.2.32	元素<LENGTH>	404
10.2.33	元素<RELATION_ATTRIBUTE>	404
10.2.34	元素<REF_TO>	408
10.2.35	元素<BASE_RELATION>	408
10.2.36	元素<MIN_OCCURS>	408
10.2.37	元素<MAX_OCCURS>	408
10.2.38	元素<INVERSE_NAME>	409
10.2.39	元素<INSTANCE_DATA>	409
10.2.40	名称中特殊字符的使用	415
10.2.41	实例属性说明	416
10.2.42	使用外部文件组件	417
10.2.43	存储安全性信息	431
10.2.44	继承概念和 ATF/XML	435
10.3	ODS 基本的 XML 模式	439
10.3.1	验证 ATF/XML 文件	439
10.3.2	ODS 基本 XML 模式的主要特征	439
10.4	特定应用模式注意事项	441
10.4.1	起因	441
10.4.2	应用元素的模式定义	443
10.4.3	实例属性的模式定义	444
10.4.4	特定应用模式的进一步支持	446
10.5	相关的模式文件	446
10.5.1	ASAM 一致的数据类型模式文件	446
10.5.2	ODS 基本的 XML 模式文件	453
10.5.3	ASAM ODS 辅助模式文件	470
10.6	ATF/XML 文件例子	488
10.7	已知问题	507
10.8	附加资源	508
11	MIME 类型与外部引用	508
11.1	简介	508
11.2	ASAM ODS MIME 类型的定义	509
11.2.1	作为可选基本属性的 MIME_TYPE	509
11.2.2	MIME_TYPE 字符串的内容	509
11.2.3	通过 ODS API 读取 MIME 类型数据	510
11.2.4	通过 ODS API 写出 MIME 类型数据	510
11.3	ASAM ODS 基础 MIME 类型	510

11.4 外部引用的处理	511
12 ASAM ODS RPC-API	512
12.1 程序声明	512
12.2 使用的数据类型	512
12.3 会话服务（开放式和封闭式）	513
12.3.1 AOP_OPENENV	513
12.3.2 AOP_CLOSEENV	515
12.4 元信息服务	515
12.4.1 AOP_GETAPPLINF	515
12.4.2 AOP_GETATTR	516
12.5 应用元素值服务	517
12.5.1 AOP_GETINSTREF	517
12.5.2 AOP_SETINSTREF	518
12.5.3 AOP_GETVAL	519
12.5.4 AOP_GETVALE	521
12.5.5 AOP_GETINSTATTR	526
12.5.6 AOP_PUTVAL	527
12.6 测量和子阵服务	528
12.6.1 AOP_GETVALATTR	528
12.6.2 AOP_GETVALINF	529
12.6.3 AOP_GETVALVAL	530
12.6.4 AOP_PUTVALVAL	531
12.7 属性处理	533
12.7.1 AOP_SETPAR	533
12.7.2 AOP_GETPAR	534
12.8 安全服务	535
12.8.1 AOP_SETSECURITYLEVEL	536
12.8.2 AOP_GETSECURITYLEVEL	537
12.8.3 AOP_SETRIGHTS	537
12.8.4 AOP_GETRIGHTS	538
12.8.5 AOP_SETINIRIGHTS	539
12.8.6 AOP_GETINIRIGHTS	539
12.8.7 AOP_SETINIRIGHTSREF	540
12.8.8 AOP_GETINIRIGHTSREF	540
12.8.9 AOP_SETCURRENTINIRIGHTS	541
12.8.10 AOP_SETPASSWORD	541
12.8.11 用户验证	542
12.8.12 ODS 中的密码加密（RPC）	542
12.9 错误处理	542
12.10 RPC-API 特定的方面	543
12.10.1 序列表达的限制	543
12.10.2 继承概念与 RPC-API	543

12.11	调用序列示例	544
12.12	ASAM ODS RPC-API IDL	545
13	ASAM ODS OO-API	572
13.1	概述	572
13.1.1	一般信息	572
13.2	OO-API 的接口	574
13.2.1	AOFACTORY	574
13.2.2	AOSESSION	577
13.2.3	APPLELEMACCESS	595
13.2.4	APPLICATIONATTRIBUTE	611
13.2.5	APPLICATIONELEMENT	623
13.2.6	APPLICATIONRELATION	638
13.2.7	APPLICATIONSTRUCTURE	645
13.2.8	BASEATTRIBUTE	662
13.2.9	BASEELEMENT	664
13.2.10	BASERELATION	668
13.2.11	BASESTRUCTURE	671
13.2.12	BLOB	677
13.2.13	COLUMN	680
13.2.14	ELEMRESULTSETTEXTSEQITERATOR	687
13.2.15	ENUMERATIONDEFINITION	689
13.2.16	INSTANCEELEMENT	692
13.2.17	INSTANCEELEMENTITERATOR	707
13.2.18	MEASUREMENT	708
13.2.19	NAMEITERATOR	711
13.2.20	NAMEVALUEITERATOR	712
13.2.21	NAMEVALUEUNITIDITERATOR	714
13.2.22	NAMEVALUEUNITITERATOR	716
13.2.23	NAMEVALUEUNITSEQUENCEITERATOR	718
13.2.24	QUERY	720
13.2.25	QUERYEVALUATOR	723
13.2.26	SMATLINK	726
13.2.27	SUBMATRIX	730
13.2.28	VALUEMATRIX	732
13.3	ASAM ODS 类型定义	744
13.3.1	ASAM ODS 数据类型	744
13.3.2	数据类型转换	745
13.3.3	ASAM ODS 常数	746
13.3.4	ASAM ODS 属性类型	747
13.3.5	ASAM ODS 构造函数	747
13.3.6	ASAM ODS 查询状态	747
13.3.7	ASAM ODS 连接类型	747

13.3.8	ASAM ODS 权限设置.....	747
13.3.9	ASAM ODS 聚合函数.....	748
13.3.10	ASAM ODS 选择操作代码.....	748
13.3.11	ASAM ODS 选择操作符.....	749
13.3.12	ASAM ODS 选择类型.....	749
13.3.13	ASAM ODS 设置类型.....	750
13.3.14	ASAM ODS 严重性标志.....	750
13.3.15	ASAM ODS 值矩阵模式.....	750
13.3.16	ASAM ODS 关系.....	751
13.3.17	ASAM ODS 模式.....	753
13.3.18	ASAM ODS 异常.....	753
13.3.19	ASAM ODS 特定类型.....	757
13.3.20	ASAM ODS 联合.....	759
13.3.21	ASAM ODS 结构.....	761
13.3.22	ASAM ODS OO-API 相关语境变量.....	770
13.4	编程示例.....	771
13.4.1	通过 CORBA 访问 ASAM ODS 工厂对象.....	771
13.5	OO-API 定义文件.....	772
13.5.1	ODS.IDL.....	772
13.6	使用 ASAM 协调一致的数据类型.....	1032
13.6.1	映射文件 A_TYPES.IDL.....	1033
13.7	快速参考.....	1036
13.7.1	接口 AOFACORY.....	1037
13.7.2	接口 AOSESSION.....	1037
13.7.3	接口 APPLELEMACCESS.....	1038
13.7.4	接口 APPLICATIONATTRIBUTE.....	1039
13.7.5	接口 APPLICATIONELEMENT.....	1039
13.7.6	接口 APPLICATIONRELATION.....	1040
13.7.7	接口 APPLICATIONSTRUCTURE.....	1041
13.7.8	接口 BASEATTRIBUTE.....	1042
13.7.9	接口 BASEELEMENT.....	1042
13.7.10	接口 BASERELATION.....	1042
13.7.11	接口 BASESTRUCTURE.....	1043
13.7.12	接口 BLOB.....	1043
13.7.13	接口 COLUMN.....	1043
13.7.14	接口 ELEMRESULTSETTEXTSEQITERATOR.....	1044
13.7.15	接口 ENUMERATIONDEFINITION.....	1044
13.7.16	接口 INSTANCEELEMENT.....	1044
13.7.17	接口 INSTANCEELEMENTITERATOR.....	1045
13.7.18	接口 MEASUREMENT.....	1046
13.7.19	接口 NAMEITERATOR.....	1046
13.7.20	接口 NAMEVALUEITERATOR.....	1046
13.7.21	接口 NAMEVALUEUNITIDITERATOR.....	1046

13.7.22	接口 NAMEVALUEUNITITERATOR.....	1047
13.7.23	接口 NAMEVALUEUNITSEQUENCEITERATOR.....	1047
13.7.24	接口 QUERY	1047
13.7.25	接口 QUERYEVALUATOR	1047
13.7.26	接口 SMATLINK	1047
13.7.27	接口 SUBMATRIX	1048
13.7.28	接口 VALUEMATRIX	1048
14	TODS API (应用编程接口)	1049
14.1	IMapleTRFactory (工厂)	1049
14.1.1	获取当前系统版本信息.....	1049
14.1.2	获取基础模型版本信息.....	1049
14.1.3	获取应用模型版本信息.....	1050
14.1.4	创建会话.....	1050
14.2	IMapleTRSession (会话)	1050
14.2.1	关闭会话.....	1050
14.2.2	获取 IMapleTRModelAccess	1050
14.2.3	获取 IMapleTRDBAccess	1051
14.2.4	获取 IMapleTRFileAccess	1051
14.3	IMapleTRModelAccess (数据模型)	1051
14.3.1	获取所有基础模型清单.....	1051
14.3.2	获取所有基础模型信息.....	1051
14.3.3	获取制定模型信息.....	1052
14.3.4	获取关联模型信息.....	1052
14.3.5	获取指定关联模型信息.....	1052
14.3.6	获取所有应用模型信息.....	1052
14.3.7	新建应用模型.....	1053
14.3.8	编辑应用模型信息.....	1053
14.3.9	删除应用模型信息.....	1053
14.4	IMapleTRDBAccess (DB 数据访问)	1053
14.4.1	获取模型实例数据.....	1054
14.4.2	获取模型实例数据集合.....	1054
14.4.3	获取关联模型的实例数据.....	1054
14.4.4	新建模型实例数据.....	1054
14.4.5	编辑模型数据.....	1055
14.4.6	删除模型实例数据.....	1055
附录 A	(规范性附录) 执行状态 (ErrorCode)	1055

前 言

本标准按照 GB/T 1.1-2009 给出的规则起草。

本标准采用重新起草法修改采用 ISO/PAS 22720: 2005《ASAM ODS 5.0》。本标准与 ISO/PAS 22720: 2005 存在技术性差异。本标准在 ISO/PAS 22720-2005 ASAM 开放数据服务 5.0 的基础上, 结合试验测试开放数据服务的新的标准化进展, 参考了 ASAM ODS 5.2 的内容进行编写。变动的内容如下:

——按照 GB/T 1.1-2009 的规范起草标准, 增加了引言、第 1 章范围、第 2 章规范性引用文件、第 3 章术语、其中第 2 章增加了标准中引用的文件; 第 3 章将标准中用到的术语进行了定义;

——ISO 标准第 1 章为本标准第 4 章, 其他章节号顺延;

——本标准第 4 章参考 ASAM ODS 5.2 的内容进行编写;

——本标准第 5.3.4 条、5.3.5 条、5.4.3 条、5.4.4 条、5.4.5 条、5.4.6 条、5.5.2 条(表后面对数据类型解释), 参考 ASAM ODS 5.2 的内容进行编写;

——本标准第 5.7 条标题修改为继承的概念, 内容参考 ASAM ODS 5.2 的内容进行编写;

——本标准第 5.9 条、6.2.1 条、6.2.2 条、6.2.2.1 条、6.2.2.3 条、6.2.3 条、6.2.4 条、6.2.5 条、6.2.6 条、6.2.7 条、6.2.8 条、6.2.9 条、6.2.10 条、6.3.4 条、6.3.6 条、6.3.7 条、6.2.9 条、6.4.1 条、6.4.2 条、6.4.3 条、6.4.4 条、6.4.5 条、6.4.6 条、6.4.7 条、7.1.3 条、7.1.4 条、7.2.1 条、7.3.3 条、7.3.4 条、7.4.2 条、7.4.4 条、7.4.5 条、7.8 条、7.9.3 条、9.7.20 条、9.7.20 条、9.7.21 条、9.7.21.1 条、9.7.22 条、9.9.1 条、9.9.2 条、10.2.1 条、10.2.4 条、10.2.5 条、10.2.6 条、10.2.7 条、10.2.8 条、10.2.9 条、10.2.10 条、10.2.11 条、10.2.12 条、10.2.13 条、10.2.14 条、10.2.15 条、10.2.16 条、10.2.17 条、10.2.18 条、10.2.19 条、10.2.20 条、10.2.21 条、10.2.22 条、10.2.23 条、10.2.24 条、10.2.25 条、10.2.26 条、10.2.27 条、10.2.28 条、10.2.29 条、10.2.30 条、10.2.31 条、10.2.32 条、10.2.33 条、10.2.34 条、10.2.35 条、10.2.36 条、10.2.37 条、10.2.38 条、10.2.39 条、10.2.40 条、10.2.41 条、10.2.42 条、10.2.43 条、12.3 条、12.8.7 条、12.10 条、13.2.4 条(19)、13.2.13 条(17)、13.2.16 条(1)(3)(24)(28)、13.2.27 条(20)、13.3.16 条、13.5 节参考 ASAM ODS 5.2 的内容进行编写;

——本标准删除了 ASAM ODS 5.0 中 USE CASES(ASAM ODS 5.0 中 3.4.2 条)、APPROACH FOR SPECIFYING MIXED-MODE-SERVER(ASAM ODS 5.0 中 3.4.3 条)和 SEGMENTATION WITHIN THE MIXED-MODE-SERVER(ASAM ODS 5.0 中 3.4.4 条)、USE-CASES(ASAM ODS 5.0 中 3.4.5 条)、USE-CASES(ASAM ODS 5.0 中 3.4.5 条)的内容;

——本标准第 6.2.10 条、9.10 条、10.1.3 条、10.1.4 条、10.1.5 条、10.2.3 条、11.4 条、12.3 条的介绍部分、12.8 条的介绍部分、12.8.4 条、13.2.1 条(2)、13.2.2 条介绍部分、13.2.2 条(5)(6)(15)(20)、13.2.3 条(8)(9)(11)(12)(21)、13.2.4 条(5)、13.2.5 条(19)、13.2.6 条(18)、13.2.13 条(2)(3)(4)(6)(7)(9)(13)(16)(29)、13.2.18 条(3)(4)、13.2.27(2)(3)(8)(11)(12)(13)、13.3.12、13.3.15 条、13.3.22 条是参考 ASAM ODS 5.2 增加的内容;

——本标准第 8 章和第 14 章是增加的内容;

——本标准 7.3 条删除了 overview(ASAM ODS 5.0 中 4.3.1 条)、information about quantities and units in asam odas(ASAM ODS 5.0 中 4.3.2 条)、conversion of units(ASAM ODS 5.0 中 4.3.8 条)的内容;

——本标准 7.8 条删除了 PARAMETERS AND PARAMETER SETS(ASAM ODS 5.0 中 4.8.3 条)的内容;

——本标准第 10 章参考 ASAM ODS 5.2 的内容进行编写;

- 删除了原 ISO 标准第 10 章 NVH 应用模型、第 11 章标定应用模型的内容；
- 为简化编辑工作，每章前面的结构简介和最后的修订历史等重复内容不再纳入。

本标准由北京瑞风协同科技股份有限公司提出。

本标准由全国自动化系统与集成标准化技术委员会工业数据分委会（SAC/TC 159 SC4）归口。

本标准主要起草单位：北京瑞风协同科技股份有限公司、中国标准化研究院。

本标准主要起草人： 。

引 言

汽车、航空、航天、建筑、电子等行业有各类测试、评价与仿真解决方案，它们都有各自独立的管理系统和数据存储的专用格式，信息维护的方法也各不相同。随着产品创新复杂程度的增加，扩展和修改这些解决方案并将它们与第三方系统实现集成日益成为具有挑战性的问题，同时由于信息描述各异且难以访问，导致无法实现基于统一知识库的数据挖掘和跨学科的协同创新，新技术和新方法难以应用于已有的系统中，因此迫切需要一种简便和标准化的信息存取方法。产品生命周期信息的存取也需要标准化的方法支撑，以应对软件工具和业务不连续带来的风险。

ISO/PAS 22720: 2005《ASAM ODS 5.0》是 ISO/TC184/SC4 工业自动化系统与集成工业数据标准化分技术委员会在 ASAM 协会标准的基础上制定的技术标准，在国际上得到了广泛的应用，通过规范数据模型、接口、数据交换语法和格式，以解决试验领域数据共享的问题。

近年来，ASAM 发布了新版 ODS 标准，国内对试验领域的数据交换与共享需求也非常迫切，为此需要结合新版 ODS 标准和国内的需求，对 ASAM ODS 5.0 进行修订，并参考 ASAM ODS 5.2，形成测试数据专用的、先进的试验测试开放数据服务（TODS）。

通过对 ASAM ODS 5.0 标准的扩展，TODS（试验测试开放数据服务）能满足试验测试信息保存与获取的可持续性需求，降低项目中的成本和风险，并为信息系统提供一种可靠的数据共享与交换的基础。通过利用标准化的接口和公共的数据结构可降低异构环境中系统集成的难度，显著提升信息交换的效益。

试验测试开放数据服务

1 范围

本标准继承 ASAM ODS 的标准，并重点定义或强调了如下技术内容：

—— 扩展《ISO/PAS 22720 ASAM Open Data Services 5.0》基础模型，定义了 TODS 公共的数据模型（TODS 标准模型），通过为数据增加语义提供了初步的信息分类，实现不同的系统能够采用相同的方式解释相似的数据。TODS 标准模型是各种派生的 TODS 应用模型的基础，这些应用模型覆盖了特定应用领域的需求；

—— 定义了 TODS 服务的标准化的数据存取接口(TODS API)，包括对实际 TODS 应用模型的形式化描述（元信息）的维护接口，可使系统用统一的方式处理不同的 TODS 服务的数据交换及存取；

—— 继承了基于 TODS 的不同系统和平台间交换数据和元信息的标准化格式（ATF/XML）。

本标准适用于试验自动化和试验台系统，但不仅限于这些领域，只要需要采用一致的方式保存试验信息，都可以使用本规范，例如试验数据管理、测量与标定、自动化与测量系统集成、仿真、数据后处理、试验报告编制等。

2 规范性引用文件

下列文件对于本文件的应用是必不可少的。凡是注日期的引用文件，仅所注日期的版本适用于本文件。凡是不注日期的引用文件，其最新版本（包括所有的修改单）适用于本文件。

GB/T 16656.11-2010 工业自动化系统与集成 产品数据表达与交换 第11部分：描述方法：EXPRESS 语言参考手册（ISO 10303.11-2004, IDT）

ISO 639-1 语种名称代码 第1部分： α -2代码

ISO 8601-2004 数据元和交换格式 信息交换 日期和时间的表示

ISO/IEC 8859-1 信息技术 8位单字节编码图形字符集 第1部分：1号拉丁字母

ISO/TS 22240道路车辆. 车辆安全信息模型(VSIM)对象管理组织（OMG）：www.omg.org

ANSI/IEEE 754 浮点运算标准

可扩展标记语言（XML）：www.w3.org/xml

RFC 1831 远程过程调用规范

3 术语和定义

3.1

TODS标准模型 test open data service （TODS） standard model

TODS必备元素组成的模型，类似于《ISO/PAS 22720 ASAM Open Data Services 5.0》的基础模型。

3.2

TODS应用模型 TODS application model

针对于试验信息化行业的具体要求，基于TODS标准模型扩展而成的应用模型，类似于《ISO/PAS 22720 ASAM Open Data Services 5.0》的应用模型。

3.3

TODS 应用编程接口 TODS Application Programming Interface

TODS API定义的接口继承于ODS API所定义的接口。使用WebService进行访问调用。

3.4

TODS atf/xml

继承ODS ATF/XML的语义及格式定义，基于xml语言的格式文件。

3.5

自动化与测量系统标准化协会 Association for Standardisation of Automation and Measuring Systems (ASAM)

该协会最初由德国汽车制造商成立于1998年12月。其宗旨是为汽车数据和服务接口提供标准从而减少开发成本并增加信息流。负责制定数据模型、接口和语法规则标准，涉及ASAM AE 汽车电子类标准、ASAM CAT 计算机辅助测试类标准、ASAM 公共通用类标准各类应用。

3.6

ASAM开放式数据服务 ASAM ODS

是定义通用数据模型（用于数据的通用释义）、接口（用于模型管理、数据存储和数据检索）、数据语法和格式交换的一个ASAM标准。

3.7

基本属性 basic attribute

一个ASAM ODS基本属性是一条总括信息，用于指定一个基 $\Rightarrow$ 本元素的特点，基础元素用于创建 $\Rightarrow$ 应用元素（通过派生方式），最终信息存储在 $\Rightarrow$ 这些应用元素的实例中。基本属性描述从该基础元素派生出的所有应用元素的所有实例的有关信息。一个基本属性拥有一个名称、一个数据类型、一个非常明确的含义，还提供下列相关信息：

- 1) 所派生出来的应用元素是否必须指定各自的基本应用属性（由 $\Rightarrow$ 必选属性关键字指定），及；
- 2) 实例数据是否必须为该属性提供取值（由 $\Rightarrow$ 可选关键字各自的零基数指定）。

3.8

基础元素 fundamental element

一个ASAM ODS基础元素是ASAM ODS预定义的全部条目集合中的一个条目。每个基础元素拥有一个全部的 $\Rightarrow$ 基本属性的集合，还有一些扩展的具体的基本属性；此外，其还拥有 $\Rightarrow$ 使其与其他基础元素（或甚至自身）联系的基本关系，

3.9

基础模型 fundamental model

ASAM ODS 基础模型是一个全部的，独立于应用的逻辑数据模型，该模型定义了 ASAM ODS 语境下所管理的数据的结构。它由一个预先定义的 $\Rightarrow$ 基础元素的集合和一个这些元素必须遵守的规则的组合组成。

3.10

基本关系 basic relation

一个 ASAM ODS 基本关系是一条被用于指定一个给定的关系派生（ $\Rightarrow$ ）该基础元素与其他基础元素（或甚至本身）的关系的全部信息；基础元素被用来创建派生（ $\Rightarrow$ ）应用元素（通过派生方式），最终信息被存储在派生（ $\Rightarrow$ ）这些应用元素的实例中，基本关系描述从该基础元素派生出的所有应用元素的所有实例的相关关系。一个基本关系拥有一个名称、一个目标基础元素、一个基数、一个非常明确的含义，还提供下列相关信息：

- 1) 所派生出来的应用元素是否必须指定一个各自的基本应用关系（由派生必选属性关键字指定），及；
- 2) 实例数据是否必须为该关系提供取值（是相关实例）（由派生可选关键字各自的零基数指定）。

3.11

应用属性 application attribute

一个 ASAM ODS 兼容的应用属性是一条用来指定相应实例派生（ $\Rightarrow$ ）应用元素特征的信息。可以是一个 $\Rightarrow$ 基本应用属性，也可以是派生（ $\Rightarrow$ ）一个扩展的应用属性。它拥有一个名称、一个数据类型、一个基数、一个非常明确的含义（ $\Rightarrow$ 在应用模型语境中），并提供关于实例数据是否必须为其提供取值的信息。

3.12

应用元素 application element

一个 ASAM ODS 兼容的应用属性是应用模型内定义的条目集合中的一个条目，并用于具体的应用领域。它是从其中一个 ASAM ODS 派生（ $\Rightarrow$ ）基础元素派生出来的。因此，拥有基础元素的属性（ $\Rightarrow$ ‘基本应用属性’）和关系（ $\Rightarrow$ 基本应用关系）。一个应用元素可以拥有扩展属性（ $\Rightarrow$ ‘扩展应用属性’）和与任何应用元素的扩展关系（ $\Rightarrow$ ‘扩展的应用关系’）。

3.13

应用模型 application model

一个 ASAM ODS 兼容的应用模型是一个依赖于应用的逻辑数据模型，这种模型根据具体的应用领域之内的 ASAM ODS 规则定义所管理的数据结构。应用模型是一个元模型，它说明信息（用实例说明）最终是如何被构造的。

任何人都可以定义应用模型；然而，应该基于相同的应用模型构建在相同信息集合工作的所有的应用。

一个应用模型必须遵守 ASAM ODS 派生（ $\Rightarrow$ ）基础模型规则。它由一个派生（ $\Rightarrow$ ）应用元素集合组成，每个元素从一个基础元素中派生。

3.14

应用关系 application relation

一个 ASAM ODS 兼容的应用是一条用于指定与相应的应用元素给定的一个实例相关的其他实例的信息。它可以是一个派生（ $\Rightarrow$ ）基本应用关系，也可以是一个派生（ $\Rightarrow$ ）扩展的应用关系。它拥有一个名

称、一个目标应用元素、一个基数、一个非常明确的含义（派生在应用模型语境中），并提供关于实例数据是否必须为其提供取值的信息（是相关实例）。

3.15

逆关系 inverse relation

一个关系（派生（⇒）基本关系和派生（⇒）应用关系）指定一个给定的元素与其他（基本或应用）元素的关系。这些其他元素通过本身指定与给定的元素的关系（‘逆关系’）识别这种关系。

3.16

必选属性 mandatory attribute

这是一个关键字，在 ASAM ODS 派生（⇒）基础模型结合派生（⇒）基本属性或派生（⇒）基本关系中，用来指定派生（⇒）该基础元素派生的派生（⇒）应用元素必须包含一个相应的派生（⇒）基本应用属性或关系。如果关键字**必选属性**不存在，派生的应用元素不需要指定一个相应的基本应用属性或关系。

3.17

可选的 optional

这是一个关键字，在 ASAM ODS 基础模型结合派生（⇒）基本属性或派生（⇒）基本关系中，用来指定派生（⇒）派生的派生（⇒）应用元素的实例可以忽略相应属性或关系的值。如果关键字**可选的**不存在，一个应用元素的实例，拥有一个相应的派生（⇒）基本应用属性或关系的，必须提供该属性或关系的一个值。

3.18

基本应用属性 basic application attribute

一个 ASAM ODS 兼容的基本应用属性是一个派生（⇒）基础元素派生（⇒）相应的基本属性派生（⇒）相应的应用元素的属性。其数据类型、基数、含义、及其实例是否必须（或不比）为其提供取值，已经在 ASAM ODS 派生（⇒）基础模型中定义，并且不能改变（有几个例外）。其名称可在应用模型设计派生（⇒）阶段指定。

3.19

基本应用关系 basic application relation

一个 ASAM ODS 兼容的基本应用关系是一个派生（⇒）应用元素与另外一个派生（⇒）基础元素相应的基本关系的相应的派生（⇒）应用元素的关系。该基础元素被用来派生目标应用元素，其基数、含义、其实例是否必须（或不比）为其提供取值（是相关实例）已经在 ASAM ODS 派生（⇒）基础模型中定义，并且不能改变（有几个例外）。其名称可在应用模型设计派生（⇒）阶段指定。

3.20

实例 instance

一个 ASAM ODS 兼容的实例最后用来存储信息。它刚好实例化一个派生（⇒）应用元素，并存储派生（⇒）应用属性和派生（⇒）应用关系的值。它可以增加扩展属性（派生（⇒）实例属性）到应用元素指定的属性集合。

3.21

实例属性 instance attribute

一个 ASAM ODS 兼容的实例属性是一条被用来进一步指定派生（⇒）一个应用元素的派生（⇒）一个实例的特征的信息。它扩展了应用模型派生（⇒）定义的应用元素的属性集合。它拥有一个名称、一个数据类型、一个值、还有与单位的关系（可选）。这些可以在使用期间指定。不同的实例可以有一个不同的实例属性集合，即使它们实例化同一个应用元素。实例属性在小范围用户（或公司）中只有一层明确的含义。它们的含义往往不在应用模型的整个范围内定义，也不在 ASAM ODS 环境定义。

3.22**应用程序编程接口 application program interface (API)**

（应用编程接口）一个计算机系统、库或应用提供的接口，以处理其他计算机程序的服务请求，和/或处理与其他计算机程序的数据交换。TODS 指定采用 TODSAPI 作为标准的应用程序访问接口，同时提供了原 ASAMODS 的两个 APIs RPC-API（远程过程调用应用编程接口）和 OO-API（面向对象应用编程接口）作为参考标准。

3.23**ASAM传输格式 ASAM transfer format (ATF)**

（ASAM 传输格式）一个允许测量数据独立于数据库存储的文件存储系统。在 ASAM ODS 版本 5.0 之前，这个文件存储系统是经典 ASAM 传输格式（引用 ATF/CLA，目前发行版本是 1.4.1）。版本 5.0 引进 ATF/XML，ATF/XML 是 ATF 文件格式基于 XML 的版本。像其旧版对口，ATF/XML 有可能交换整个环境，甚至是不同计算机和环境之间的那些组件。同时，除了基于 XML 差别之外，没有其他差别，ATF/XML 及 ATF/CLA 是如此相似，以致被认为相同标准的两个不同“传输格式”。ATF/XML 能够传输所要求的结构和实例信息，还有各自的数据，并能够以标准化格式存储所有的这种信息。

ATF 还有可能被无法访问 ASAM ODS 服务器的系统，或无法执行 ASAM ODS API 的系统用来存储数据。

3.24**ASAM传输格式/CLA ATF/CLA**

ATF/CLA 是经典 ASAM 传输格式（ATF），由一组语法图和附加规则指定。AFT/CLA 文件是纯美国信息互换标准代码（ASCII）文件，只允许有几个特定字符。

3.25**ASAM传输格式/可扩展标记语言 ATF/ eXtensible Markup Language (XML)**

ATF/XML 是基于 XML 的 ASAM 传输格式（ATF），由一组 XML 架构文件和扩展规则指定。AFT/XML 文件兼容 XML，并使用标准的 XML 分析器/工具阅读和写入。

3.26**物理存储器 physical storage**

在 ASAM ODS 的意义上，物理存储是一个关系数据库的结构规范。关系数据库的结构允许在 ASAM ODS 服务器内存储所有相关信息。由于它是标准化的，整个 ASAM ODS 数据存储可以从一台 ASAM ODS 服务器移动到另一台 ASAM ODS 服务器，通过简单地移动数据库文件和数据库文件引用的所有二进制文件。此外，物理存储规范包括将海量数据存入单独的、高度紧凑的二进制文件（派生（⇒）混合模式服务器）的选项，因此，节省存储空间和也不用导入/导出。

3.27

EXPRESS 语言 EXPRESS language

EXPRESS 是 ISO 10303 (STEP: 产品模型数据交换标准) 中指定的一个数据说明语言。它被用来说明 ASAM ODS 派生 (⇒) 基础模型。EXPRESS 不仅仅提供说明结构数据的方法, 还提供一种指定限制和其他规则的形式化方法。这样, 在很多程度上, 形式化地说明了 ASAM ODS 基础模型。只有几个方面需要进一步的文字说明。

3.28

统一建模语言 unified modeling language (UML)

UML 是一个图形化语言的定义, 通常用于说明一个软件系统。它提供各种图形演示一个软件系统的不同方面。最著名的有:

- 1) 类图 (静态系统说明);
- 2) 序列图 (动态系统的使用说明);
- 3) 状态图 (一个系统的部分一般动态说明)。

3.29

可扩展标记语言 Extensible Markup Language Document (XML)

XML 是的文件存储语法、信息传输和处理的定义。XML 是 SGML 的一个子集, 它允许进一步指定在特定环境有用的语言。一个 XML 兼容的文件的主要特点如下:

- 1) 文件包含元素;
- 2) 每个元素通常由一双标签 (开始标签和结束标签) 组成, 这些标签环绕着该元素的内容;
- 3) 元素内容可以由纯文本或扩展元素组成。

3.30

万维网路联盟 world wide web consortium (W3C)

W3C 是一个国际联盟, 旨在开发网页技术。这包括形成准则和标准 (所谓的 W3C 建议), 如, XML 或 SOAP。

4 概述

试验测试开放数据服务 (TODS) 是在 ASAM ODS 的基础上, 采用与 ASAM ODS 一致的技术方法、基础数据模型、交换格式, 扩展形成了 TODS 标准模型, 继承 ODS ATFX 标准, 参考 ASAM 00-API 重新定义了基于 W3C Service 的 TODS API。

4.1 目的

4.1.1 定位

ASAM ODS 主要关注数据的持续存储和访问。ASAM ODS 描述了信息的物理存储以及服务接口。为了完成相应的操作, 测试环境中的任何组件都可以使用服务接口来存取或者访问数据。

使用这些接口的典型组件包括:

- 采集车辆、发动机等数据的数据采集系统;
- 用于运行测试过程的测试控制系统;
- 寻求标定参数最优集合的优化工具;

- 为工程师和决策者提供数据展示的分析 and 报表工具；
- 支持研发活动的评价工具等。

存储和访问的典型数据包括：

- 测试控制系统的测试程序配置数据；
- 测试设备及待测单元的描述数据；
- 测试台上发动机上的测试数据，风洞中车辆的测试数据等；
- 在仿真运行中产生的数据；
- 测试评价工具产生的数据；
- 发动机或车辆部件的标定数据。

ASAM ODS 接口访问的常规方式是通过 OMG（对象管理组织）制定的 CORBA（公共对象请求代理结构）标准，该标准要求采用面向对象的系统设计。

CORBA 并没有明确接口将会提供何种服务；这要由具体的服务提供商来决定（在本例中提供商是：ASAM ODS）。相反，CORBA 明确了要使用接口的组件如何确定、发现和连接接口。因此，如果必要，CORBA 关注的是网络透明传输。

CORBA 是一个开放标准，在各种大型工商业程序中使用广泛。ASAM ODS 确定了面向对象的（OO-）API 接口，可以使用 CORBA 方法实现。然而确定使用的接口并不仅限于采用 CORBA 方法，在将来技术更先进的服务器也可能提供同样的接口。

由于第一版 ASAM ODS 接口规范采用 RPC 作为通信方法，许多应用至今仍使用 RPC-API 接口。在将来，为了保持兼容性，ASAM ODS 仍然会支持 RPC-API。尽管以面向对象的 OO-API 将是主流。

由于实现持续的数据存储一般要求网络传输通畅，ASAM ODS 的实时性能并非最优。然而，ASAM ODS 也适用于实时操作，与时间相关的操作取决于一个具体 ODS 服务器的设计/实时和所采用的对象请求代理（ORB）。

ASAM ODS 规定了接口，但最终的数据持续存储的方式并不唯一。使用某种面向对象或者是关系数据库（使用共同的数据库模型）可能是合理的，但决定权取决于 ODS 服务器实施方。

尽管 ASAM ODS 在 ASAM 系列标准中主要面向数据的存储和访问，但它并未规定文件和数据库格式。如果必要，其它 ASAM 标准也可以这样做。例如：在操作中其他标准很可能要求在本地上存储控制系统的配置和测量数据；或者它也可以要求采用国际通行的数据格式与监管机构进行数据交换（如 ASAM MCD 方面要求）。

TODS 参考原有 ASAM ODS OO-API 的定义，采用 WBSERVICE 及 XML 规范，重新定义数据访问接口，TODS 继承 ODS 的特性，更加适用于国内更多的厂商现有的系统或实现方式。

4.1.2 现有系统的缺点

目前汽车工业中的测试、评估及仿真环境系统通常都采用自己的专用格式存储数据。这些格式对配置描述（被测单元、测试序列、测试设备等）和结果存储方式（数据库，二进制文件等）的说明各不相同。

（1）用户观点

过去几十年中，安装程序越来越多。早期的供应商被取代，新供应商引进了新的存储方式和另一种

不兼容的存储格式，因为这些系统是根据各自需要独立研发和升级的，存取这些不同种类的系统需要各类不同的接口适配器和转换器。底层系统越来越复杂产生了高昂、专用、孤立的系统，因此，解决方案越来越不经济。

这与汽车、航空、航天、建筑、电子等行业想要开发和使用简单、易于管理的应用系统，为公司内部其他部门和供应商提供宝贵信息的目标是背道而驰的。

此外，即使替代供应商采用了技术更先进的系统，他们往往也没有考虑到要将现有系统和他们的新系统连接起来所需要花费的后续工作量有多大。

(2) 厂商观点

由于每个客户都有专用的各种系统，引入新产品都要与这些系统交互，所以，新产品的研发工作多半花费在兼容性上。这不仅工作量大，而且所需知识（大多是系统运用所得的经验）起着关键作用。因此，与现存环境连接的能力需要专门人士支持。

而且，不同的客户要求不同，发布的产品针对特定客户，版本控制成为一个大难题。

提出新理念和解决方案的新公司，很少有机会落实到产品上，它们要么在接口上投资过大，要么在产品实施当中作用甚微。这就阻碍了汽车、航空、航天、建筑、电子行业的创新，最终损害了企业竞争力。

4.1.3 益处

数据存取接口标准化的主要目的是减少项目的成本和风险，为数据存储和使用提供一个稳定可靠的基础。使用标准接口和通用结构使上述异构环境中系统整合工作量更小，数据的交换更加简便。ASAM ODS 标准的基本要素见图 1。



图1 ASAM ODS 标准的基本要素

为克服上述问题，ASAM ODS 提出了：

——通用数据模型（基础模型）提供清晰、完整的数据存储。这个基础模型能够视为初步的数据分类，在这里将语义信息添加到数据中，最后能够允许不同系统用相同方式解释相同的数据。数

据模型能够满足多个应用领域的需要,也能够通过从基础模型派生应用模型适应某个系统甚至项目的特殊需要;

- 接口(应用编程接口, APIs),通过标准方式存取与 ASAM 兼容的系统和工具的数据;
- 接口能够创建和存取某个实际应用模型(元信息)的自我描述。这就允许系统对各种不同的 ASAM ODS 数据进行操作;
- 为应用广泛的关系数据库提供数据模型。这个数据模型明确了数据的物理存储。还允许用同一数据库管理系统(DBMS)在各系统之间进行数据库文件的交换。最后(如果出现某个专用 ODS 服务器应用不可用)这个数据模型甚至能够在不同平台,不同的数据库管理系统之间通过 SQL 命令很方便地进行存取;
- 一个标准的,易于操作的,基于文本的数据交换格式(ASAM 传输格式——ATF)。方便不同系统和不同平台之间进行 ASAM ODS 数据(包括元信息)交换。ASAM ODS 5.0 及更高版本支持经典的 ATF(ATF/CLA)格式及基于 XML 的 ATF/XML 两种格式;
- 一组能够反映 ASAM ODS 典型应用场景的应用模型。这组模型能够非常方便地,在不同公司数据之间建立映射。(关于更多的应用模型的信息,请参考 ODS 的原标准。)

该标准的另一好处体现在对产品质量所起影响上。确定的标准运行测量产品实施。认证程序确定后,检测产品互操作能力的另一种办法是做交叉测量,这种测量已经采用并产生了非常好的结果。两种方法都可以确保产品符合标准。

最后,ASAM ODS 提供整合汽车产品整个生命周期数据的机会。尽管在研发,生产和售后领域的数据种类彼此不同,ODS 仍能够存储(每种采用各自专用的数据模型)和检索它们,因此保存了数据项的含义。而且 ODS 接口能够采用工具组合各种领域的信息,分析它们的依存关系,生成整体报告等等。

4.2 技术方法

ASAM ODS 从五个重要方面来规范数据存取:

- 数据模型;
- 接口;
- 物理存储;
- 传输格式;
- 应用模型。

4.2.1 数据模型

ASAM ODS 规范最主要和最重要的就是其数据模型。

数据项的交换一般有几种不同层次:

- 两种系统或者软件包很可能根本无法交换数据项。这种情况发生在如系统 A 期望交换数据项采用 4 字节的浮点数,而系统 B 则期望采用 2 字节的整数值进行数据交换。缺点很明显:如果不改变其中一个系统(如:软件系统),无法建立自动数据交换;
- 两个系统或者软件包能够以数字的方式交换数据项,因为两个系统交换数据项的数据表述完全相同。这主要是由于某种标准化的结果,确定了统一数据格式。例子之一就是应用广泛的 CSV(逗号分开值)格式,在这种格式中数据项由 ASCII 码(美国信息交换标准码)表示,后续

数据项用逗号相互隔开。系统 A 产生了这样一个 CSV 文件，而系统 B 能够读取并且清楚其中所含数字的含义。缺点仍然很明显：如果不了解数据项的含义，数据的值就无意义；

- 类似的两个系统或者软件包能够以带名称的数字方式交换数据项，比如，因为它们使用相同的数据库。在这种情况下，系统 A 在某表格某个位置，使用某个行名或列名来存储数据项的值，通常是通过数据库接口实现的。系统 B 可以使用同样的数据库接口，按照数据的名称或者在表格的位置识别数据，从而取回其值。缺点不是很明显，看起来数据项由其值和名称做出了完整描述。然而，只有系统 B 理解了数据项名称含义时，它的值才有意义。这就需要对数据项的命名做额外的约定。而且诸如“这个值的单位是什么？”“这是否是数据项最新的一组可用值？”等等问题仍然没有解决；
- 某些数据交换解决方案通过规定所有信息来克服这些缺点。例如某些串口协议说明或者某些相当固定的数据库模型。虽然这可以最大程度提供信息，但是这种方法缺乏灵活性，只能满足某项特殊任务的需要。

ASAM ODS 规范试图寻求最佳的综合方案。

一方面，本标准可适用于各种应用领域。从小型车辆零部件获取的数据，可以和整车数据一样的处理方式存取。从海外获取的数据（不论语言和单位方案有何不同），可以和现场的数据一样使用。研究项目中的数据和生产过程中的数据甚至是售后的信息都可以互相交换。在这些过程中所有系统能够根据 ASAM ODS 数据模型存储和检索数据。而且对于已有的大量数据，ASAM ODS 将提供数据集成方法，将其整合到新的管理方式中。

另一方面，存储信息要保持有价值，而且包含足够的语义支持自动检索和数据分析。为了解一个物理值的真实情况，数据值（数字）和相应的单位之间应该存在某种关系。数据项与数据提取的车辆或部件存有某种关系，还有与提取数据的设备和数据提取者等等之间都存在某种关系。

这最终形成了当前的 ASAM ODS 数据模型。它分为“基础模型”和“应用模型”。

基础模型是一组预定义的基础元素集及它们之间的基本关系集。

每个基础元素表达一类信息。例如，**AoUnit** 是表达一个物理单位（如牛顿、开尔文等）信息的基础元素，**AoMeasurementQuantity** 是表达一个被测物理量（如力、温度等）信息的基础元素，…等等。

每个基本关系代表了两个基础元素之间一种具有特定语义的联系。例如 **AoMeasurementQuantity** 与 **AoUnit** 两元素关联时，它们的关系能够明确所有可能的单位中哪个是当前量的单位。

基础模型对于 ASAM ODS 的各类应用而言是唯一的。

应用模型与具体应用有关。风洞试验系统与发动机疲劳试验系统采用的应用模型是不同的。

应用模型明确了应用使用哪些元素。例如，发动机试验会测量到温度、压力和力，且要使用 SI 单位开尔文，帕尔卡和牛顿。这六个实例是被测量的量（因而属于基础元素 **AoMeasurementQuantity**）或者是单位（因而属于基础元素 **AoUnit**）。它们每个都是单独的应用元素（派生自相应的基础元素），因而，每个使用的量和对应的单位之间存在关系。

尽管每个应用都可有自己的应用模型，但应用模型中的所有应用元素都知道各自属于哪个基础元素类型。因此任何存取此类应用元素的软件不仅了解应用元素的名称和数值：还了解应用元素代表信息的类型以及数值目前所对应的单位等等。

关于数据模型的描述详见 4.5 节。

TODS 在原有 ODS 基础数据模型的基础上，针对国内试验测试信息化的特点，定义了 TODS 标准数据模型，使其更加适用于国内试验测试信息化要求，并具有一定的前瞻性。详细描述参见第 8 章 TODS 标准模型。

4.2.2 接口

ASAM ODS 规范的第二个方面是其接口，通过接口进行数据存取的方法见图 2 所示。

显然，存储和检索信息需要某种物理存储介质（如磁盘，磁带等等）。同样，信息存储也有不同层次。

通常操作系统提供一个基本接口从存储介质文件当中读取或者写入单个信息项。这是一个相当低级的接口，要求文件中的每一字节都有唯一标记。否则，尽管系统 A 可以用某种方式写入信息，系统 B 将很可能不能读出来。除了需要大量的描述说明，仍然可能出现系统实施者误解或曲解说明，产生不兼容文件的情况，特别是在文件当中数据项存在依赖关系时。

更加方便的方式是使用 DBMS（数据库管理系统），让其来存储数据库信息。今天，关系数据库的应用相当广泛，DBMS 系统提供的标准的 SQL（结构化查询语言）接口能够存取数据库存储的信息。这种方法中各应用不用关心它们数据的准确文件表示，它们只需使用 SQL 指令来存储和检索数据项，DBMS 系统负责系统 A 存储的数据能被系统 B 检索出。

关系数据库的数据以表的形式有序存储，表之间互相关联。因而描述说明集中在对表格及表格之间关系的确定。ASAM ODS 采用的是此种方式（见下文“物理存储”描述）。

然而，单一面向数据库标准的缺点之一是没有纳入数据库之前存在的数据。如果没有转换/导入数据库中就不能作为 ASAM ODS 数据进行存取。

另一缺点主要是 DBMS（数据库管理系统）不知道 ASAM ODS 的具体细节，因而不能确保符合 ASAM ODS 规范的数据库内容的一致性。

此外，DBMS（数据库管理系统）一般要求一定量的管理工作（需要充足的人手），还涉及许可费用。这给小规模的应用造成了困难。

最有前景的解决方案是提供一个与 ASAM ODS 数据模型规范紧密配合的专用接口标准。接口将客户应用与服务器应用区分开来。服务器应用保存和组织数据，客户应用存储和检索数据。如果接口能够照顾到 ASAM ODS 具体特性，那么可以预期客户端和服务端之间能够实现优化合作。

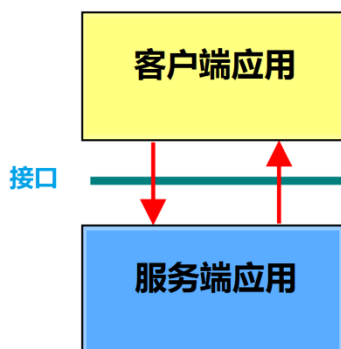


图2 通过接口进行数据存取

这是 ASAM ODS 规范采用的理念。

服务器提供存储和检索数据的接口。客户端可以使用这些接口。服务器内部存储数据的方式完全由服务器决定。服务器实施质量会影响到它自身的表现，比如性能、整个数据量以及客户端并发存取，等等。

由于数据存储时完全封包，服务器应用可以决定持续存储是采用关系数据库还是面向对象数据库。或者如果对所针对的应用领域更为合适，而用户也不会察觉到功能上有何差异的话，可以采用平面文件存储系统。而且，封包支持建立可以处理先前存在数据（不论以何种存储格式出现）的服务器应用，并像 ASAM ODS 正常数据一样提供给客户端应用使用。

5.0 版本和更高的 ASAM ODS 规范包含两个不同类型的接口：

- 基于 PRC 技术的程序接口或者称之为 PRC-API 接口；
- 面向对象接口或者称之为 OO-API 接口。目前采用基于公共对象请求代理结构的构造，但在将来会扩展到其它技术领域；
- ASAM ODS 规范将来还会支持 PRC-API 接口，尽管目前 PRC-API 接口已停止更新，预计也不会做进一步修订。

TODS 标准，参考了 ODS OO-API 的功能要求，基于 WWebService 和 XML 规范，重新定义一套数据访问接口，其更符合国内现有的试验测试信息化产品的实现方式，也可以减轻现有产品实现 TODS 标准的难度和成本。

4.6 节将会对上述接口进行详尽描述。

4.2.3 物理存储

ASAM ODS 规范的第三个方面是物理存储格式的说明。

通常主要存在这几种物理存储格式：（关系）数据库存储，独立文件集（平面文件）存储，和混合模式存储。这种所谓的混合模式存储主要指的是大量数据存储在文件中，而这些文件的描述性数据存储在数据库里。

如今，ASAM ODS 已经确定了关系数据库内部应当如何建构，以 ASAM ODS 兼容的方式存储数据。确定的内容包括：建立哪些表，存储哪些信息，哪些列是主键（因此必须唯一）等等。

没必要仅仅为了标准化数据存储概念，确定物理存储的格式。而且，ASAM ODS 规范的主要好处只需实现数据模型和接口即可体现出来，不需要对物理存储格式进行任何规范（产品仍然会无缝的交互）。

然而，对物理存储格式进行规范也有好处：

——首先，如果不同的服务器应用使用同个一平台上的同一种 DBMS，如果它们之间必须进行数据交换的话，数据库文件可以直接进行拷贝。因而毋须导出-导入程序就可实现从某一服务器到另一服务器之间的数据转换；

——此外，常用基于 SQL 的工具（如浏览器，分析器和报表生成器）都可绕过服务器应用程序存取数据。这些工具随处可以买到，且易于使用，不是很昂贵，而且通常员工都已经熟悉它们的使用。尽管 ASAM ODS 并没有对可用数据进行专门的解读，这些工具仍然足以应付（特别是小型任务）。在这种情况下不需要 ASAM ODS 服务器应用许可和安装；

——最后 ODS 服务器自身能够进行数据交换，不会造成数据丢失，也不需要将数据从原始服务器导出再导入新服务器。

物理存储将会在 4.7 条进行详细描述。

TODS 继承 ODS 物理存储的规范，但不作强制实现。

4.2.4 传输格式（ATF）

ASAM ODS 第四个方面的内容是传输格式，即所谓的“ASAM 传输格式”——ATF。这种传输格式可用于所有平台和操作系统，而且能够允许不同系统间传输数据，并不会造成信息丢失。

因为几乎所有的系统或平台都支持普通 ASCII（美国信息交换标准码）文本作为信息表述方式，所以 ASAM ODS 决定采用 ASCII 文本来确定 ATF 传输模型。其确定的语法规则允许传输数据模型中所有信息。

因为 ASCII 的特性 ASAM ODS 传输模型文件会非常之大；因而标准允许将信息分成几个单独的 ASCII 文件，甚至允许将大量数据存储在单独的二进制文件中，这些二进制文件的结构信息将会保存在 ASCII 文件。

预计 ASAM ODS 服务器应用允许导入 ATF 文件，也允许导出全部或部分的信息到 ATF 文件。这也适用于 ASAM ODS 客户端，可以采用 API 接口导入导出 ATF 文件。

ATF 的应用主要在传输数据：例如将独立的车内测量系统的测量数据转传输到部门 ODS 服务器数据，或者在 CORBA 网络不可用时，将服务器的部分（或者全部）数据从一平台传输到另一平台。因为采用 ASCII 文本，ATF 能够用于手工分析数据和修改具体数据项。

ATF 的另一好处体现在它能够很容易由任何应用建立。不需要知道 CORBA 或者 RPC，也不需要软件接口或者数据库有经验。因而它是任何产品与 ASAM ODS 规范兼容的第一步也是最容易的一步。

5.0 和更高一级的 ASAM ODS 规范包含两个不同的传输格式：

——第一个确定的就是所谓的经典 ASAM 传输格式（ATF/CLA）。这种格式已经在服务器和客户端应用多年，是基于 ASCII 数据交换的一种稳定可靠的传输格式；

——随着在众多领域的应用，XML（可扩展标识语言）已经成为描述数据存储的基本语言，ASAM ODS 采用了 ATF/XML 传输格式。这种传输格式以 XML 语言为基础，确定文件内容，允许符合 ASAM ODS 规范的应用之间进行数据交换。

ASAM ODS 规范将来还会支持 ATF/CLA，尽管目前 ATF 类型已停止更新，预计也不会做进一步修订。

在 4.8 节中将对 ATF 进行详细描述。

TODS 继承 ODS 传输格式规范，并建议采用 ATF/XML 作为 TODS 标准实现的优先形式。

4.2.5 应用模型

ASAM ODS 5.0 版及更高版包括标准化应用模型，让实施者可以非常容易地实施非常普遍的应用模型。在本章发布之际，TODS 应用模型已经可以采用，详见 4.9 节中描述。

实际上，TODS 标准数据模型，就是一种基于 ODS 基础模型通过扩展定义而来的一种应用模型。

4.3 对产品的影响

ASAM ODS 将会对现存的能进行测量，仿真和执行控制的软件产品或者是组织，分析数据提供报表

的软件产品产生下列影响：

- 大型的自动化，数据获取或分析产品将会进行扩展，使用 ODS 接口连接 ODS 服务器。它们可以使用 ODS 服务器存储配置信息和测量结果；
- 功能小的数据获取产品将只用 ATF 文件输出数据，因为使用 ASAM ODS 接口连接服务器对它们而言花费太大；
- 仿真工具可以扩展，它们不仅可以自己内部的仿真程序进行仿真，还可以基于先前测量所得的数据进行仿真；通过 ODS 接口可以很容易单独存取这些数据；
- 校准和优化组合包将整合数据获取系统实际测量结果和之前获取并保存在 ODS 服务器中数据（通过 ODS 接口读取）。这样做可以减少整个测量所耗时间，增加数据重复使用率；
- 报表生成和展示工具现在可以通过 ODS 接口以标准方式任何测量环境数据，因而操作也更加容易。对不同设备测量所得结果的比较将更容易，并且这种比较结果与测量设备无关。

ASAM ODS 将会为新产品提供以下几种可能：

- ASAM ODS 服务器应用将会被研发出来。这些应用范围广泛，从小型的只用于一定量测速器的基于 Windows NT 小型解决方案到能够管理不同信息源数据的面向大企业的系统。有些应用会采用关系数据库和专用物理存储格式，其它应用则可能采用自己专属的格式（例如，使用面向对象的数据库或者只是一组“平面文件”）；
- 数据浏览器和编辑器可以迅速浏览整个 ODS 数据存储的内容。某些工具还可以将几个 ODS 服务器数据组合成一个视图，二另外一些工具可能只能处理 ATF 文件；
- 标准还将提供连接器（比如通过使用 VBA），能够将现存的通用程序包如 EXCEL, MATLAB 与 ODS 数据存储连接起来；
- 分析和报表工具将提供 CORBA 或者 RPC（软件）组件。它们非常轻巧，致力于专门的任务，并且能与框架结构内其它组件结合按照所需功能生成单独工具。

4.4 实施的技术水平

除了需要好的软件开发技巧和理解 ASAM ODS 规范，理解并实施 ASAM ODS 规范还需要掌握以下关键技术：

- ASAM ODS 数据模型采用 STEP EXPRESS 语法确定。熟悉这种语法大有好处。尽管 ODS 元素之间的关系有非常容易理解的图形描述，这些图形并不包含所有信息。熟悉数据建模工具、实体关系模型等都非常有益；
- ASAM ODS 接口使用 CORBA IDL 或者 RPC IDL 作为接口定义语言。理解接口的功能只需要掌握它们既可；
- 将它们作为客户端，要求在 ODS 服务器（一些已经由 ASAM 成员提供）实施 CORBA ORB（只在使用基于 CORBA 的 OO-API 接口时才需要，ORB 来自不同供应商）时会使用到上述两种语言，还要求掌握一些操作远程对象和本地代理的经验；
- 采用它们实施 ODS 服务器是一项更加艰难的任务。需要熟悉 CORBA 服务器实施，或者熟悉 RPC 接口，熟悉 RPC 服务器实施；
- 当前确定的物理存储基于关系数据库。要求专门的数据库管理系统才能使用 ASAM ODS 规范。

此外，还需要 DBMS 的管理经验和 SQL 的使用经验；

——实施 ATF-CLA 导出导出文件并不需要什么专门知识，但实施 ATF/XML 则需懂得 XML 技术。

TODS 标准的实施，除了上述对 ODS 标准实施的关键技术外，最重要的补充是：

——凡是需要实现 TODS 标准的系统，可以直接使用 TODS 的标准数据模型或在其基础上进行扩展，形成新的 TODS 应用模型。

——TODS 标准实施，优先采用 TODS API。

4.5 数据模型

本节将对 ASAM ODS 规范的第一个方面进行更为详尽的描述。但是，要应用此方面内容还需要学习本标准的第 5 章和第 7 章。

ASAM ODS 数据模型分为基础模型和应用模型，两者都描述了所存储数据的结构，真实值最终存储在应用元素的实例中。数据模型因而分为三层，见图 3 所示。模型中的对象将会在以后章节中论述。

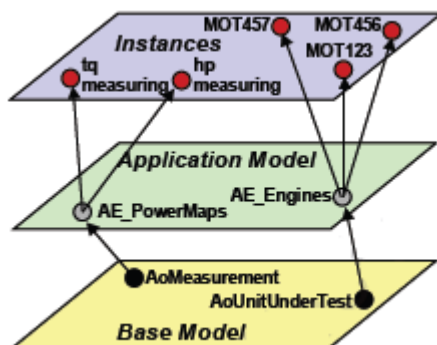


图3 基础模型、应用模型和实例的关系图示

4.5.1 基础模型

基础模型完全由 ASAM ODS 确定。它由一组基础元素和元素之间的基本关系组成，见图 4 所示。

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

图4 ASAM ODS 基础元素表

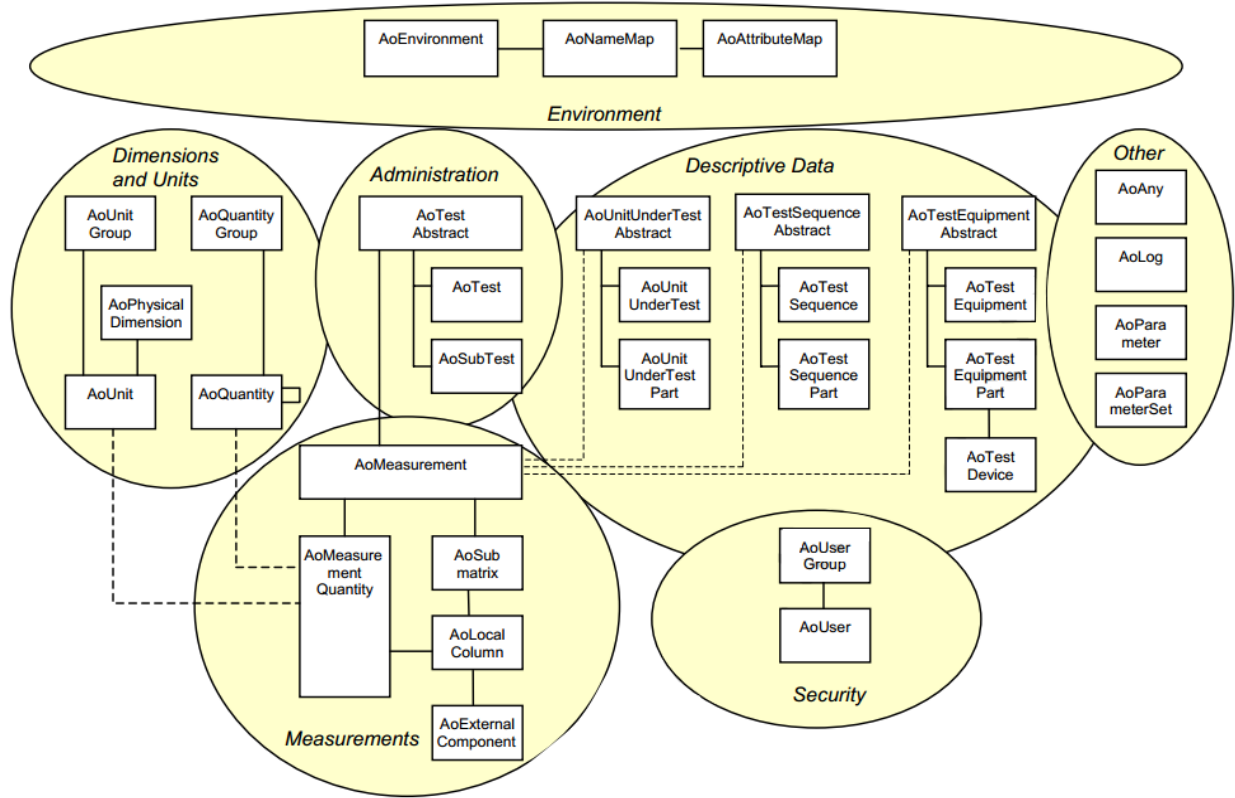


图5 ASAM ODS 基础元素集合图示

基础元素有点像信息分类。图 5 描述了目前确定的基础元素类型以及它们是如何分组的。前缀 Ao 是“ASAM ODS”的缩写形式。

基本关系是这些基础元素之间的联系，每一个联系代表一种具体意义。

AoTest、AoSubTest, and AoMeasurement 元素用来组织测量和相应的输入输出数据。这三个基础元

素让应用构建一个测量的层次结构并处理测量的结果。

AoSubMatrix 和 AoLocalColumn 是用于存储测量结果的基础元素。ASAM ODS 假定测量结果以表（子矩阵）的形式存储，其中列代表测量量的值而行则是后续测量点的值。

一个测量可以包含几个这样的子矩阵；这些子矩阵结合在一起组成一个大型的测量矩阵（这个矩阵有大量空白区域，这就是存储测量结果是通过大量子矩阵而非一个测量矩阵进行的原因）。

AoQuantity 是记录与数据库中某些或者全部测量相关的物理量的信息。AoQuantity 与 AoUnit 相连，这意味着每一个量都有一个默认的单位。

AoMeasurementQuantity 代表在测量中使用到的量。（例如测量结果可以用到的地方）。AoMeasurementQuantity 分别与 AoMeasurementQuantity 及 AoUnit 相连。

AoUnit 和 AoPhysicalDimension 组件用于记录结果数值（一般数字）和对应单位之间的关系信息，因而非常注重结果数值的准确性（即使在单位系统发生改变时，比如英制和公制之间转换）。

成对元素 {AoUnitUnder Test, AoUnitUnderTestPart}, {Ao TestEquipment, Ao TestEquipmentPart}, 和 {Ao TestSequence, Ao TestSequencePart} 记录的信息是：

- 测量的内容（UnitUnder Test, UUT）；
- 测量所用设备（TestEquipment, TE）；
- 测量进行的步骤序列（TestSequence, TS）。

这些成对元素的基本关系由基础模型按照真实应用的层级树确定，其中，根元素是 UUT，它有一个或多个 UnitUnder TestPart 与之相连，而这些 UnitUnder TestPart 同样由一个或多个 UnitUnder TestPart 与之相连，如此类推。

AoUser 和 AoUserGroup 是用于安全方面的基础元素。

ASAM ODS 规范一直在不断发展，ASAM ODS 5.0 版和更高版本包含以下新的基础元素：AoLog, AoParameter, AoParameterSet, AoNameMap, 和 AoAttributeMap。

AoLog 是用于记录日志。

AoParameter 和 AoParameterSet：用于避免信息存储的冗余。比如存储几个元素只用一次的信息。应用属性和实例属性会使用替代数据，因而产生更多数据。

AoNameMap 记载有某个应用元素（例如语言版本）任何数目的别名。这个清单能够允许应用软件进行语言版本的切换。

AoAttributeMap 确定某个应用属性任何数目的别名。

最后，AoAny 是一个没有特定含义的基础元素。它可用来存储所有与其它基础元素的含义不符的信息。但是它的使用应当十分小心，应该优先使用其它基础元素存储信息。

基础元素之间除了有特定的基本关系之外，每一个基础元素还有一组定义明确的基本属性。它们是详细描述元素的特征值，基本属性可以是必选或可选项。

每一个基础元素至少必须有的基本属性是“名称”和“ID”号，也可有“版本”，“版本日期”和“说明”等属性。

某些基础元素有专门的基本属性，如 AoMeasurement 就有“测量开始”和“测量结束”基本属性，AoMeasurementQuantity 就有一些额外的基本属性，“数据类型”，“类型-大小”，“秩次”和用于处理存储测量数量数据的“尺寸”。

4.5.2 应用模型

应用模型必须使用基础元素来构建实际应用通过 ASAM ODS 存储或检索的信息的模型。构建应用模型意味着确定一组应用元素，每一个应用元素都与某一基础元素相关联，这样每一个需存储的信息项能够保存到某一应用元素中。

图 6 简要显示了示例应用模型和基础模型相关部分关系。

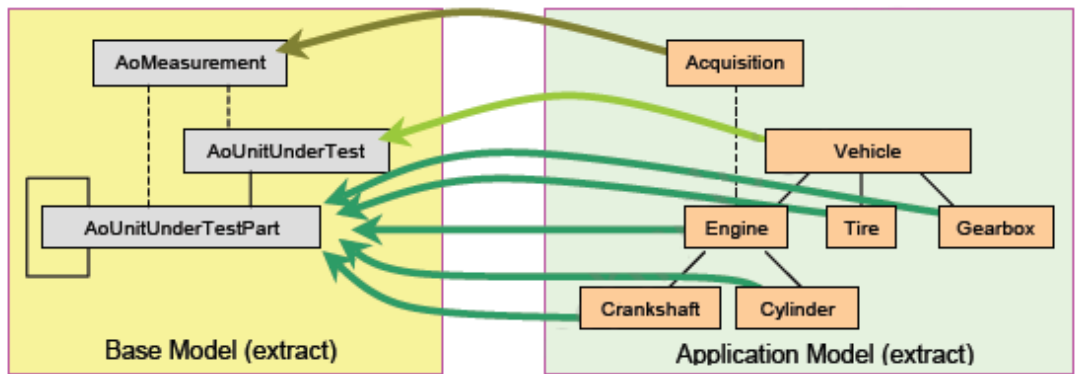


图6 从基础模型派生出来的的应用模型示例

基础模型简图显示的是 AoMeasurement、AoUnitUnder Test、AoUnitUnderTestPart 这几个基础元素。它们之间的线代表的是基本关系。

关系的基数（在上图中未显示）如下：

- [0, n] 代表 AoMeasurement 的所有关系，
- 一个 AoUnitUnderTest 和一个 AoUnitUnderTestPart 可以有[0, n]个 AoUnitUnderTestPart；
- — 一个 AoUnitUnderTestPart 可以属于一个到多个 AoUnitUnder Test 或者 AoUnitUnderTestPart。

应用模型简图显示的是某种可能的设置，在此例中，是专供引擎测量用设置。应用模型的设计者做出了以下决定：

- 称每次测量为“Acquisition”因而在应用模型中设立了“Acquisition”应用元素，它属于基本类型 AoMeasurement；
- 认为尽管主要是对引擎进行测量，车辆及车辆其它部件的信息也与测量相关（例如为了后期处理需要）。因而他将应用元素“Vehicle”作为他测量对象描述的根元素，它属于基本类型 AoUnitUnderTes；
- 认为（至少是为了处理需要）车辆由三个不同部分组成：引擎，变速器和轮胎。因而他在应用模型中设置了三个 AoUnitUnderTestPart 类的应用元素。每一个应用元素只与引擎相关；
- 认为（也是为了处理需要）引擎自身也是由两个不同的部分组成：曲轴和汽缸。因而应用模型中多设置了两个。AoUnitUnderTestPart 类应用元素。每个元素也是与引擎相关。
- 认为获取行为与收集数据的引擎相关。“Acquisition”需要知道安装了那个引擎，而引擎也需要知道“获取”了哪些信息；
- 按照这些想法就产生了图 7 的应用模型简图。

除了基础元素之间具体的基本关系会在应用模型当中再次以应用元素之间关系出现之外，应用模型的设计者也可以在应用元素之间建立新的关系(特别是应用元素之间的关系在相应的基础元素之间关系不存在时)。因而整个应用关系的组成包括已经存在于基础模型中的关系，以及后来对基础模型进行增加和扩展产生的关系。

应用属性的情况也是如此：每一个应用元素都有一组确定其特性的属性。必选的基本属性也必须出现在每个对应的应用元素中，而可选的基本属性可以出现在应用元素中。

除了这些基本属性，应用模型的设计者也可增加没有出现在对应的基础元素应用属性中的应用属性。这种情况经常出现在基本属性不能确定的时候，它们在可能出现的应用元素可能发生变化的时候尤其有用。

在上例中，设计者可以考虑将应用属性“weight” 增加到应用元素“Vehicle” 上，增加应用属性“number_of_cylinders”到应用元素“Engine”上，增加应用属性“bore”，“stroke”和“spark_plug_type”到应用元素“Cylinder” 上等等。

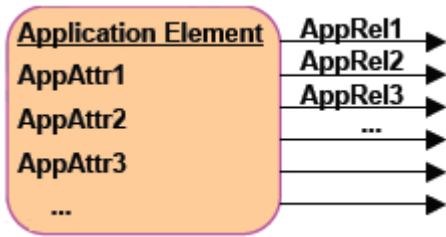


图7 应用元素

图 7 简要显示了作为应用模型构成模块的应用元素：

- 每一应用元素与一基础元素相对应（“属于基础元素类型”）；
- 每一应用元素都有一个进一步确定其特性的应用属性，一些属性在对应的基础元素中已确定，另一些则是后来添加的；
- 每一应用元素与其他应用元素间存在着应用关系，一些关系已经在基础模型中以对应的基础元素关系形式确定，而一些则是后来添加的；

只有考虑到这些，才能说 ODS 数据模型的几个方面，数据传输和数据含义是一体的：

- 应用元素的含义通过对应的基础元素获得；
- 应用属性意义是通过最终与之对应的基本属性获得，即使原本并不属于某个基本属性；它们有某种意义是因为它们属于某个有意义的应用元素；
- 应用关系意义通过最终与之对应的基本关系获得，即使应用关系没有对应的基本关系，它们有某种意义是因为它们属于某个有意义的应用元素。

4.5.3 实例

一旦数据模型设计出来，数据存储和检索的结构就确定了。基础模型和应用模型都不会存储任何(测量性或是描述性的)数值。它们只是提供了解数据存储结构的“元信息”。任何时候要存储真实值，首先要从应用元素派生出一个实例。

实例能够存储每一个与应用元素相对应的应用属性值。此外可以添加在应用元素中没有出现过的新的实例属性。

实例可以存储其它应用元素实例之间的关系（标识符）；应用元素 AE1 和 AE2 之间的每一个应用关系可以产生一个或多个在 AE1 实例和 AE2 实之间的关系。此外还可以添加在对应应用元素的两个实例之间不存在的新关系。

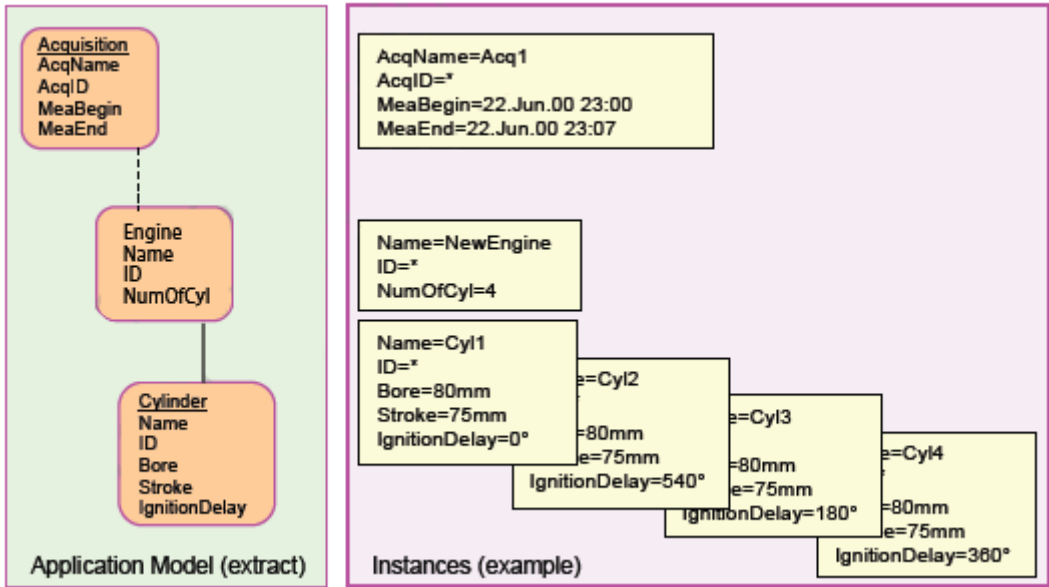


图8 数据存储实例及与应用元素的关系

按照图 8 给出的应用模型，测量系统可以建立实例（ID 属性值由 ODS 服务器设置），如下文所述：

- 一个应用元素 “Engine” 的实例；
- 四个应用元素 “Cylinder” 的实例；
- 一个应用元素 “Acquisition” 的实例。

而且，测量系统会为每一个与 “NewEngine” 相连的汽缸设立一个关系属性，设立四个分别将引擎与 “Cyl1”、“Cyl2”、“Cyl3”、“Cyl4” 相连的关系，一个将引擎与 “Acq1” 实例相连的关系和一个将 “Acq1” 与 “NewEngine” 相连的关系。

为了进行每一个后续测量，应用元素 “Acquisition” 要创建新的实例，实例要有新的名字，新的（服务器提供）ID 以及从测量始到测量末的具体数值，而且要与引擎和汽缸的同一实例相关。

（事实上，测量系统将要更多工作。例如，它要创建最终存储样本数据的应用元素实例。那些应用元素对应的基础元素类型是 “AoSubmatrix” 和 “AoLocalColumn”，在这个小型示例中没有包含在内。而且测量系统还要为测量量，测量单位和测量设备等等建立实例。）

4.6 应用程序编程接口（API）

本节将对 ASAM ODS 的第二个方面内容进行详细论述。但要实施此标准需要学习第 13 章的内容。

ASAM ODS 提供两种不同的接口：

- 程序接口（RPC-API），ASAM ODS 第一代接口，也是大多数在用 ASAM ODS 服务器和客户端使用的接口；

——面向对象接口 (OO-API)，提供最先进的数据存取和最便捷的数据操作方法。目前基于 OMG (对象管理架构) 对象模型和 CORBA (公共对象请求代理结构) 的平台实施以及其他技术在将来也会得到支持。

兼容 ASAM ODS 的服务器必须完全支持 OO-API 或者 RPC-API 的所有功能，或者两者都支持。服务器的产品说明中必须清晰申明使用的是哪种 API (具体到哪个版本)，否则厂商不得声称符合 ASAM ODS 规范。

ASAM ODS 的客户端可以使用任意两种接口之一的子集。不过，对功能多少没有具体要求，但所实施内容必须完全符合 API 规范要求。客户端的产品描述必须清晰明确申明采用的是哪种 API 接口 (哪种版本) (这也是与服务器交互提出的要求)；否则厂商不得声称符合 ASAM ODS 规范。

有时，ASAM ODS 会提供交叉测量的机会。几个客户端与服务器相连，根据事先拟定好的场景调查客户端行为。检测结果可以用于进一步的讨论，用于验证和进一步开发。服务器和客户端的实施者可以在 ASAM 注册参加这样的交叉测量。

ASAM 提供每种 API 接口的定义文件：OO-API 接口定义在文件 `ods.idl` 中，RPC-API 定义在 `aods.x` 中。

所以 OO-API 接口的模块及功能将在第十章中详细介绍 (包括用途，调用序列，返回值和示例代码)。所有 RPC-API 接口定义方法将在第 13 章详述。

两种 API 接口在功能上有所不同。RPC-API 的功能是 OO-API 功能的子集。下面列出了 RPC-API 接口中所没有的重要功能：

- 无法创建和改变应用模型 (只读访问)。AoSubmatix 和 AoLocalColumn 通过特殊方法处理 (在 OO-API 接口中，它们可以按照正常的应用元素来处理)；
- 不支持用于应用属性值序列传输的数据类型 (如 DS-LONG) (尽管支持测量数据内序列)；
- 实例中无迭代器 (查询返回返回结构内所有数据)；
- 服务器中无数值矩阵 (子矩阵必须通过客户端合并)；
- 不支持事务处理；
- 不支持锁定。

ASAM ODS API 接口将客户端应用和服务器应用区分开来。图 9 显示了其操作原理。

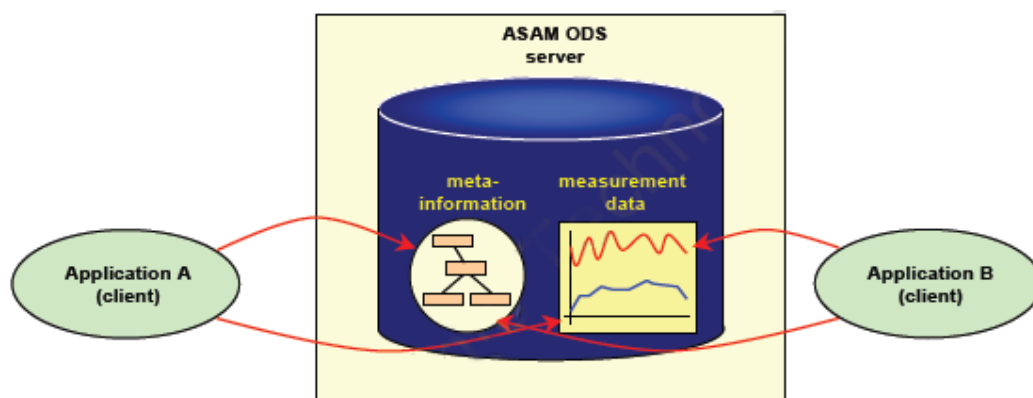


图9 访问 ASAM ODS 服务器的客户端应用

在 ASAM ODS 服务器中主要有两种信息类型：

- 数据本身（测量数据，测量设备的描述数据等）
- 关于数据内容和这些数据结构的信息（所谓的元信息），以应用模型为代表，而应用模型自身则依赖基础模型。

这两种信息都可以通过客户端应用访问。因而想从服务器中读取数据的客户端所使用的典型序列是：与服务器相连：

- 读取元信息；
- 存取数据本身。

当确定 ASAM ODS 使用 API 接口时，要考虑到以下通用要求和功能需求。

主要的通用需求有：

- API 应该隐藏客户端与服务器连接所用网络。
- API 接口需要考虑某些客户端可能要与服务器进行大量数据交换的同时，其它客户端也可能需要一个一个地存取数据。两种场景都不能让服务器或者网络过载；
- API 应该适用于所有大型平台和编程语言。这让客户端和服务端实施自行选择使用应用平台和操作环境，而不必考虑 API 规范要求；
- API 接口允许进行事务处理操作。这意味着根据使用 API 接口的动作序列要么所有动作成功，要么所有行动被客户端所取消（如同从未发起过）。这样可以保证数据的一致性。

主要的功能需求是：

- API 能够存取与基础模型、应用模型和实例相关的所有信息；
- 基础模型只用于提供信息，不能够通过接口修改。应用模型和实例必须支持读写操作；
- API 接口只关心 ASAM ODS 数据模型确定的规则是否被遵守；
- API 接口应当提供能够进行访问控制的工具。某些数据操作只对有特别授权的客户开放；
- API 接口应当能提供一个基本的搜索和选择功能，以减少网络流量。

下节内容将对 API 接口做一个简短介绍，粗略概述其内在原理和如何启动一个应用。如何正确使用 API 接口详见第 12 章和第 13 章。

4.6.1 面向对象 API 接口（OO-API）

ASAM ODS OO-API 事实上是一组按照 ASAM ODS 数据存储标准处理数据的接口。OO-API 接口确定时考虑了上述的通用需求和功能需求。

ASAM 工作组调查发现 CORBA（公共对象请求代理结构）是唯一非专用分布式面向对象传输机制，它允许定义符合上述要求 API 接口。CORBA 是 OMG（对象管理组织）设立的标准，对象管理组织是由一些大型 IT 公司设立和支持的机构。CORBA 的基本理念是提供一个在系统上传输透明的通信机制，关注在此系统上各应用之间的中介服务。一个内核程序，即所谓的 ORB（对象请求代理）在各系统上运行。每当提供接口（指的是提供展示具体对象及使用对象方法）系统上出现应用，它通知其 ORB 这个接口，并被视为服务器应用，在同一个或者其他系统上的应用这时就可以使用这个接口作为客户端来（通过询问 ORB 可用服务器，静态或动态）连接这个服务器。

结果，驻留不同平台/系统的交互对象可以在网络上透明地结果。对象实施与任何传输相关的代码无关，传输由 ORB 独自负责。

ORB 在几大供应商的不同平台上都可实现，还包括有些免授权费的产品。需要使用 CORBA（公共对象请求代理结构）2.3 或更高版本，才能保证不同供应商提供的 ORB 能够互相操作。

关于更多的 CORBA 及 OMG 信息可以登录网站（www.omg.org , wwwv.corba.org）查看相关文件和文献。

OO-API 接口规范使用 CORBA IDL（接口定义语言）语言。CORBA IDL 语言和通用的程序语言（如 C,C++,JAVA）之间有映射关系，IDL 编译器可以将接口说明规范转换为，比如 C++头文件或者 JAVA 文件，这些文件能够直接用于生成客户桩和服务端架构代码。

现有接口具有向上兼容性，如果未来发生改变则先有接口定义也会相应扩展。

以下描述的是 OO-API 接口的主要方面。然而这不过是一个概略描述，不能涵盖所有接口，如果想进一步了解 API 的具体应用，强烈推荐学习详细描述接口的专门章节。

4.6.1.1 入口点

根据现代系统结构最新描述，API 提供一个厂商接口。

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

每当网络中存在可用的 ASAM ODS 服务器，并且在 ORB 注册后，服务器会暴露这个厂商接口，允许用户请求其一般信息，并发起与服务器的对话（通过使用用户名和密码的认证字符串）。

会话自身也暴露一个接口。

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

会话的主要目的是：

- 让客户端存取 ASAM ODS 服务器中的基础模型，应用模型和实例。（这包括引用这些接口，允许定义和执行查询，进一步缩小感兴趣实例集合）。
- 根据与 ASAM ODS 服务器行为相关的一些可选项（所谓语境变量）和实例操作中的缺省设置。
- 管理事务处理，允许客户端调用接口操作序列，这些接口操作要么成功，要么由服务器全部取消。

通过传回与之关联的信息或者是完全传回数值，会话接口能够通过应用结构接口（见下文）连接应用模型。这最终体现为服务器和网络负荷的问题，这将由客户端来决定在会话期间究竟需要的是部分应用结构还是全部信息。

4.6.1.2 存取基础模型

通过以下接口存取基础模型：

BaseStructure{...}	基本结构
BaseElement{...}	基础元素
BaseAttribute{...}	基本属性
BaseRelation{...}	基本关系

会话时，可以请求引用基础模型。

基本结构接口允许访问所有的基础元素和所有的基本关系（两个基础元素之间）；客户端可以在需要时确定几个选择标准（模式）来浏览基础模型。

ASAM ODS 基础模型的版本号码也可以从此接口中获取。

基础元素知道（而且也可以要求提供）它所有的基本属性、基本关系和与之相关的其它基础元素。

基本属性知道（而且也可以要求提供）它的数据类型、名称，是必选属性还是其它属性。它只能通过它所依附的基础元素来访问。

基本关系可以被看做是一个带有专门接口的特殊基本属性。它描述的是两个基础元素之间的直接联系，保存（其他之外）第一个和第二个基础元素（起点和终点）的引用，关系类型（如 **FATHER**，**CHILD**，**INFO** 等），还有关系的范围（如[0,1]，[1,MANY]。它还知道逆反关系（元素相同但方向相反），如果逆反关系存在的话。

这些基础模型接口提供专门对象的所有信息，但不允许修改。

4.6.1.3 存取应用模型

存取应用模型通过接口提供：

ApplicationStructure{...}	应用结构
ApplicationElement{...}	应用元素
ApplicationAttribute{...}	应用属性
ApplicationRelation{...}	应用关系

会话时，可以请求引用应用模型。

应用结构接口允许访问所有的应用元素和所有的应用关系（两个应用元素之间）；客户端可以在需要时确定几个选择标准（模式）来浏览应用模型。

此外应用结构接口允许增加和删除应用元素和应用关系。

应用结构接口允许访问实例，而且允许确定 ASAM ODS 服务器使用的选择标准。

最后，客户端可以请求应用模型检测自身的一致性。

应用元素允许一系列动作，包括：

- 浏览应用结构，找到与之相关的应用元素、应用关系、实例和相对应的基础元素等等；
- 创建和删除该应用元素的应用属性和实例；
- 请求和确定访问应用元素及实例的权限。

应用属性知道（也可以要求提供）它的数据类型、名称、是否为必选属性或其他属性。它将与具体单位相联，确定访问权限，只允许访问限定的应用属性。与基本属性相比，所有这些信息可以由客户端在最初设定或者在今后进行改变。

应用关系（与基本关系相似）可以视为有专门接口的特殊应用属性，保存与上述基本关系同类的信息。它可以确定两个应用元素之间的直接联系。然而，客户端不仅可以请求访问应用关系的信息，而且可以改变这些信息。

4.6.1.4 存取一般实例

应用元素和应用模型自身可用于引用实例。如果出现不只一个实例被引用，系统提供一个迭代接口方便在一系列实例中进行导航。

实例自身提供的接口是：

InstanceElement{...}	实例元素

要操作一组实例时，可以采用下面的接口：

InstanceElementIterator{...}	实例元素迭代器

可以要求实例提供名称、ID、与其它实例的关系以及其它特征。

可以设置与其他实例之间的关系。实例可以增加或者删除属性，而且属性可以设置为固定的数值（含单位）。

而且，有处理实例存取权限的方法。

4.6.1.5 存取结果数据

结果数据一般是海量数据，因此处理方式不同。它们以实例形式存储在 ASAM ODS 中，但这些实例行为特别，因而通过以下特殊接口存取：

Measurement{...}	测量
SubMatrix{...}	子矩阵
ValueMatrix{...}	数值矩阵
Column{...}	列

测量和子矩阵接口显示实例行为（派生自实例接口）但允许执行其它动作。

为了使用这五个接口，理解结果数据的组织结构是很重要的：

测量结果是在后续测量点中测量数量的数值。然而并非所有测量数量都有必要在同一测量点上取值，一些可以在更高比率和密度的测量点上进行取样；一些取样只需在整个测量期间的某个部分进行。在一个大矩阵中存储测量信息会使矩阵的大部分空间空白，这就造成了存储空间浪费。

因此，ASAM ODS 允许将一次测量存储到一组矩阵中（称为子矩阵）。每一个子矩阵存储的量的数据是按照同种方式取样，因而不会导致不相关（空白）的矩阵位置。

存取结果数据有多种方式：

- 全部的测量数据可以通过 Measurement 接口获取。在此种情况中，返回 ValueMatrix，可以要求 ValueMatrix 提供一个或多个结果数据（包括数据单位）。可以要求提供同一测量点（行）全部或部分数据，或者同一测量的量（列）的全部或部分数据；
- 可以添加、修改、和删除数据。包括增加或删除矩阵中的整个行或者列；
- 子矩阵中的数据可以通过 SubMatrix 接口存取。同上，返回 ValueMatrix，可以要求 ValueMatrix 提供一个或多个结果数据（包括数据单位）。但是，这个数值矩阵只提供来自子矩阵的数据；
- 单个子矩阵如何组合才能获得（部分空白）的完整的测量矩阵，可以通过 SMatLink 接口进行设置和提出访问请求；
- 通过 Column 接口可以获得子矩阵列的含义以及列的其他信息。

4.6.1.6 查询

ASAM ODS API 接口允许客户端应用设置一组条件缩小向服务器请求查询信息的范围。只有与条件相符合的实例才会被服务器返回。这种处理海量数据的方法称为“查询”。

通过比较操作符 “>”，“<”，“=”（包括枚举型属性的操作如“包含在集合中”和“不包含在集合

中”)可以设定查询条件,

还有逻辑操作符 AND,OR 和 NOT (可能使用括号)。

服务器以同步方式执行查询,用户需必须等待服务器执行完查询任务。

为了处理查询,ASAM ODS 服务器暴露下面的接口:

APPIElemAccess{...}	应用元素存取

4.6.2 程序接口 (RPC-API)

与 OO-API 接口相比, RPC-API 接口只有几种客户端可以调用的方法,而且要通过使用 ONC-RPC (开放的网络计算远程过程调用) 机制提交给服务器。当调用这些方法时,相当复杂的结构一般转交给服务器。服务器自身也是通过结构来返回服务结果。

客户端联系和使用服务器必须遵守的主要顺序和 OO-API 接口类似:

- 客户端首先必须通过 AOP_OpenEnv (..) 建立会话;
- 客户端随后使用 AOP_GetApplinf (..) 请求应用模型。服务器返回完整的应用模型,包括所有的应用元素和应用关系列表。客户端可以通过 AOP_GetAtt (..) 获得每个应用元素的所有应用属性;
- 客户端可通过 AOP_GetInstRef (..) 获取某个应用元素所有实例的列表;
- 客户端可以通过 AOP_GetVal (..) ,AOP_GetInstAttr (..) 和 AOP_GetValE (..) 获取应用和实例属性的数值。建立了筛选机制,确保只有符合用户特定标准的数值可以传输;
- 客户端可以通过 AOP_PutVal (..) 删除或者增加实例,或者修改属性数值;
- 从服务器中读取或写入服务器的测量数据是采用了操作子矩阵的特殊方法,存取这些数据需通过 AOP_GetValAttr (..)、AOP_GetValInf (..)、AOP_GetValVal (..)、AOP_PutValVal (..) 接口进行;
- 最后,客户端必须通过 AOP_CloseEnv (..) 结束会话。

有更多方法设置和检索安全信息 (因而可以限制访问权限只授予特定用户组)、管理个人设置,以及其它目的。

尽管与早先版本相比,接口定义文件有些细微的改变,但从 PRC-API 接口 3.2 版以来所有方法及它们的参数都没有改变,而且将来也没有改变的计划。

除以下限制外,其互操作性是有保证的:

- 老客户端仍然可以访问采用最新接口定义编译的服务器,尽管它们许不能访问最新的基础元素,因为它们可能不清楚对应的基本标识符;
- 新客户端可以访问新旧两个版本服务器。但仍然会出现访问旧服务器不能存取最新基础元素的情况,因为旧服务器可能不清楚新的基本标识符。

两种限制都只需对旧客户端或者服务器做出细微改进即可解决。

4.6.3 TODS API 接口 (TODS-API)

TODS API 是继承于 ODS API 定义的数据存储标准处理数据的接口。接口的定义格式为 web service

标准格式，这样就可以使用 web 调用访问使用 TODS API。

4.6.1.1 入口点

做为 TODS 访问入口，API 提供一个接口。

IMapleTRFactory{...}

访问这个接口可以得到 TODS 全局的信息。

会话自身也暴露一个接口。

IMapleTRSession{...}

会话的主要目的是：

- 关闭会话退出登录。
- 获取数据模型访问。
- 获取 DB 数据的访问。
- 获取文件的访问。

4.6.1.2 存取基础模型

通过以下接口存取基础模型：

GetBaseModelNameList {...}	获取所有基础模型清单
GetBaseModels {...}	获取所有基础模型表信息
GetModelByName {...}	根据模型表名称获取基础模型信息
GetModelRelationsByName {...}	根据模型表名称获取关联模型信息

这些基础模型接口提供专门对象的所有信息，但不允许修改。

4.6.1.3 存取应用模型

通过以下接口存取应用模型：

GetAppModels {...}	获取所有应用模型表信息
AddAppModel {...}	新建应用模型表信息
EditAppModel {...}	编辑模型表信息
DropAppModelByName {...}	根据模型表名称删除模型表信息

4.6.1.4 存取一般实例

通过以下接口存取实例及实例数据：

GetRecord{...}	获取模型实例数据
GetRecordSet {...}	获取模型实例数据集合
GetRelationnalRDSet {...}	获取指定模型的关联模型数据
AddRecord {...}	新建模型实例数据
EditRecord {...}	编辑模型实例数据
DelRecord{...}	删除模型实例数据和其包含的实例数据及文件

4.7 物理数据存储

本节将详细介绍 ASAM ODS 规范的第三个方面。但是要实施本标准则必须学习本标准的第 6 章。

目前，对使用关系数据库和混合模式存储的物理存储格式有说明：尽管根据 ASAM ODS 基本物理数据存储规范，所有数据都可以存储自数据库表中，混合存储模式允许将海量数据存储到单独的文件中。

使用关系数据库存储信息要求明确数据库中使用哪张表，需要哪些列，哪个是表字段必须唯一（因为他们是主键）等等。ASAM ODS 信息已经说明完毕，在本节里也会简要介绍。

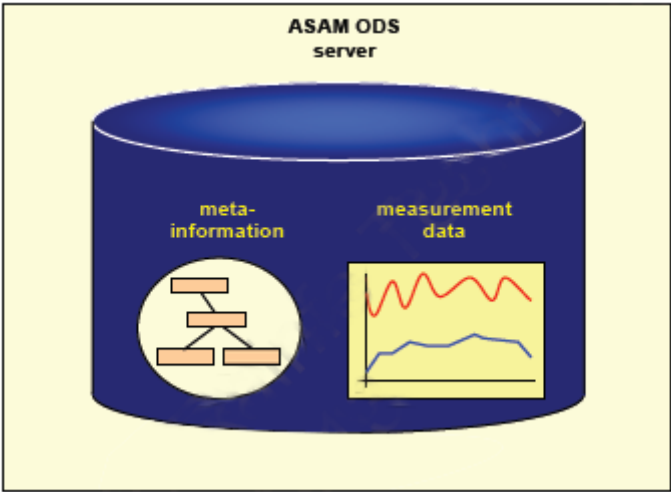


图10 ASAM ODS 服务器的物理数据存储

首先，如图 10 所示，要对元信息存储和实际值存储进行区分（实体值指的是：所有测量数据，以及实例的属性值及之间关系。）

基础模型上的信息并非完全存储于数据库中，相反，它们位于 ODS 服务器的其它位置，（例如源代码中）。

应用模型中的信息主要存储在下面三张表格中。

（1）应用元素表（即 SVCENT）

表中列明所有应用元素，每个应用元素允许保存以下信息：

- 应用元素的标识符（由 ASAM ODS 服务器提供，在表中必须唯一）；
- 应用元素的名称（由用户或者应用模型设计者命名）；
- 应用元素所属的基础元素的标识符（BE-ID）（在确定基础模型时，由 ASAM ODS 唯一指定）；
- 应用元素各实例存储所用表的名称。

图 11 显示了此表的结构，和一些示例数据：

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

图11 SVCENT： 存储应用元素描述

(2) 应用属性表（即 SVCATTR）

此表列明所有应用元素对应的应用属性，包括所有属于[0: n]和[n: n]关系的应用关系。

图 12 显示了此表的结构，和一些示例数据：

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

图12 SVCATTR: 存储应用属性描述

其机构允许存储以下类似信息：

- 应用属性或者应用关系的名称（AA/AR-名称），这个名称由应用模型设计者提供，在同一应用元素的应用属性和关系中必须唯一；
- 相应应用元素的标识符（AE-ID）：该标识符与 SVCENT 表标示符相同；
- （如果应用属性代表基本属性）相应基本属性名称（BA-名称）（确定基础模型时，由 ASAM ODS 定义的名称）；
- （如果输入数据描述应用关系）应用关系指向（AET-ID，目标）应用元素的标识符；
- 实例表存储应用属性或应用关系值的列名；
- 应用属性或是应用关系存储实例表使用数据类型的标识符（DT-ID）。

(3) 应用关系表（即 SVCREF）

该表包含应用元素之间所有[n,m]型关系[0,n]、[1,n]型关系的处理方法类似于属性，存储在上述的 SVCATTR 表中。SVCREF 表中的内容主要是：

- 两个具有[n, m]型关系应用元素其中之一的标识符（AE1-ID）；
- 两个具有[n, m]型关系应用元素另外一个的标识符（AE2-ID）；
- 每个单一关系实例存储所用表名。

除了以上三种表格（或多或少为静态表）外，还有一组、存储真实值（实例）的表格：

(1) 保存实例及其属性的表

每一应用元素有专门的表格，保存其实例。表名在 SVCENT 表中给出。每列包含一个属性/关系，应用属性/关系与其对应列之间的关系由列名称给出，在 SVCATTR 表中确定。每行包含一个应用元素实例。

图 13 是这种表格的简单示例（气缸实例：其 ID 由服务器提供）。

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

图13 气缸实例表：存储气缸实例属性值

(2) 存储[n, m]型实例关系的表

在应用模型中，每一[n, m]型应用关系对应一个表。每两列只保存一个应用元素实例。每行说明哪两个实例相关，同一实例在表格中出现往往不止一次。

(3) 保存实例属性的表

属性/关系和添加到实例上，因此，不存在于相应的应用元素之中，因而也不能保存到 A 类表中。因而专门设置了存储全部实例属性/关系的表格 SCVCINST。每行是某一属性或者关系，列存储的信息有：名称，单位标识符，值，所属实例的标识符。

(4) 保存测量结果（海量数据）的表

数据库中有专门存储测量数据的表格。测量数据由一些子矩阵组成，它们共同组成了完整（稀疏）测量矩阵。子矩阵的列是某一测量量的数据，子矩阵的行是所有测量量的数据，这些数据都取自同一个测量点（一般是时间点）。

(5) 保存安全信息的表

要限制或者允许对 ASAM ODS 的数据存取，物理存储设置了三张表 SVCALI、SVCACLA、SVCTPLI。SVCALI 存储实例保护的安全数据，SVCACLA 存储保护属性和元素的安全数据。SVCTPLI 存储安全的 ACL 模板。

4.8 传输格式

本节将详细介绍 ASAM ODS 规范的第四个方面。但是实施该标准则必须学习 ASAM ODS 第 10 章。ATF 是描述以文本文件形式传输 ASAM ODS 信息的语法规范。它通常用于：

- 将某个 ODS 数据库中的全部或部分信息传输给另一数据库；
- 当系统不能通过 ODS API 接口连接服务器或者暂时不能通过网络连接到服务器时，将信息从系统中导入 ODS 数据库。

ATF 可以传输完整的应用模型和带有属性和关系的实例。因为基础模型已经标准化，所以 ATF 当中不必包括基础模型。然而，要处理随着时间推移基础模型标准出现的变化，应该提前准备好，在 ATF 当中包括设定的基础模型标准版本。

ATF 规范最经典的变体就是普通的 ASCII 文本格式（字符 3210..25510）。因而它能够在任何应用平台（操作系统）上传输。ATF 的另一变体 XML（可扩展标识语言）基于 UTF8 字符集。

ATF 规范允许将整个信息分成一组独立的文件，每个文件（除第一个外）都被另一个所包含。这样处理有几点好处：

- 如果文件大小受到某个传输媒介的限制，传输可以采用几个连续步骤进行；

- 如果传输信道嘈杂，则几小文件相对一个大文件更易于传输；
- 如果整个信息中有些部分的信息是不变的（如应用模型或者某些描述数据），只有信息的变化部分需要额外磁盘空间，其不变部分能够在需要时随时调取。

ATF 规范允许单独存储海量数据，这些数据一般出现在本地列信息（真实测量值）必须传输时。由于 ASCII 格式传输数据所需空间是二进制格式传输数据的几倍，ATF 允许采用二进制格式传输海量数据。为此，相应的 ATF ASCII 文件包含这些二进制文件结构的说明，以及标准数据类型字节序列的信息（即“首先写入文件流的是高字节还是低字节”）。

在本标准中存在两种不同的 ATF 规范：经典的 ATF/CLA 规范和新生的 ATF/XML 规范。ATF/CLA 规范是大家认可的数据交换标准，至今仍在使用，ATF/XML 规范很有希望从日益盛行的 XML 技术中获益。

尽管预计 ATF/CLA 不会进一步修订，但未来 ASAM ODS 仍然会继续支持 ATF/CLA 规范。

4.8.1 ATF/CLA

经典 ATF 规范的语法会在第五章采用语法图表进行详尽描述。

ATF 规范的微缩结构见图 14 所示。

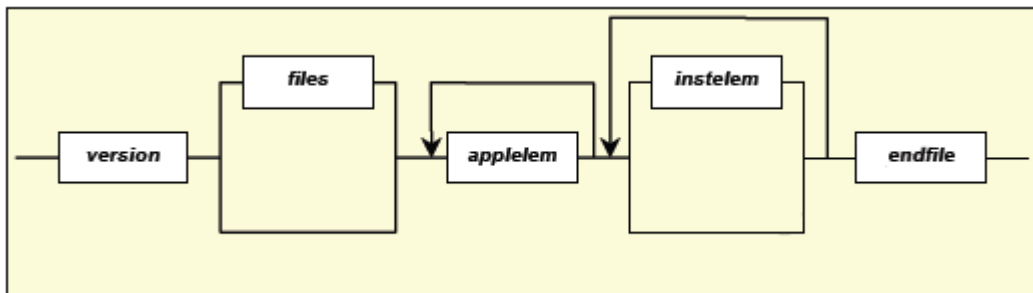


图14 ATF/CLA 文件结构

如图 15 所示，ATF/CLA 文件逻辑上分为五个主要部分：

- 版本（version）：提供关于 ATF/CLA 版本的信息；
- 文件（files，可选）：提供一组物理文件列表，它们共同构成完整的 ATF/CLA 逻辑文件；
- 应用元素（applelem）：是对应用元素的描述，并会为即将由 ATF/CLA 文件传输的应用元素重复出现；
- 实例元素（instelem，可选）：是对应用元素实例的描述；它在 ATF/CLA 文件传输每个实例中只出现一次。如果 ATF/CLA 文件只传输元信息，实例元素的数字为零；
- 结束文件（endfile）：文件中提供再没有 ATF 相关信息的信息。

这些语法元素在 ATF/CLA 章节中有详尽描述，详细到说明 ASCII 字符集中每个字符的序列程度。必须重视语义关系。ATF/CLA 规范是一种语法规则，并不关系语义关系。但是操作 ATF/CLA 文件的应用必须重视语义关系。

例如：根据 ATF/CLA 语法定义很有可能生成一个从语法看完全正确的 ATF/CLA 文件：

- 实例与一个根本不存在的元素相关，或者；
- 应用元素不包含必选的基本属性，或者；
- 应用元素或实例与其它任何元素没有关系，或者与 ATF/CLA 文件中其它任何元素没有关系。

这些例子在语法上都行得通，但是一旦将其导入 ODS 服务器中将会立即出现问题。
因而，如果不懂得 ASAM ODS 数据模型，就无法生成恰当的 ATF/CLA 文件。

4.8.2 ATF/XML

过去几年来，随着基于文本的 XML 数据交换标准的提出，ASAM ODS 工作组也确定了基于 XML 的 ATF 标准。这种交换格式将会在第 10 章进行详尽描述。

为了便于升级，ATF/XML 定义的关键词类似于 ATF/CLA。

引入 XML 文件进行数据交换和数据存档是基于以下几个目的：

- XML 工具对 ATF/XML 文件自动验证比对 ATF/CLA 文件更为细致。XML-Schema 对 XML 文件内容采用 XML 语言进行描述，这种描述可以由全球通用的自动化工具进行处理。
- XML 相对易于使用，且查看和编辑 XML 的工具比较普及。
- XML 应用广泛且众所周知，XML 将是一种可以长期使用的方法，能够保证 ASAM ODS 文件稳定性。

与经典的 ATF/CLA 类似，ATF/XML 同样在同一数据文件中提供模型数据（分别与数据相连）。这就保证了模型（元数据）与数据之间的一致性，从而确保在数据交换和归档时，数据可以被正确理解。其它特点：

- 文件头部分允许出现版本，基础模型和来源系统的信息；
- XML 语法能够更加清楚地传输引用信息，例如使用指针，这能够使得引用信息易于被自动工具验证；
- 元信息附在标签上，这使得文件结构更加完善，更易于检查；
- ATF/XML 文件是逻辑文件，由一个或多个物理文件组成。

因此，ATF/XML 文件的整体结构可用如图 15 表示。

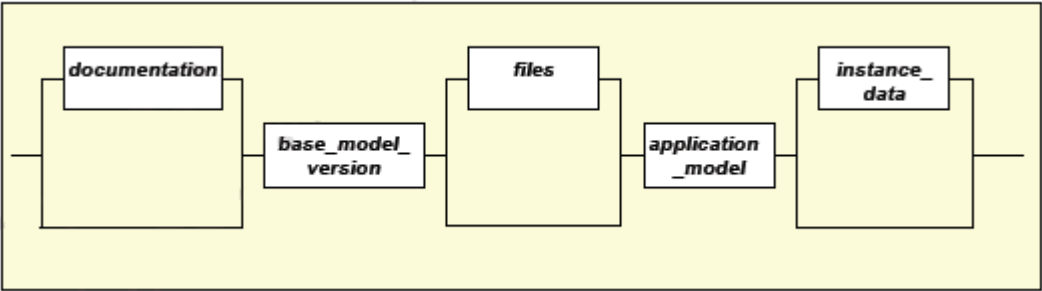


图15 ATF/XML 文件结构

4.9 应用模型

本节将详细介绍 ASAM ODS 规范的第五个方面。但是要实施本标准则必须学习本标准相关章节。

不论何时，基于同一应用模型的不同产品之间数据交换都非常容易。共同的应用模型确保了产品能够采用同一种方式，以相同的数据格式来存储同样的信息，因而这一信息的含义大家都了解。

5 体系结构

5.1 简介

ODS 标准的内容由 4 部分构成，每部分相互依赖：

- 数据模型（由基础模型和应用模型派生规则组成）；
- 物理存储；
- 应用编程接口（APIs）；
- ASAM 传输格式（ATF）。

数据模型是基础，所有其他部分与数据模型相统一。图 16 图示了数据模型、应用、应用接口、物理存储、传输格式及应用模型之间的关系。

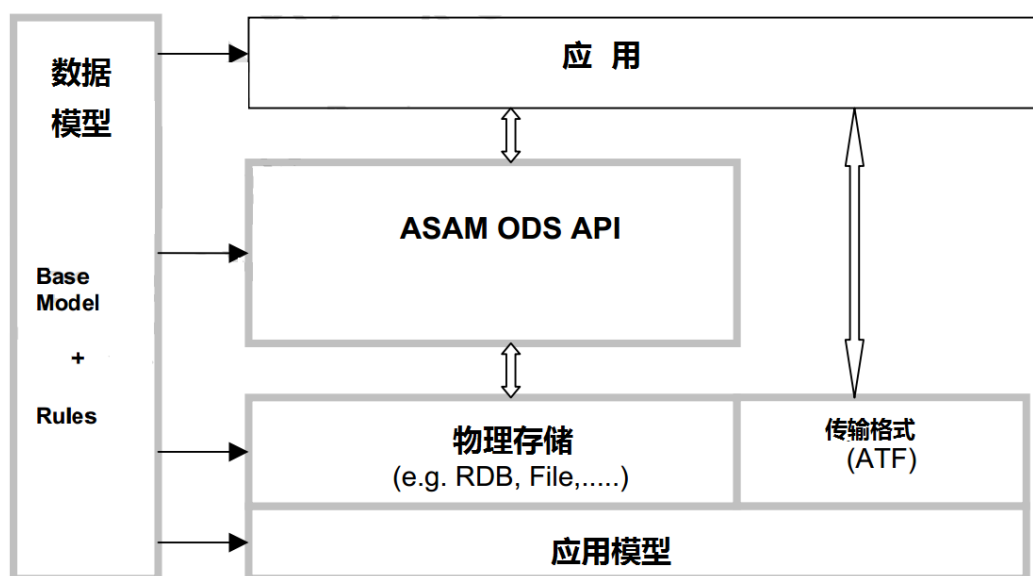


图16 ASAM ODS 规范的组成

所有组件依赖于基础模型和规则（通用数据模型）。

本概念可以让客户根据基础模型建立与客户相关的具体的应用模型。基础模型与应用模型之间的映射关系就是元信息。元信息与最终数据（实例信息）一起被存储。

例如，关系型数据库的物理存储指定用某种方式，这样，元信息就可以存储在固定的指定的数据表中。另外，还可以指定传输格式（ATF）的元信息的存储，定义访问层的相应方法。

由于数据模型是所有规范的基础，本模型的元素在本章简单介绍，在第 7 章作详细描述。数据模型的规范包括下列定义：

- 基础模型；
- 派生项目-具体应用模型的规则；
- 构建应用模型实例的规则。

他们之间的关系是：基础模型派生（→）应用模型派生（→）实例。

TODS 标准的各部分组成与关系，与 ODS 类似，其中重点强调了，TODS 标准数据模型、TODS 应用模型、TODS APIs、以及 ATF/XML 以及它们的关系，如图 17。

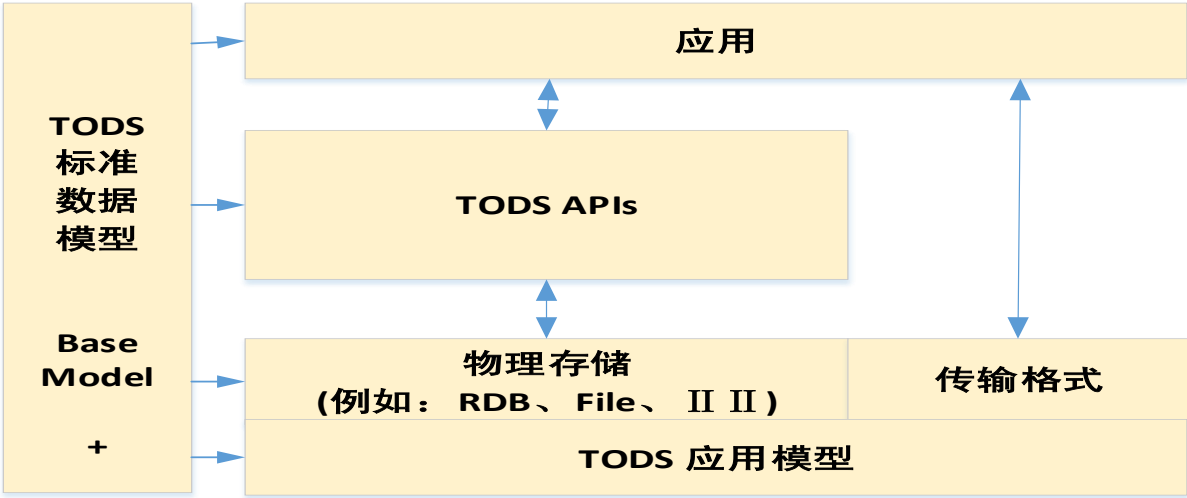


图17 TODS 规范的组成

5.2 基础模型的元素

5.2.1 概述

图 18 是 ASAM ODS 基础模型的示意图。

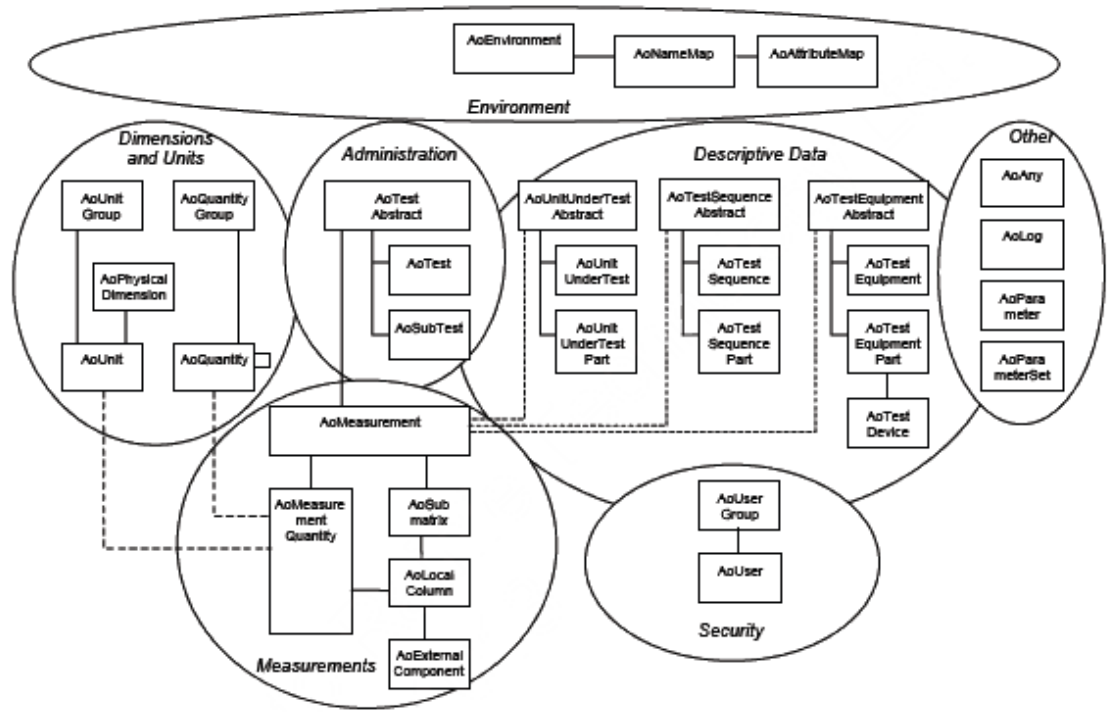


图18 ASAM ODS 基础模型

基础模型包括以下基础元素,从这些基础元素中可以派生出应用元素。这些元素根据用法进行分组。第7章有每个基础元素的详细描述。

5.2.2 环境

采用下列元素描述 ASAM ODS 服务器及数据存储的环境:

- AoEnvironment (环境);
- AoNameMap (名称映射);
- AoAttributeMap (属性映射)。

5.2.3 维度和单位

采用下列元素构建单位目录,(通过 *AoPhysicalDimension*) 定义与国际单位制(SI)单位的关系:

- AoUnit (单位);
- AoQuantity (量及其属性);
- AoPhysicalDimension (物理维度);
- AoUnitGroup (单位组);
- AoQuantityGroup (量组)。

5.2.4 数据管理

采用下列元素构建测量数据及采集数据的树状结构化管理,支持多层次的子试验:

- AoTest (AoTestAbstract 的子类型);
- AoSubtest (AoTestAbstract 的子类型)。

5.2.5 测量

采用下列元素构造测量(及评价)数据结果的结构:

- AoMeasurement (包括测量量和 AoLocalColumn 数组的测量);
- AoMeasurementQuantity (测量中的测量量);
- AoSubmatrix (AoLocalColumn 数组);
- AoLocalColumn (测量向量);
- AoExternalComponent (引用外部文件的测量向量)。

5.2.6 说明性数据

采用下列元素构建试验补充说明的结构:

对于试验对象:

- AoUnitUnderTest (AoUnitUnderTestAbstract 的子类型);
- AoUnitUnderTestPart (AoUnitUnderTestAbstract 的子类型);

对于试验序列:

- AoTestSequence (AoTestSequenceAbstract 的子类型);
- AoTestSequencePart (AoTestSequenceAbstract 的子类型)。

对于试验设备:

- AoTestEquipment (AoTestEquipmentAbstract 的子类型);
- AoTestEquipmentPart (AoTestEquipmentAbstract 的子类型);
- AoTestDevice (AoTestEquipmentPart 的子类型)。

5.2.7 安全

针对安全数据（允许限制存取 ASAM ODS 服务器的内容）存储，提供下列元素：

- AoUser（用户）;
- AoUserGroup（用户组）。

5.2.8 其他数据

针对其他数据的存储，提供下列元素：

- AoAny（任意）;
- AoLog（日志）;
- AoParameter（参数）;
- AoParameterSet（参数集）。

5.3 应用模型规则

从基础模型中派生应用模型，需要遵循几个规则，包括从基础元素中派生应用元素的规则，对派生元素命名的规则和添加附加属性的规则。

5.3.1 应用元素命名规则

基础模型 → 应用模型

注：符号“→”表示派生。

例如：AoMeasurement → 耐久性测试

派生的应用元素可根据项目的特定要求命名。在元数据信息中要维护和保存派生应用元素与基础模型间的关系。应用元素的名称不能以“Ao”开头（“Ao”是保留给基础元素命名用的）。

5.3.2 构建应用模型

从基础模型中派生一个或多个的应用元素有不同的规则。附加的规定将控制哪些关系是被许可或被规定的注意以下3组基础元素采用不同的派生规则：

- 单个派生；
- 实例级的树状结构；
- 应用级的树状结构。

（1）单个派生

从下列基础元素中只可能派生出一个应用元素。在基础模型内，这些元素是可以最详细描述的基础元素（通过基本属性）：

- AoAttributeMap;
- AoEnvironment;
- AoExternalComponent;

- AoLocalColumn;
- AoMeasurement;
- AoMeasurementQuantity;
- AoNameMap;
- AoPhysicalDimension;
- AoQuantity;
- AoQuantityGroup;
- AoSubmatrix;
- AoTest;
- AoUnit;
- AoUnitGroup;
- AoUser;
- AoUserGroup。

(2) 实例级的树状结构

实例级树状结构允许使用下列基础元素。示例见图 19。

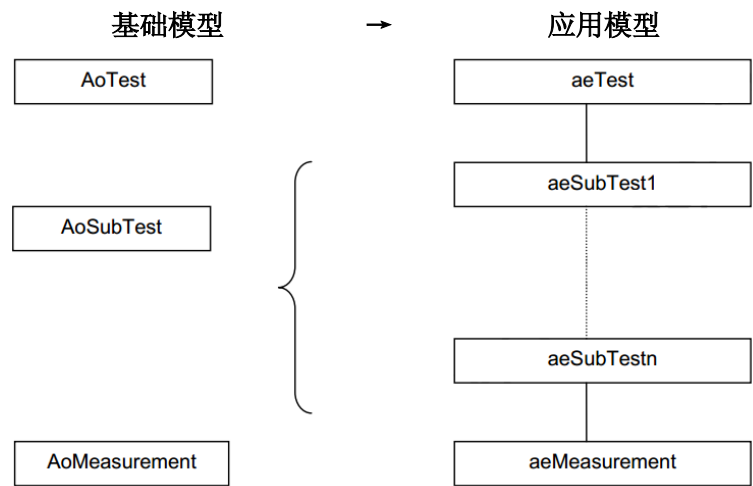


图19 实例水平的树状结构（示例）

从 **AoTest** 中正好派生出一个应用元素，从 **AoSubtest** 中，每个层次级别（1…n）正好派生一个应用元素。实例级树状结构是由一个应用元素的多个实例构造而成。作为特殊情况，允许不用 **AoSubtest** 类型的应用元素。从 **AoMeasurement** 派生出来的应用元素只附加到“Test”族的最低层元素上，对实例级树状结构的测量数据（**AoMeasurement**）进行管理。

(3) 应用级树状结构

使用下列基础元素，甚至可以构建应用级树状结构。对于试验设备示例见图 21。

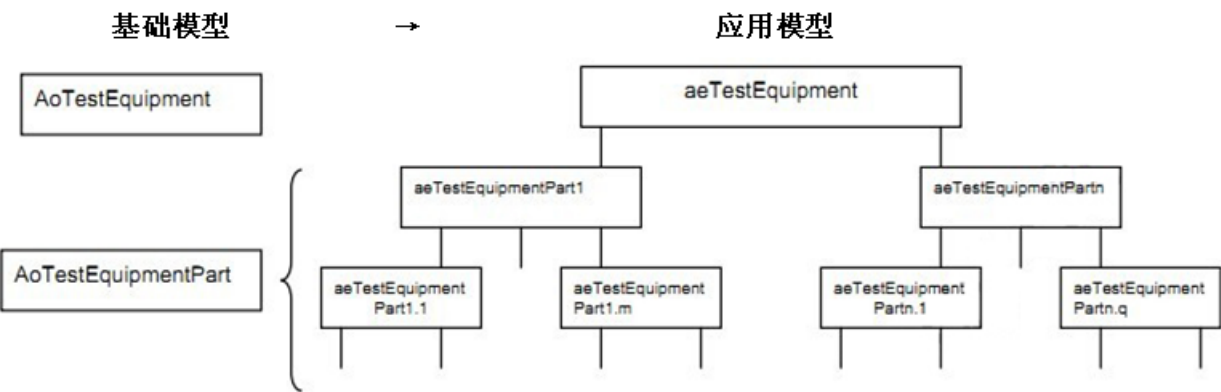


图20 针对测试设备的应用树状结构（示例）

这也可以应用到 AoTestDevice（针对测试装置的层次安排）。
对于试验序列，基础模型派生应用模型示例见图 22。

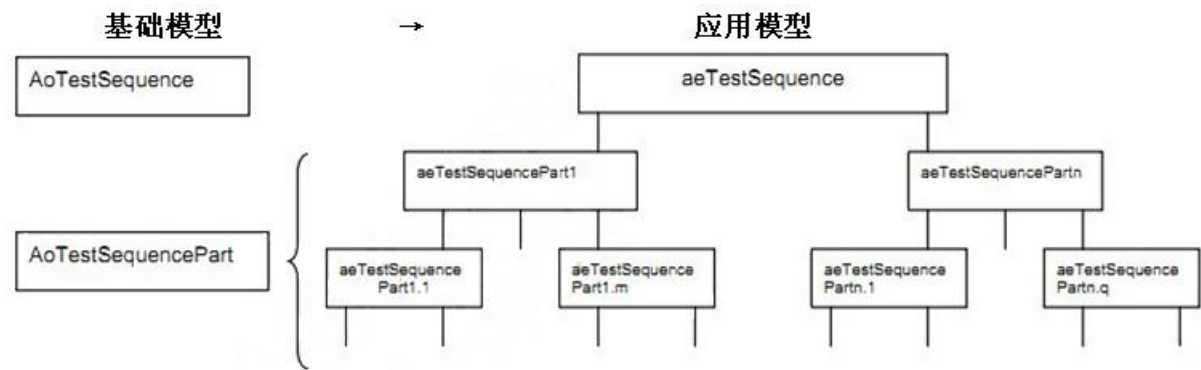


图21 针对测试序列的应用水平树状结构（示例）

对于被测设备示例见图 21。

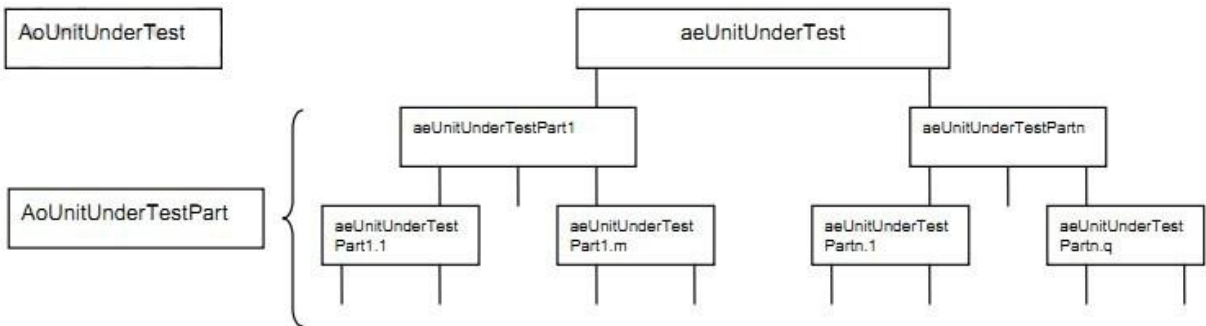


图22 针对受测设备的应用级树状结构

在这些情况下，可通过从 **AoTestEquipmentPart**、**AoTestDevice**、**AoTestSequencePart** 及 **AoUnitUnderTestPart** 派生多个应用元素构建实例级中已存在的树状结构。此外，从 **AoTestEquipment**、

AoTestSequence 及 AoUnitUnderTest 中可以尽可能多地派生出所需的应用元素。通过此方式，有可能定义尽可能多的实例级树状结构。

— 对于从 AoAny、AoLog、AoParameter 和 AoParameterSet 派生的应用元素

基础元素 AoAny 和 AoLog 可能经常需要被派生。与应用关系关联，可构建任意应用模型（也包括树状结构）。

基础元素 AoParameter 和 AoParameterSet 可能经常需要被派生。

5.3.3 映射基础模型和应用模型

为了在同一个 ODS 数据存储中支持多语种，所有应用元素、应用属性及关系的名称可以有任意数量的别名。

那些替代名称保存在两个实体 AoNameMap 和 AoAttributeMap 中。此外，实体 AoEnvironment 用于支持多种语言。

(1) **AoNameMap**: 在基础元素 AoNameMap 派生的应用元素每个实例中，为应用模型（应用元素）的实体给出了应用元素替代名称的列表，因此，可为应用元素构建别名列表。另外，这些实例中的每个实例关联了一个“属性映射”集合，包含了该应用元素的应用属性和关系别名。

(2) **AoAttributeMap**: 基础元素 AoAttributeMap 派生的每个应用元素实例包含一个应用元素的一个应用属性或关系的可替代名称列表。因此，有可能构建别名列表。

(3) **AoEnvironment**: 在该实体中，应该只有一个实例出现在任意一个数据集中。该实例被用来作为 AoNameMap 实例的阶层（这反过来作为 AoAttributeMap 实例层），并被用来存储一些全球声明：

——测试层级数；

——别名替代的意义，包括指定官方语言代码的选项。

5.3.4 存储测量数据

本标准的主要任务之一是提供管理测量数据（包括控制设备的定位点数据和测量前或后的计算数据）的方法。为此，ASAM ODS 数据模型引进了几个特定的基础元素如下：

——*AoMeasurement*;

——*AoMeasurementQuantity*;

——*AoLocalColumn*;

——*AoSubmatrix*。

这些基础元素相互关系及其与 ASAM ODS 其他基础元素的关系如图 23 所示：

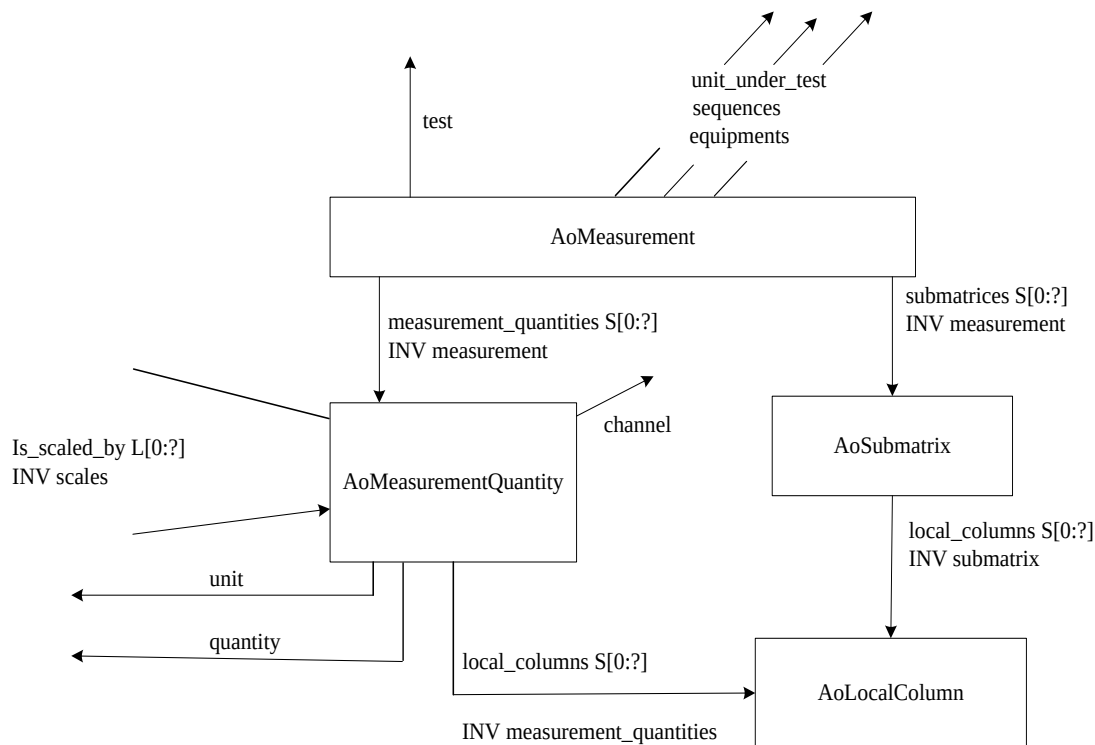


图23 为存储测量数据而设计的基础元素

除了这些，基础元素 *AoExternalComponent* 还提供一种用外部文件存储海量数据的方法，并可以在 ASAM ODS 模型内引用这些文件。

请注意基数说明中的区别：

- S[0: ?]=集合，元素从零到很多，集合里的元素不按次序排列；
- L[0: ?]=列表，元素从零到很多，按一定次序排列。

虽然第 7 章将详细说明这些基础元素，但本节描述了这些基础元素的全部特征及用法：

- AoMeasurement**：测量将所有描述测量环境和测量结果的信息关联在一起。
- AoMeasurementQuantity**：测量量是在测量里所使用的量。它引用一个通用物理量（**AoQuantity**），描述具体测量案例中物理量的特征。比如，测量‘长度’的单位通常是“米”，在特定测量中，测量所用单位可能是“英寸”。一次测量中可以使用多个测量量。在一次测量里，它们的名称必须是唯一的。不同测量量可引用同一个量。不同的测量要求不同的测量量集合（因为某些说明性数据有可能已经改变）；
- AoLocalColumn**：测量数据最终要保存在 **AoLocalColumn** 实例内。一个本地列只包含一个测量量的数据。其名称必须与相应测量量的名称相同。一次测量一般产生一个以上的测量量的数据。它们存储在不同的本地列中。此外，一个测量量的结果数据也有可能分布在不同的本地列中。
- AoSubmatrix**：子矩阵包含具有共同特征的所有本地列数据。这些数据必须包含相同量的

— 据样本，每个本地列的 n -th 样本必须在某种意义上彼此相关。

可能出现的几种情况如下：

——最常见的情况是，子矩阵所有的本地列的所有相应样本的每个样本在同一时刻(时间点)采集。

发生这种情况是因为，使用一系列传感器进行测量时，分析传感器取值的实时跟踪，同时对所有的取值进行采样。在这种情况下，除了传感器测量结果提供每个样本的时间戳之外，通常还会利用‘时间’类型的一个测量量；

——另一种情况是，子矩阵每个本地列的所有相应样本在同一位置(空间点)获取。这是因为，使用一系列传感器进行空间相关测量时，分析不同量的测量值的空间分配，这些空间分配彼此相关。在这种情况下，除了传感器测量结果之外，通常还利用唯一标识空间位置的一个测量量。

每个子矩阵与测量相关。因此，通过随后处理所有的相关子矩阵，可能从测量中访问所有的结果数据。有可能一个子矩阵只包含一个传感器的所有结果，需要提供大量子矩阵描述测量结果。也有可能所有的测量结果非常有规律，所以，只需要一个子矩阵来描述所有的测量结果。

一个子矩阵有可能只包含属于不同测量量的本地列。如果一个测量量在不同的本地列存储其结果取值，它们必须属于不同的子矩阵。

一个测量关联了所有可用的测量量以及所有含有测量结果的子矩阵。此外，它还关联所属的试验以及试验相关的被测单元、试验设备及试验序列。

一个测量量关联了所有存储其取值的本地列。此外，它还关联了其所属的测量及量，以及单位、负责产生取值的测试设备的项(通常是一个传感器或测量装置)、用于调整比例的其他测量量(最终结果值可以从不同样本的组合中计算得到)。

一个子矩阵关联了所有其包含的本地列。此外，还关联了它所属的测量。

因此，从测量到结果数据有两条路径：客户可以首先浏览一次测量的所有测量量；然后，选择一个指定测量量的本地列，访问其结果取值。另外，客户也可以首先浏览一次测量的所有子矩阵；然后，选择一个指定的子矩阵的本地列，浏览器结果取值。

为了在读取访问，方便客户应用实现，可以用 ASAM ODS 服务器产生一个完整的矩阵——“值矩阵”，包含一次测量所有的测量量的所有取值。值矩阵的所有列与测量量相对应，矩阵行包含个别结果取值。只有子矩阵之间的关系已知才可能将所有的子矩阵组合成一个整体矩阵。为此目的，每个子矩阵之内的本地列的每列可以被声明为子矩阵的‘独立列’。在很多情况下，这种独立列包含时间戳(相应的测量量因此有时间类型的单位)。在其他情况下，这种独立列只包含一个编号(“测量点”)或一个空间位置。

独立列所有候选列的共同特征是，它们可以被用于排序目的。由此产生的值矩阵行可以根据独立列排序。

只有那些子矩阵(其独立列具有同一相关测量量)可以组合成一个值矩阵。

一个值矩阵通常包含很多空字段。将测量结果存储在子矩阵有助于优化存储资源的利用。子矩阵是同质的。每个字段包含一个取值。如果有人物理(作为值矩阵)存储整个测量结果，将浪费大量存储空间。

此外，在一次测量期间，这种分离通常非常好地对应着物理条件：往往给定数据，作为子矩阵，因

为，在一次测量期间，所使用的测量装置通常用媒介物本地存储已经测量的数据，不一定与其他装置处于相同的时间分辨率。

下列示例（每个示例具有有限量的数据值）有助于进一步理解测量数据的存储概念。

示例1：子矩阵和值矩阵。子矩阵和值矩阵关系见图 24。

在该简单示例中，一次测量包含两个子矩阵，每个子矩阵有两个本地列。通过合并时间信息，子矩阵可能被组合成一个值矩阵。可以看到，两个子矩阵都有本地列‘时间’这一列，这列被声明为独立列。这个独立列被用来创建一个包含所有测量结果的整体值矩阵。该值矩阵包含 30% 以内的空字段。

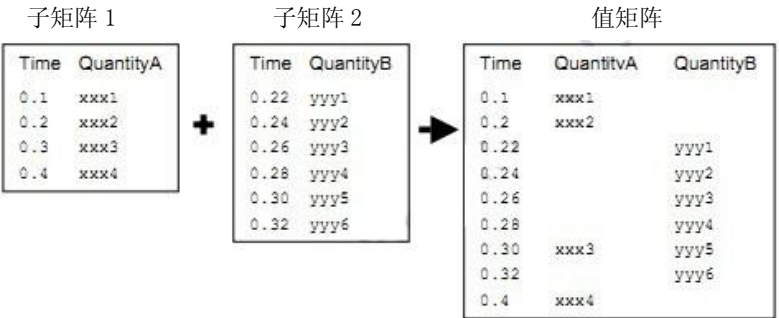


图24 子矩阵和值矩阵的关系

示例2：多个子矩阵和一个值矩阵。

在这个有点复杂的示例中，假设了下列情况：使用五个温度传感器 S1, S2, S3, S4, 和 S5 测量一台引擎不同位置的温度。应该存储采样取值。

- 对 S1 和 S2 在等距离的时间间隔（1 分钟）进行同步采样；
- S3 的以一半的采样速度，但与 S1 和 S2 同步；
- S4 与 S1 和 S2 同步采样，速度也相同。但只有在 S2 的取值低于 16.0℃（事件驱动采样）；
- S5 与其他传感器异步采样，但是，速度为 1 样本/分钟（如果由一个单独的装置取样，通常是这样）。

每个样本标有时间戳。所有的结果应该做进一步分析，并被可作一个时间函数。

虽然有几种可能将这些结果存储在 ASAM ODS，但是，以下可能是最有效的：

- 创建一个测量实例。命名为“时间”。其基础元素 ‘measurement_begin’ 将会包含取值“20061125161725”，‘measurement_end’ 将会是“20061125162237”，两个都是 DT_DATE 类型。
- 创建六个测量量实例：名称为“S1”，“S2”，“S3”，“S4”，“S5”的五个传感器量，每个都与“温度”这个量相关，也与单位“deg.C”相关，除此之外，第六个测量量应该是“ts”，描述一个样本的时间戳，与量“时间”及单位“秒”相关。
- 创建九个本地列实例，如下：

- “S1”与本地列 S1-lc 相关，其取值序列为：{20.0； 20.3； 20.8； 20.6； 20.9； 23.1}；
- “S2”与本地列 S2-lc 相关，其取值序列为：{15.5； 15.8； 15.9； 16.0； 16.3； 15.8}；
- “S3”与本地列 S3-lc 相关，其取值序列为：{0.12； 0.13； 0.18}；
- “S4”与本地列 S4-lc 相关，其取值序列为：{15.51； 15.84； 15.93； 15.88}；
- “S5”与本地列 S5-lc 相关，其取值序列为：{31.1； 30.0； 29.2； 32.1； 28.2； 28.3}。
- “ts”与四个本地列相关：
 - ts-lc1，取值为{0； 60； 120； 180； 240； 300}；
 - ts-lc2，取值为{0； 120； 240}；

- 3) ts-lc3, 取值为{0; 60; 120; 300};
 - 4) ts-lc4, 取值为{12; 72; 132; 192; 252; 312}。
- 这四个本地列被指定为‘独立列’。

- 创建四个子矩阵实例：
- a) “sub1”包含本地列 ts-lc1, s1-lc1, s2-lc1, 有 6 行；
 - b) “sub2”包含本地列 ts-lc2, s3-lc1, 有 3 行；
 - c) “sub3”包含本地列 ts-lc3, s4-lc1, 有 4 行；
 - d) “sub4”包含本地列 ts-lc4, s5-lc1, 有 6 行。

子矩阵“sub1”描述见表 1（标题给出了测量量的名称）。

表1 子矩阵“sub1”描述

ts	s1	s2
0	20.0	15.5
60	20.3	15.6
120	20.8	15.9
180	20.6	16.0
240	20.9	16.3
300	23.1	15.8

子矩阵“sub2”描述见表 2（标题给出了测量量的名称）。

表2 子矩阵“sub2”描述

t _s	s3
0	0.12
120	0.13
240	0.18

子矩阵“sub3”描述见表 3（标题给出了测量量的名称）。

表3 子矩阵“sub3”描述

t _s	s4
0	15.51
60	15.84
120	15.93
300	15.88

子矩阵“sub4”描述见表 4（标题给出了测量量的名称）。

表4 子矩阵“sub4”描述

ts	s5
12	31.1
72	30.0
132	29.2
192	32.1
252	28.2
312	28.3

值矩阵描述见表 5（标题给出了测量量的名称），其包含 50%以内的空字段。

表5 值矩阵描述

ts	s1	s2	s3	s4	s5
0	20.0	15.5	0.12	15.51	
12					31.1
60	20.3	15.6		15.84	
72					30.0
120	20.6	15.9	0.13	15.93	
132					29.2
180	20.6	16.0			
192					32.1
240	20.9	16.3	0.18		
252					28.2
300	23.0	15.8		15.88	
312					28.3

客户应用可以进一步将‘ts’列的时间戳数据与‘measurement_begin’的取值（也有可能是环境列的‘时区’信息）组合，以给出每个样本的绝对时间。此外，应用每个测量量的单位信息，可以呈现，例如，以 deg.f 或 Kelvin 表示的温度。

注：本地列的名称必须与相应的测量量的名称相同。在该示例中，‘xx-lc’表达仅仅用来标识本地列。

5.3.5 应用元素之间的关系

ASAM ODS 指定了元素之间的三种类型关系。

（1）父子关系

如果在元素之间找到这种关系，其中一个元素指定与另一个元素存在子关系，而另一个元素指定（相反）存在相反关系父关系。

一个父实例可以引用任何数量的子实例；一个子实例只能引用一个父实例（1：n 关系）。它们类似 UML‘构成’关系。

这种关系被用来识别唯一的父实例。一个应用元素的所有子实例引用同一个父实例，至少必须结合名称和版本区别所有子实例。这可以通过提供‘ASAM 路径’，识别单一实例。

所有 FATHER_CHILD 关系在 ODS 基础模型；应用模型不能添加这种类型的任何扩展应用关系。更多细节可在第 7 章的基础模型规范中找到。

(2) INFO 关系

如果在元素之间找到这种关系，这既不是一种定向关系（在这种情况下，其中一个元素指定与 INFO_TO 关系的的关系，而另一个元素指定关系 INFO_FROM），也不是一种非定向关系（在这种情况下，两个元素指定与关系 INFO-REL 的关系）。

INFO 关系可能出现定向 1: n 关系或非定向 n: m 关系。它们类似 UML‘关联’关系。它们为特定实例提供存储内可能相关的信息的参考。

ODS 基础模型已经指定了几种 INFO 关系，应用模型也可以进一步添加这种关系。

(3) 继承关系

如果在元素之间找到这种关系，其中一个元素指定与关系 SUBTYPE 的关系，而另一个元素指定关系 SUPERTYPE。一个 SUPERTYPE 实例可以引用任何量的 SUBTYPE 实例。一个 SUBTYPE 实例只能引用一个 SUPERTYPE 实例。

虽然它们类似面向对象编程语言的继承概念，但是，它们在多个方面的表现不同。要理解继承关系，建议阅读第 5.7 节及 ASAM ODS 规范其他章节中具体的说明继承关系的相关小节。

ODS 基础模型不存在继承关系。它们的概念只用于应用模型。

5.4 模型中的属性和关系

5.4.1 属性规则

与元素类似，在应用级和实例级的属性也有规则。由于应用领域范围广泛，不可能规定基础模型内的所有属性。因此，下列规则控制基本属性的命名及特定应用属性的增加。

所有从一个模型级过渡为下一个模型级的，可以添加属性。

基础模型	→	应用模型	→	实例
基本属性		基本应用属性		基本应用属性的取值
		扩展的应用属性		扩展的应用属性的取值
				实例属性的取值

规则如下：

- 基本应用属性的名称可以与相应的基本属性的名称不同。这种命名信息存储在元信息内；
- 可以在应用模型内添加项目-具体应用属性。这种信息也存储在元信息内；
- 任何实例都可以添加属性。在这种情况下，属性名称、类型及取值是必要的。它们存储在数据内。

5.4.2 所有基础元素的共同属性

所有基础元素（因此，所有应用元素）主要有下列基本属性：

- 标识：每个应用元素的实例有唯一 ID；

- 名称：实例的名称，只在层次结构中的前任内具有唯一性；
- 描述：实例的说明文本（可选）；
- 版本：实例的版本（可选）；
- 版本日期：版本变更的日期（可选）；
- MIME 类型：实例的多重目的因特网邮件扩展（MIME）类型（可选）；
- 外部引用：引用外部信息（可选）；
- 对象类型：该实例所属的应用元素的 ID（可选）。

基础元素派生的每个应用元素都可以使用该基础元素的基本属性。

每个基础元素都有一个（ASAM ODS 预定义的）基本 ID（BID），该 ID 是唯一的，以识别基础元素，不需要比较字符串（名称）。

第 7 章描述了基础元素、基本 ID（BIDs）及其基本属性。

5.4.3 应用属性

每个应用元素有一个应用属性集。

一个应用属性既可以是一个基本应用属性，也可以是一个扩展的应用属性。

基本应用属性是已经存在于相应的基础元素中的属性。在一个应用模型内，可以赋予这种基本应用属性一个指定的名称（而不是相应的基本属性的名称）；其含义和数据类型由相应的基本属性赋予，而且不能改变，除了下列情况：

——表 6 规定了几个基本属性的替代数据类型；服务器及客户应用必须支持这些替代数据类型。由应用模型设计师决定使用哪种数据类型。除了基础模型给定的替代数据类型之外，服务器及客户应用不得支持任何其他数据类型替代基础模型给定的数据类型；

表6 可替代的客户和服务数据数据类型

基本属性名称	默认数据枚举类型名称	允许的替代数据类型
external_references	DS_EXTERNALREFERENCE	DT_EXTERNALREFERENCE
dimension	DS_LONG	DT_LONG
default_dimension	DS_LONG	DT_LONG

——表 7 指定了几个基本属性的替代数据类型；在已安装的应用模型和服务端可应用这些替代数据类型，并由服务器传输；客户应用必须支持它们。在新设计的应用模型中不允许使用。

表7 可替代的服务数据数据类型

基本属性名称	默认数据枚举类型名称	允许的替代数据类型
id	DT_LONGLONG	DT_LONG
factor	DT_DOUBLE	DT_FLOAT
offset	DS_DOUBLE	DT_FLOAT
superuser_flag	DT_SHORT	DT_BYTE
version	DT_STRING	DT_LONG
start_offset	DT_LONGLONG	DT_LONG

基本属性名称	默认数据枚举类型名称	允许的替代数据类型
flags_start_offset	DT_LONGLONG	DT_LONG

允许使用基本应用属性的最大数目派生自任何基本属性。一些基本属性（说明设置为‘必选属性’的那些基本属性）必须从相应的应用元素中派生；其他基本属性不必要从应用元素中派生。

扩展的应用属性是在相应的基础元素中没有对应属性的属性。根据需要，一个应用元素可以添加很多个应用属性。对命名没有限制。

其数据类型可从 ASAM ODS 数据类型列表（参见第 5.5 节）中选择，除了条目 DT_UNKNOWN, DT_ID, 及 DS_ID。这种属性通常没有已知的含义。然而，即将问世的标准化应用模型可以指定扩展的应用属性集合，这些扩展的应用属性具有预先定义的名称及数据类型，还有明确定义的含义。

因此，应用属性的完整集合由基本应用属性和扩展的应用属性组成。

5.4.4 实例属性

一个应用元素的实例为其应用属性提供取值。

为增加灵活性，ASAM ODS 允许每个实例通过添加附加属性的方式扩展其属性集，这些附加属性被称为**实例属性**。

实例属性对相应的应用元素来说，是未知的（因此，对相应的基础元素里也是未知的）。它专属于该实例。因此，它必须由其名称、取值、数据类型、及（最好是）单位确定。对其命名没有限制。其数据类型可从 ASAM ODS 数据类型列表子集（参见第 9.5 节）中选择：允许的数据类型有 T_STRING, T_SHORT, T_FLOAT, T_BYTE, T_LONG, T_DOUBLE, T_DATE。一个实例属性的含义通常是未知的。

实例属性的概念允许在实例中添加在创建应用模型时无法预见的信息，这是非常灵活的。在大量实例中只有几个实例要求一种指定信息的情况下，它还可以用于减少存储空间。

因此，实例提供：

- 其相应应用元素的应用属性的取值；
- 其实例属性的取值（还有名称、数据类型、还可能有单位）。

5.4.5 关系规则

与属性类似，在 ASAM ODS 模型中也有关系规则。

从基础模型级过渡到应用模型级，可以添加关系。

基础模型	→	应用模型	→	实例
基本关系		基本应用关系		基本关系的引用实例的 IDs
		扩展的应用关系		引用实例的 IDs

规则如下：

- 基本应用关系的名称可以与相应的基本关系的名称不同。这种命名信息存储在元信息内；
- 可以在应用模型内添加项目-具体应用关系。这种信息也存储在元信息内；
- 任何实例都不能添加关系（属性除外）。

继承概念所使用关系适用的特殊规则（参照 5.4.6）。

第7章描述了基础元素的基本关系。

5.4.6 应用关系

每个应用元素都有一个应用关系集合。

每个应用关系既可以是一个基本应用关系，也可以是一个扩展的应用关系。每个关系还必须有一个逆关系。

基本应用关系是相应的基础元素已经提供的关系。在一个应用模型内，可以赋予这种基本应用关系一个指定的名称（不同于相应的基本关系的名称）；其基数（‘关系范围’）的含义和规则由相应的基本关系赋予，而且不能改变，除了下列情况：

- a) 对于在基础模型中已经定义为 **N: M** 关系的信息关系，允许减少其基数并使其成为 **1: N** 关系。
- b) 任何基本关系通常可以派生一个基本应用关系的最大值。一些基本关系（说明是‘必选属性’集合的那些基本关系）必须从相应的应用元素中派生；其他基本关系不必要从应用元素中派生。

在一些特殊情况下，一个基本关系可以派生一个以上的应用关系。例外情况如下：

- a) 基础元素允许应用模型采用树状结构，并且拥有一个被称为‘子’的基本关系。可以派生这种基本关系一次以上，因此，可以参照一个以上的应用元素。这些基本应用关系的名称必须互不相同。它们的基数也可以互不相同。

发生这种情况的基础元素有 **AoUnitUnderTest**、**AoUnitUnderTestPart**、**AoTestEquipment**、**AoTestEquipmentPart**、**AoTestDevice**、**AoTestSequence**、**AoTestSequencePart** 以及 **AoAny**。

- b) 基础元素 **AoUnit** 拥有一个（逆）基本关系，被称为‘parameters’，该关系引用基础元素 **AoParameter**。因为基础元素 **AoParameter** 可以派生一个以上的应用元素，所以，也可以多次派生这种基本关系。

- c) 基础元素 **AoEnvironment** 拥有被称为‘uuts’，‘equipments’，‘sequences’的基本关系，这些关系引用该环境内的所有被测对象、测试设备和测试序列。因为这个基础元素可能有一个以上的应用元素，所以，可以多次派生这三个关系。这些基本应用关系的名称必须互不相同。它们的基数也可以互不相同。

- d) 基础元素 **AoMeasurement** 拥有被称为‘units_under_test’，‘equipments’，‘sequences’的基本关系，这些基本关系可以用于引用几个该环境内的被测对象、测试设备、测试序列（及它们的零件）。因为这个基础元素可能有一个以上的应用元素，所以，可以多次派生这三个关系。

- e) 这些基本应用关系的名称必须互不相同。它们的基数也可以互不相同。

- f) 基础元素 **AoMeasurementQuantity** 拥有一个基本关系，被称为‘channel’，该基本关系可以被用来引用一台测试设备（或其中一个零件）。因为这个基础元素可能有一个以上的应用元素，所以，可以派生多次这个关系。这些基本应用关系的名称必须互不相同。它们的基数要么是（0: 1），要么是（1: 1）。

扩展的应用关系是一种在相应的基础元素中没有对应属性的关系。扩展的应用关系的关系类型总是‘INFO’类型。根据需要，在一个应用元素中可以添加很多个扩展的应用关系。对其命名没有限制。根据需要，可以指定引用的应用元素及关系的基数。

有一个对基数的限制：从一个应用元素建立的扩展的应用关系与自身不能成为 **N: M** 关系。

即将问世的标准化应用模型可以指定扩展的应用关系集合，每个扩展的引用关系拥有预定义名称、

引用的应用元素、基数，还有一个明确定义的含义。

因此，应用关系的完整集合由基本应用关系和扩展的应用关系组成。

5.5 数据类型的用法

5.5.1 ASAM ODS 中的数据类型的一般说明

在说明 ASAM ODS 规范的章节中，数据类型几乎无所不在。必须特别注意理解它们和正确使用它们。

首先，本节只考虑基本数据类型（因此，“数据类型”的意思就是“基本数据类型”）。然而，基本类型和非基本类型的区别不明显。

从面向对象的观点来看，每个类或接口可以被视为一种数据类型。本节不将类或接口视为基本数据类型。

从数据建模观点来看，每个数据结构（甚至最复杂的数据结构）可视为一种数据类型。再次说明，本节只将结构非常简单（因此，‘基本’的定义非常模糊）的数据结构视为一种基本数据类型。

ASAM ODS 明确区分了数据类型及其编号。

每个数据类型被赋予一个名称（例如，T_SHORT）和一个类型定义。类型定义把数据类型和先前已知原始类型联系起来。什么是原始类型？这取决于具体的语境。为 Java 语境定义数据类型，要求将数据类型与预定义的 Java 数据类型联系起来，为 CORBA 语境定义数据类型，要求将数据类型与 CORBA 预先定义的数据类型（依赖原始数据类型通过 CORBA 语言绑定，这些预先定义的数据类型本身进一步与语言相关）联系起来，为 STEP EXPRESS 语境定义数据类型，要求将数据类型与预定义的 STEP EXPRESS 数据类型联系起来，为 RPC 语境定义数据类型，要求将数据类型与 RPC 预定义的数据类型联系起来。在随后的说明 ASAM ODS 规范的各章中，为什么数据类型定义会不相同，这就是原因。

大部分数据类型被赋予唯一的编号（在可枚举的数据类型，‘datatype_enum’内），以便在运行时可以引用它们的编号。这也便于区别在运行时通过接口交换的数据的类型、或包含在物理存储（数据库、文件）的数据的类型。测量取值（例如，引擎速度）的数据类型保持在数据存储中，因此，不需要提前规定将来可能出现的所有情况。相反，产生数据的应用可以决定每种情况下首选的数据类型。其编号在应用编程接口（API）传输，或到物理存储同数据一起传输，接收应用能够根据当前的数据类型恰当处理数据。

每个编号都赋予一个名称（例如，DT_SHORT）；这些名称不能与数据类型（即使它们之间的关系是 1:1）的名称混淆。数据类型编号的名称有助于理解 switch-case-constructs 所用的枚举类型。人们宁愿读‘case DT_SHORT:’，而不是‘case 2:’。数据类型编号的名称也往往在用户界面呈现。因此，人们往往说‘数据类型 DT_SHORT’（严格来说，这是不正确的，但是，是可以容忍的）。

数据类型的定义：

- 基础模型（详见第 7 章）。这些数据类型最普遍，独立实施，并被用来指定应用模型。它们由 STEP EXPRESS 标记的一个恰当的 ENTITU 构造定义，因此，与 STEP EXPRESS 的原始类型有关系；
- RPC-API（详见第 12 章）。由于实施限制，不是所有由基础模型定义的数据类型都能用于 RPC-API。这可能会对通过 RPC-API 实现的数据的可视性和可访问性造成重大影响。因此，只有当应用模型不包含未获支持的数据类型时才能使用 RPC-API。另一方面，设计一个应用模

型，通过 **RPC-API** 访问数据的，应该不使用未获支持的数据类型；

——**OO-API**（见第 13 章）。此 **API** 更加灵活。除了在基础模型中定义数据类型之外，它定义并使用附加数据类型。这些附加类型中大部分是基础模型类型的序列（甚至是序列的序列），因此，允许使用更加简单的，更加通用的程序访问数据；

——**ATF/CLA**（见第 9 章）。为了不仅仅存储放在 **ATF** 文件的信息的本身，还要存储信息的数据类型，**ATF/CLA** 为使用的每个数据类型指定一个关键字。这些关键字与在基础模型定义的所有数据类型的数据类型编号（枚举类型编号！）的名称匹配；

——**ATF/XML**（见第 10 章）。为了不仅仅存储放在 **ATF** 文件的信息的本身，还要存储信息的数据类型，**ATF/XML** 为使用的每个数据类型指定一个关键字。这些关键字不仅仅与在基础模型定义的所有数据类型的数据类型编号（枚举类型编号！）的名称匹配，还包含只在 **OO-API** 使用的几个数据类型；

——物理存储规范（见第 6 章）。物理存储提供一个关系型数据库中的表格规范，该关系型数据库最终应该包含关于测试、测量、等方面的信息。然而，通常不预先定义存储信息的数据类型。因此，数据库必须不仅仅存储信息，还要存储信息所使用的数据类型。为此目的，只存储数据类型编号。物理存储允许指定所有在基础模型中定义的基本数据类型，还指定在 **OO-API** 指定的一阶序列数据类型；

——在此使用的数据类型必须与基础模型中定义的数据类型匹配。第 5.4.3 节解释例外。

在大多数情况下，数据类型的名称大写。在整个 ASAM ODS 规范中，数据类型枚举类型的名称同样用大写字母书写。

如果代表相同的数据类型（在整个 ASAM ODS 规范中，用相同顺序定义的枚举类型），数据类型编号总是一样的。在任何语境下，数据类型到其物理表示的映射应该是相同的。这意味着，例如，**T_SHORT** 应该映射到一个 16 位整数数据类型，不管是否在 Java、C++、Pascal 实施，或使用什么语言和平台。

在随后几个小节进一步解释数据类型的特殊用法。

第 5.5.2 节说明了 **API** 数据类型和基础模型数据类型之间的关系。同时，说明了 ASAM-级别标准化数据类型与 ASAM ODS 使用的数据类型之间的关系。

第 5.5.3 节和表 3 最后解释了 ASAM-级别标准化数据类型如何匹配 ASAM ODS 数据类型，ASAM ODS 未来将如何使用这些 ASAM 数据类型。

5.5.2 基础模型和 APIs 的数据类型

基础模型和 **APIs** 数据类型的名称，还有数据类型枚举类型的名称不总是一样。表 8 对基础模型与 **APIs** 的数据类型名称及数据类型的编号进行了比较。表 10 还说明了 ASAM 统一数据类型与 ASAM ODS 使用的数据类型之间的对应关系。ASAM 统一数据类型的说明可以要求 ASAM e.V. 提供。

表8 ASAM ODS 规范内的数据类型

ASAM 统一数据类型	OO-API API 数据类型 枚举类型	RPC-API API 数据类型 枚举类型	基础模型 基础模型数据类型 枚举类型

	-- DT_UNKNOWN (=0)	-- DT_RESERVED (=0)	-- DT_UNKNOWN (=0)
A_ASCIISTRING (*)	T_STRING DT_STRING (=1)	char[] DT_STRING (=1)	T_STRING DT_STRING (=1)
A_INT16	T_SHORT DT_SHORT (=2)	short DT_SHORT (=2)	T_SHORT DT_SHORT (=2)
A_FLOAT32	T_FLOAT DT_FLOAT (=3)	float DT_FLOAT (=3)	T_FLOAT DT_FLOAT (=3)
A_BOOLEAN	T_BOOLEAN DT_BOOLEAN (=4)		T_BOOLEAN DT_BOOLEAN (=4)
A_UINT8	T_BYTE DT_BYTE (=5)	unsigned char DT_BYTE (=5)	T_BYTE DT_BYTE (=5)
A_INT32	T_LONG DT_LONG (=6)	long DT_LONG (=6)	T_LONG DT_LONG (=6)
A_FLOAT64	T_DOUBLE DT_DOUBLE (=7)	double DT_DOUBLE (=7)	T_DOUBLE DT_DOUBLE (=7)
A_INT64 (*)	T_LONGLONG DT_LONGLONG (=8)		T_LONGLONG DT_LONGLONG (=8)
	-- DT_ID (=9)		-- DT_ID (=9)
	T_DATE DT_DATE (=10)	char[] DT_DATE (=10)	T_DATE DT_DATE (=10)
	T_BYTESTR DT_BYTESTR (=11)	bstream DT_BYTESTR (=11)	T_BYTESTR DT_BYTESTR (=11)
	T_BLOB DT_BLOB (=12)	Blob DT_BLOB (=12)	T_BLOB DT_BLOB (=12)
A_COMPLEX32 (*)	T_COMPLEX DT_COMPLEX (=13)		T_COMPLEX DT_COMPLEX (=13)
A_COMPLEX64 (*)	T_DCOMPLEX DT_DCOMPLEX (=14)		T_DCOMPLEX DT_DCOMPLEX (=14)
A_SEQUENCE_ASCIISTRING (*)	S_STRING DS_STRING (=15)		

A_SEQUENCE_INT16	S_SHORT DS_SHORT (=16)		(b)
A_SEQUENCE_FLOAT32	S_FLOAT DS_FLOAT (=17)		(b)
A_SEQUENCE_BOOLEAN	S_BOOLEAN DS_BOOLEAN (=18)		(b)
A_SEQUENCE_UINT8	S_BYTE DS_BYTE (=19)		(b)
A_SEQUENCE_INT32	S_LONG DS_LONG (=20)		(b)
A_SEQUENCE_FLOAT64	S_DOUBLE DS_DOUBLE (=21)		(b)
A_SEQUENCE_INT64 (*)	S_LONGLONG DS_LONGLONG (=22)		(a)
A_SEQUENCE_COMPLEX_32 (*)	S_COMPLEX DS_COMPLEX (=23)		(c)
A_SEQUENCE_COMPLEX_64 (*)	S_DCOMPLEX DS_DCOMPLEX (=24)		(c)
	-- DS_ID (=25)		
	S_DATE DS_DATE (=26)		(a)
	S_BYTESTR DS_BYTESTR (=27)		(b)
	T_ExternalReference DT_EXTERNALREFERENCE (=28)		T_EXTERNALREFERENCE DT_EXTERNAL_REFERENCE (=28)
	S_ExternalReference DS_EXTERNALREFERENCE (=29)		(a)
A_ENUM	T_LONG DT_ENUM (=30)		DT_ENUM (=30)
A_SEQUENCE_ENUM	S_LONG DS_ENUM (=31)		

备注:

(*): 需要注意

a) A_ASCIISTRING 是二进制的, 在写入一个文件时, 兼容相应的 OO-API 数据类型。然而, 这些数据类型与 API 数据类型不兼容, 原因如下:

- 1) A_ASCIISTRING 最终被指定为一个八进制的 CORBA 序列。一个 CORBA 编译器, 例如, 在 JAVA 中, 生成一个八位字节的数组。
- 2) T_STRING 被指定为 CORBA 字符串。一个 CORBA 编译器, 例如, 在 JAVA 中, 生成一个字符串类。
- 3) 如果一个应用采用处理字符串类的方法, 与处理一个八位字节的数组相比, 它用完全不同的方法来操作字符串。
- 4) 这点同样适用于相应的序列。

b) A_INT64 是二进制的, 兼容相应的 OO-API 数据类型。然而, 由于这些类型被定义为结构, 各结构组件的名称不同。这些类型与 API 类型不兼容。

这点同样适用于相应的序列。

- c) **A_COMPLEX32** 是二进制的，兼容相应的 OO-API 数据类型。然而，由于这些类型被定义为结构，各结构组件的名称不同。这些类型与 API 类型不兼容。

这点同样适用于相应的序列。

- d) **A_COMPLEX64** 是二进制的，兼容相应的 OO-API 数据类型。然而，由于这些类型被定义为结构，各结构组件的名称不同。这些类型与 API 类型不兼容。

这点同样适用于相应的序列。

- a), b), c): OO-API 的序列数据类型在 STEP EXPRESS 可以表示为：

LIST [0: ?] OF ### (在上述 (a) 情况下);

LIST [1: ?] OF ### (在上述 (b) 情况下);

LIST [2: ?] OF ### (在上述 (c) 情况下)。

其中，###表示一个恰当的基本数据类型。

所使用的数据类型中，大部分参照相应的 ASAM 统一数据类型指定。其他数据类型定义如下：

(1) T_LONGLONG

数据类型 T_LONGLONG 被用来表示一个 64 位有符号整数值。物理存储和 APIs 的相关章节提供更多关于其表示和用法的信息。

由于很多环境不支持 64 位整数数字，其由两个 32 位整数值组成，如下（使用公用对象请求代理体系结构接口定义语言（CORBA IDL））：

```
struct T_LONGLONG {
    T_LONG high;
    T_LONG low;
}
```

虽然，两个结构组件形式上都是有符号的整数值，但是，low 部分应该被视为没有符号的整数，以便 T_LONGLONG 总共拥有与一个 64 位有符号整数相同的存储表示。

(2) T_DATE

数据类型 T_DATE 允许存储日期和时间信息，带无限精度，具体见 ISO 8601。

通常使用一个字符串“YYYYMMDDhhmmsslllcccnnn…”表示。

各部分的含义是：

- YYYY：年份（0000…9999）；
- MM：月份（01…12）；
- DD：日（01…31）；
- hh：小时（00…23）；使用每天 24 小时时间表；
- mm：分钟（00…59）；
- ss：秒钟（00…59）；
- lll：毫秒（000…999）；
- ccc：微秒（000…999）；
- nnn：纳秒（000…999）；
- …（等等）。

可以减掉任何元素之后的字符串，因此，可以提供任何需要的解析度。最不确定的信息似乎是年份；如果字符串长度正好是 8 个字符，就精确到日期。

所有未给出部分被假定为最小值。因此，“200611031032”的意思是，2006年11月3日10时32分0秒的开始，“200610”的意思是，2006年10月1日的开始。

需要注意，在一些情况下，物理存储允许使用优化表示。物理存储相关章节提供更多关于这方面的信息。

(3) T_BYTESTR

数据类型 T_BYTESTR 允许存储八位字节（八位位组）序列。因此，八位字节数，即使用的存储量是可变的。物理存储和 APIs 之间字节流的表示方式和使用方法，还有允许的最大八位字节数都不同。物理存储和 APIs 相关章节提供更多关于字节流的表示方式和用法方面的信息。

需要注意，八位字节的一个序列可能用两种方式表示：表示为 T_BYTESTR 或 DS_BYTE。物理存储提供它们的不同表示方法，对 T_BYTESTR 的存储用优化的表示方法。应用属性可以使用两种类型，本地列只能包含 T_BYTESTR 类型的八位字节序列。

根据所使用的物理存储和 API，可能适用进一步限制。

(4) T_BLOB

数据类型 t_blob 允许使用描述性头存储字节序列（八位字节）。

头的长度和序列中的字节数（因此使用的存储）是可变的。blob 的表示和使用方式以及头的长度以及允许的最大字节数在物理存储和 API 之间是不同的。关于其表示方式和用法方面的更多信息在物理存储和 APIs 相关章节给出。

(5) T_COMPLEX

数据类型 T_COMPLEX 允许以单精度格式存储复数。

物理存储和 OO-API 相关章节提供更多关于其表示方式和用法方面的信息。

其由两个单精度浮点值组成，如下（使用公用对象请求代理体系结构接口定义语言（CORBA IDL））：

```
struct T_COMPLEX {
    T_FLOAT r;
    T_FLOAT i;
};
```

r 是复数的实数部分，i 是复数的虚数部分。

(6) T_DCOMPLEX

数据类型 T_COMPLEX 允许存储双精度格式的复数。

物理存储和 OO-API 相关章节提供更多关于其表示方式和用法方面的信息。

其由两个双精度浮点值组成，如下（使用公用对象请求代理体系结构接口定义语言（CORBA IDL））：

```
struct T_COMPLEX {
    T_DOUBLE r;
    T_DOUBLE i;
};
```

r 是复数的实数部分，i 是复数的虚数部分。

(7) T_EXTERNAL REFERENCE

数据类型 T_ExternalReference 允许存储外部对象的引用（比如，文件）。它结合了文字说明、MIME 类型、及找到外部对象的位置规范。

物理存储和 OO-API 相关章节提供更多关于其表示方式和用法方面的信息。

其由以下元素组成：（使用公用对象请求代理体系结构接口定义语言（CORBA IDL））：

```
struct T_ExternalReference {
    T_STRING description;
    T_STRING mimeType;
    T_STRING location;
};
```

Description 是外部引用所引用的对象的文字说明，mimeType 是所引用对象的实例的多重目的因特网邮件扩展（MIME）类型，location 提供关于如何找到所引用的对象的信息（ASAM 路径或统一资源定位器（URL）；更多关于外部文件的绝对和相对路径名称信息，参见第 5.10.6 节）。

(8) S_*: SEQUENCE TYPES

ODS 数据类型，名称以 ‘S_’ 开头的，是相应的 ‘T_’ 数据类型的序列。物理存储和 OO-API 相关章节提供更多关于其表示方式和用法方面的信息。

应用公用对象请求代理体系结构（CORBA）的序列定义及其各自的语言绑定。

5.5.3 标准化 ASAM 数据类型

在 2000 年，ASAM e. V. 启动一个项目，统一应用于不同工作领域的数据类型（ACI, GDI, MCD, ODS, 等）。于是，在 2002 年正式发布了统一文件，对现有 ASAM ODS 数据类型与为在整个 ASAM 使用而定义的数据类型进行了比较，指定了 ASAM 级别标准化数据类型（‘ASAM 统一数据类型’）。完整集合的详细信息可以要求 ASAM e. V. 提供。

迁移路径：

在 ASAM ODS 规范很多部分使用数据类型枚举元素的名称（比如，DT_SHORT）。在 ATF/CLA、ATF/XML、物理存储、特定应用领域的应用模型部分，这种情况将持续下去，因为，ASAM 统一数据类型规范不涵盖枚举名称和其值。

在世界范围内，由于 ODS 应用时间很长，有很多实现的 ASAM 开放式数据服务，结果证明，在现有所实现的 ASAM 开放式数据服务（ODS）中，几乎不可能改变数据类型。因此，ASAM 开放式数据服务（ODS）决定继续支持 ASAM 开放式数据服务（ODS）数据类型，并将它们应用于面向对象-应用编程接口（OO-API），支持开放网络计算环境的远程过程调用（ONC RPC）规范、远程过程调用应用编程接口（RPC-API），以及远程过程调用（RPC）。

5.6 应用接口

5.6.1 基本信息

ASAM 开放式数据服务（ODS）提供应用编程接口（APIs），为任何应用访问数据提供一个标准化、有效的设计路径，并隐藏存储数据的特定方式。

5.6.1.1 应用接口在 ASAM ODS 的定位

ASAM ODS 基础模型是一个普通的、独立于应用的、逻辑的数据模型，已经明确定义了 ASAM ODS 中处理的数据结构，即使它只是粗略地概述了将要存储的信息。基础模型是一个非常静态的模型，预计

不会经常发生改变。然而，由于有可能发生改变，已经定义了通过 APIs 存取基础模型的各种方法。

除了存取基础模型的方法，还在 ASAM ODS 中定义了有助于一个应用从基础模型中创建一个应用具体数据模型的方法。该应用模型说明了应用的完整的数据模型。ASAM ODS 还提供了存取实例的其他方法，那就是在应用模型的数据存储。

图 25 表明，一个应用只通过调用 ASAM ODS API 的功能存取 ASAM ODS 数据。API 隐藏任何网络传输；在应用和物理数据存储位于不同系统的情况下，CORBA（使用公用对象请求代理体系结构）（使用 OO-API（面向对象-应用编程接口）时），或 RPC（远程过程调用）（使用 RPC-API（远程过程调用-应用编程接口）时）技术，注重客户端请求是否正确转化为服务器端请求和响应数据的传输。



图25 API 的定位

API 提供与硬件及数据格式（如文件或数据库）无关的 ASAM ODS 组件；所有的 API 数据的存取由 ASAM ODS API 转发到数据服务器。在此过程中，ASAM ODS API 从硬件相关的描述中抽取不同计算机平台的 ASAM 数据类型（例如，4-字节的整数）。ASAM ODS API 将与服务器相关的描述转换为与客户端相关的描述。ASAM ODS API 不改变信息的内容和结构。

最后，数据服务器抽象出数据格式和物理使用的数据类型。这意味着，它将数据转化一个符合 ASAM 标准结构和 ASAM 内部使用的数据类型。

ODS 服务器实施实际的数据存取路由。通常，支持 ASAM 传输格式（ATF）和 ASAM 数据基本格式（物理存储），但是，除了这两种格式之外，还应该支持现有的其他格式——外来格式，只要它们能够通过 ASAM ODS API 提供给客户。API 及其应用接口的用途在下节说明。

5.6.1.2 API 的任务和一般注意事项

在此概念上定义的 API 提供支持应用的功能。

根据基础模型规则，有一些函数可以创建应用模型。ASAM ODS 将一些关于应用模型的基本信息与数据一起存储在 environment。如果能够访问该 environment，那么，该信息可以被用来自动创建应用模型。

写入、读取及变更数据功能也是 API 的一部分功能。

API 操作对象主要可以细分为以下三组：

- a) 基础元素；
- b) 应用元素；
- c) 实例。

基础元素和应用元素定义了将在应用处理的数据类型。基础元素是在 ASAM ODS 定义的数据结构，

ASAM ODS 创建了基础模型。它们定义了 ASAM ODS 支持的各个数据对象之间的所有关系。还有，它们为每个基础元素定义了几个独立于应用的基本属性，这些基本属性包含在从该基础元素派生出来的每个应用元素中。应用能够增加其他的应用具体的属性到每个应用元素。

在每个应用元素中对所有关于应用模型的信息进行编码。应用将要处理的数据将作为属性值存储在這些应用元素的实例中。除了相应的应用元素明确定义的属性之外，一个实例还能够包含自身的实例具体的属性。

示例：基础元素、应用元素及实例。

图 26 表明了基础元素有两个基本属性。从该基础元素中，创建了一个应用元素‘车辆’，该应用元素使用基本属性‘名字’（不变）和‘说明’（现在命名为‘shortdesc’），并增加两个额外属性（扩展应用属性）‘司机’和‘车辆类型’。

基础元素和应用元素都不保存属性值。相反，为该目的，创建了应用元素‘车辆’的实例。其中一个实例如图示，它包含了每个应用属性的值，除此之外，它还增加了另外一个属性（一个实例属性）命名为‘颜色’及相对于的值。该实例属性是专属这个实例的；其他实例可以不用它。

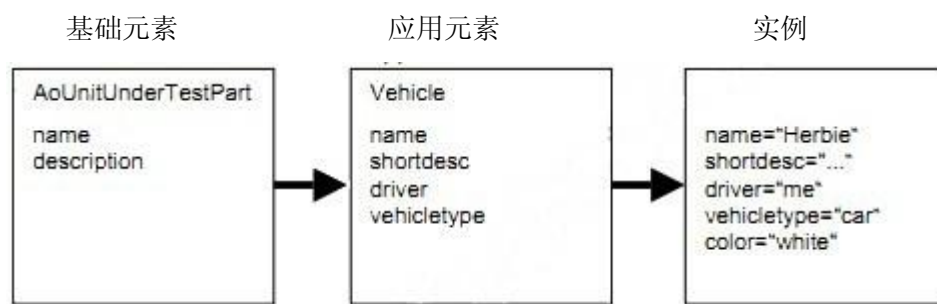


图26 基础元素、应用元素、及实例

该接口面向应用，应用编程接口（API），并提供了创建、存储和读取应用模型、及实例的方法。存储和读取存储在这些模型的数据的方法也是该接口的一部分。基础模型对于服务器可以是隐式或显式，并只能被应用读取。

对于一个 ASAM ODS 应用，在一台确定的计算机上运行时，它应该可能存取存储在其他计算机或甚至其他平台的数据。因此，使用了客户端-服务器结构。

5.6.1.3 数据模型的要求

ASAM ODS 定义了基础模型；必须从基础模型中构建应用模型。在定义一个新的应用模型过程中，ASAM ODS 服务器必须保证与基础模型和其规则的一致性。读取时，必须通过最便捷的方式重新创建存储的应用模型。

如在第 5.3.4 节（对技术原因）的说明，一次测量的所有数据并不是都作为一个完整的矩阵存储在一起，而是会分配到不同的子矩阵。

对于应用，然而，需要提供简易方法存取完整数据。应用通常不关注数据存储的效率；这是为什么在应用提供值矩阵的原因；在运行期间，它需要 ASAM ODS 服务器自动组合成一个子矩阵。

基础模型的一部分包含了允许对控制 ASAM ODS 对象的存取采取非常灵活控制措施（可选）的机制的信息。

5.6.1.4 基于应用的 API 要求和通常用法

有很多不同的应用类型，它们对 API 的要求也不同。最常见的应用类型往往可以根据它们的通常用法判断：

- a) 测量定义：创建一个新的应用模型；对于每个环境，定义只能进行一次；
- b) 测量准备：创建测试说明，创建模式结构，变更测试说明；
- c) 测量数据记录：写入测量数据；
- d) 数据的可视化：读取测量量的所有的测量数据，读取其属性；
- e) 数据分析：读取测量数据；创建新数据，连续读取；读取显式时间信息，在一个给定的时间网格中读取，如 $t_0-\Delta t$ -网格（比如数学过程）；
- f) 数据准备：读取将要进一步处理（排序、压缩、校准等）的精选测量数据或属性组；
- g) 数据编辑：读取并变更单个测量值（比如，数据编辑器）；
- h) 数据储存转换器：读取或写入所有的数据（一次完成）。

在测量期间的数据记录通常是一个时间-关键过程，在此期间，必须在非常短的时间段内完成测量值的存储。即使这不是 ASAM ODS 支持这些时间-关键操作的（主要）目的，ASAM ODS 也可以在其测量记录的限制范围内使用。

此外，大部分应用必须能够处理一个现有的应用模型，这意味着确定从 API 中创建一个应用模型在其范围内进行导航，并寻找所要求的数据。

具体应用类型的这些不同的动作对 API 的功能有下列要求：

- a) 处理应用模型
 - 通过应用模型导航；
- b) 处理数据
 - 1) 连续读取一个或几个测量量的所有数据；在这种情况下，处理时间最重要；
 - 2) 读取精选测量数据，此处，精选准则的符合性应放在首位，但是，进行处理的时间也重要；
 - 3) 顺序写入测量数据；
 - 4) 随机存取写入或变更的单一测量数据；由于值矩阵（通常）由几个子矩阵组成，变更的测量值只可能受到一些限制的制约；
 - 5) 读取属性；一个个或者成组读取；
 - 6) 写入或变更属性。

5.6.1.5 物理存储的要求

物理存储的要求实际上是支持相应平台上相应文件格式的数据服务器的要求。应该列出以下要求：

- a) 不是所有的数据服务器都能够写入数据；
- b) 数据服务器、特别是外来格式服务器，通常不支持 ASAM ODS 提供的可以写入的所有数据类型；在读取期间，当然，在读取期间只有存储的数据需要支持这些数据类型。

5.6.1.6 一般要求

除了前面的要求之外，还有几个一般要求：

- a) API 功能必须与机器无关;
- b) API 功能的调用必须与实施无关;
- c) 不允许功能假设将要处理的数据可以完全存储在它们的内存。特别在排序和连接时, 分区通常导致大量时间延迟, 因此, 应该避免;
- d) 值矩阵和视图的创建有些非常复杂, 并耗时(例如, 连接、合并或排序), 需要足够的信息存储在内存, 因此, 才不必在每次数据存取时执行;
- e) 考虑派生了当前应用模型的基础模型的版本, 需要集成版本管理。

5.6.2 APIs 工作

本节将进一步说明 APIs 的主要功能。

除了处理应用模型之外, 存取权限的管理和评价、与测量相关的所有数据的管理都是 ASAM ODS 的主要功能。在测量期间将要处理的信息是完全不同的。因此, 数据被划分为两类: 测量数据和属性:

- a) 在测量数据里, 一个测量量往往结合了所有的由完整数据向量(通常是时间流)组成的数据。这类包括实际的测量结果, 引用量及处理过的数据(如数学计算的结果);
- b) 术语属性涵盖了一次测量的框架里所有的信息, 例如, 关于测量量(名字、单位)的信息, 受测设备、测试所使用的设备, 等等。这类型数据通常由少量取值组成。

API 最重要的功能可以划分为四个领域:

- a) 存取 ASAM ODS 数据;
- b) 应用模型处理;
- c) 测量数据管理;
- d) 属性管理。

在解释 API 的这些要求之前, 先说明一些关于如何存取 ASAM ODS 数据的一般信息。详情请参考第 12 章和第 13 章。

5.6.2.1 存取 ASAM ODS 数据

存取一个现有的应用模型或其属性, 应用需要连接一台 ASAM ODS 服务器。由于所使用的 API 和底层的分布式技术不同, 连接过程详情可能会不同。

使用 OO-API(面向对象-应用编程接口)及 CORBA(公用对象请求代理体系结构)通常要求连接 CORBA 域名服务, 并请求一个 ASAM ODS 工厂对象(AoFactory); 这允许使用 ASAM ODS 服务器创建一个新的会话, 然后, 该会话使基础模型、应用模型、及实例能够被存取。

使用 RPC-API 时, 必须创建一个客户端(假设一个节点名称和一个 RPC 编号)。

该客户端需要创建一个与服务器的会话(在那个语境下被称为环境); 在客户端进一步使用那个过程去存取应用模型和实例期间, env_ld 进行交换。

5.6.2.2 应用模型处理

应用模型是一个逻辑的、提供 ASAM ODS 管理的应用数据说明的应用具体的数据模型。不同应用处理应用模型可以分为两个部分: 创建或修改应用模型, 通过一个应用模型导航。

(1) 创建/修改应用模型

下面的例子显示了基础模型的一部分和一个可能从该基础模型中派生出来的应用模型。说明了 ASAM ODS 结构的两个重要方面，以阐明如何创建一个应用模型：

- a) 针对应用元素分配应用具体名称，有可能使用应用具体术语而不是基础模型的全部术语；
- b) 通过从一个已经递归定义的基础元素（例如，测试部分单位）中创建层次结构，可以创建一个应用元素的完整的树状结构。

示例：创建一个应用模型见图 27。

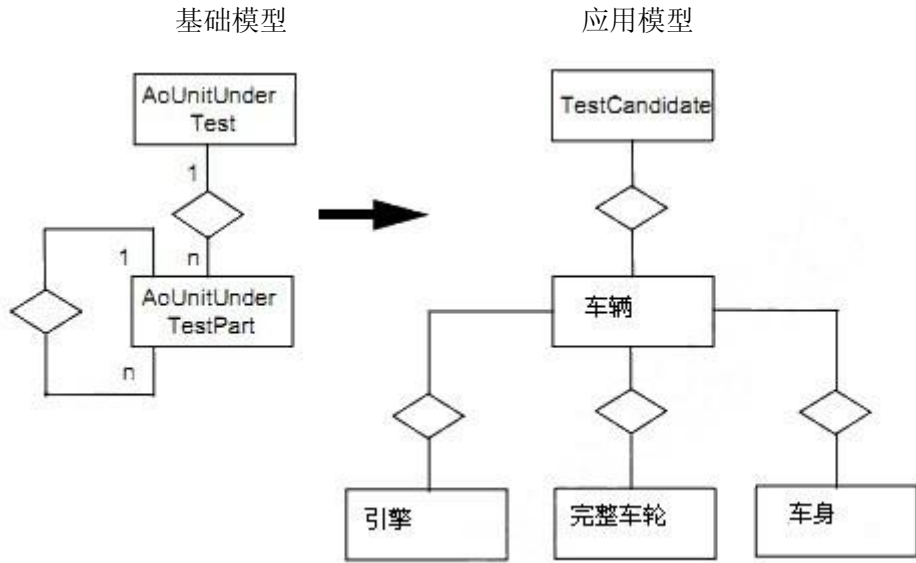


图27 创建一个模型

ASAM ODS 定义应用模型中的第三方面不包含在此例子中（参见上一节的例子）：

——应用具体属性的定义更加具体地说明了应用元素。

(2) 通过应用模型导航

一些应用，特别是那些用来评价数据的应用，必须能够操作任何应用模型。因此，提供了一些方法使应用能够请求创建模型或在类似于一个目录的应用模型之内树进行导航。这包括列出一个应用模型的所有子元素，或从一个确定的基础元素派生出来的所有子元素。此外，其中一部分，是列出该应用元素的所有实例。为了完成这些列表工作，可以指定元素或实例名称的搜索模式来限制筛选。

一个应用模型的所有元素的名称必须唯一，这意味着，通过名称，一个应用元素已经足够合格了。

对于处理应用模型，通过 API 提供下列功能：

- 找到所有的应用元素；
- 找到一个应用与元素的所有属性；
- 请求属性信息（数据类型，等）；
- 找到一个应用元素的所有子元素（通过搜索模式）；
- 找到一个已经从一个确定的基础元素中派生的应用元素的所有子元素（通过搜索模式）；
- 找到一个应用元素的所有实例（通过搜索模式）。

5.6.2.3 测量数据管理

ASAM ODS 的主要任务是管理测量数据。根据 ASAM ODS 数据模型，测量数据将保存在 AoLocalColumn 的实例中。一个本地列正好包含一个测量量的数据。一次测量一般会产生多于一个测量量的数据；它们中每个测量量的数据被保存在一个本地列中。存储同时产生的数据（这些数据值有相同的编号，相应的数据值相同时间戳相关）的本地列与相同的子矩阵相关。子矩阵最终与创建了它们的测量相关。关于存储测量数据的更多信息，请参见第 5.3.4 节

在数据读取、写入、变更及删除期间，测量数据的存取完全不同：

a) 读取：

- 1) 从一次具体的测量中请求数据；从该数据中取得的子矩阵不会被命名；
- 2) 可以一次读取几个测量量（如果需要）；
- 3) 通过一个筛选准则筛选所要求的本地列。

b) 写入（增加新的量或新的本地列到测量中）：

- 1) 并行写入几个本地列；
- 2) 本地列可以存储为新增加的本地列，或添加到现有的子矩阵中；
- 3) 在一个子矩阵中给定的位置添加/增加一个或一个意思的‘测量点’（行）；
- 4) 应该存储数据的子矩阵必须已知。

c) 变更（变更现有量和现有本地列）：

- 1) 取值本身，还有值的标志，例如可视化和有效性，都可以改变。
- 2) 可以变更点法测量（一个子矩阵中的一个或多个的‘测量点’（行））；
- 3) 通常，一个本地列（各自的测量量）将依次变更；
- 4) 只有测量才会被命名；从内部关系可以知道子矩阵；
- 5) 能够给定一个筛选准则；
- 6) 已经变更的数据可以作为量的新版本存储；旧数据可以保留。

d) 删除

- 1) 可以删除一个完整的子矩阵；
- 2) 一个或多个的‘测量点’（行）可以从一个子矩阵中删除；
- 3) 创建本地列的子矩阵必须由客户端指定。

在所有的数据存取过程中，除了取值之外，值的标志也将被读取或写入；值的标志无法访问自己。如果所有的取值拥有相同的标志，例如，所有取值都有效、可视并可修改，那么，标志将会被作为所有取值的一个标量处理。如果不同取值的标志也不同，那么，每个取值将插入自己的一组标志，这意味着，值的标志创建了一个完整的向量。

这些不同类型的存取导致了完全不同的 API 方法的一个集合。

5.6.2.4 管理属性

ASAM ODS 可以对几乎所有的基础元素的属性进行管理。一些元素只有对属性进行管理的目的，例如，受测设备元素，受测设备单位，测试设备，等。

属性与相应的对象以名称-取值组的形式一起存储。可以将单位目录中的一个单位分配给一个属性。

（1）预定义属性处理

每个实例从相应的应用元素中取得几个属性。这些属性是基础元素已经定义的基本属性（基本应用属性）和相对于应用元素的应用具体的属性（扩展应用属性）的结合。由于一个应用元素的每个实例还可以包含其他实例属性，所以，一个应用可能需要列出一个实例的所有属性。为了方便，可以使用按属性名称进行搜索的模式。

可以在一个应用模型之内对基本属性重新命名。一个应用必须能够确定基本属性的基本名称与应用名称之间的关系。通过这两个名称都必须能够存取基本属性。

（2）实例属性处理

除了预定义属性之外，一个应用还能够存储扩展属性作为一个实例中的实例属性。为创建和删除实例属性提供了方法。实例属性的取值直接在创建期间设置。

在读取和搜索期间，实例属性与预定义属性同等对待。

（3）关系处理

关系（可以被视为一个元素一个特定类型的属性）能够指定实例之间是如何彼此相关的（例如，测量量与相对于单位之间的关系）。

这些关系可以划分为三类：

- 父子关系：一些应用元素及其相关的实例在数据模型的层次结构中刚好只有一个父元素。此关系类型被用来说明它们之间的相互引用；
- 属性关系：如果一个实例有某些属性，例如，一个测量量有某个测量当中使用的单位，那么，会用到这种关系；
- 信息关系：一些信息不单独存储在对应的实例中。而是创建了一个独立的、可以使用几次的对象，例如，一个受测设备或一个量组的说明。信息关系通常说明这些关系。

基础模型定义那些可能的关系；而应用模型将会定义那些实际可用的关系。在每个应用模型中，必选关系必须可用。

通过使用 API，可以创建或删除关系。

（4）通过属性值进行搜索

在很多情况下，列出一个应用元素的所有实例并不起到作用。一般要求是：

- “列出 1993 年所有的车辆模型”；
- “列出属于基本单位‘abc’的所有单位”；
- 列出涉及引擎‘xyz’的所有测量。

限制方法调用的结果为有限数目的项目。列出实例的方法包括一个限定其属性值请求实例的选择条件。不仅仅可以通过使用实例本身的属性来指定筛选准则，还可以通过使用与该实例相关的其他应用元素的实例的属性来指定筛选准则。

（5）提供的功能

以下列表总结了属性处理的最重要的功能：

- 读取属性值；
- 写入属性值；
- 创建实例属性；

- 读取实例属性；
- 删除实例属性；
- 列出一个实例现有的所有属性（通过通配符）；
- 关系分配；
- 关系删除；
- 区分必选和可选关系；
- 使用属性值进行筛选。

5.7 继承的概念

5.7.1 目的

ASAM ODS 中引入继承的概念，其主要目的是为构造数据提供另外一个机制。除此之外，在特定情况下继承机制可以减少所需的存储空间。

5.7.1.1 在应用模型中增加子类型结构

没有继承机制概念，一个应用模型可以指定应用元素（使用应用属性），并将它们彼此联系起来（通过一个父子关系或 INFO 关系）。这类应用元素没有太多的类似性信息。

继承机制概念允许指定有共同关系、有几个共同属性但扩展属性不同的应用元素。在这种情况下，一个应用元素被定义为‘超类’，扩展应用元素可以被定义为这些超类元素的子类。由于在面向对象系统中，一个超类包含了所有的关系和共同属性，而继承的子类包含了不与其他子类共享的扩展属性。

ASAM ODS 只允许每个应用元素指定一个超类。因此，不支持多个继承机制。此外，只提供一个继承级别。因此，一个子类元素不可能成为另一个应用元素的超类。

数据最后存储在实例时，这些实例可以实例化超类或任何子类。在它们实例化超类的情况下，只提供共同属性值。如果它们实例化子类，共同属性值和子类具体属性值都提供。

存取数据时，可以请求一个子类的一个实例的属性值，或超类的一个实例的属性值。在处理子类的情况下，可以存取共同属性值和子类具体属性值。在处理超类的情况下，只可以存取共同属性值。然而，在这种情况下，超类的实例的属性值和子类的实例的共同属性值都可以存取。

5.7.2 具有继承机制的数据建模

熟悉面向对象继承原理的人都非常容易理解数据建模。本节简单解释数据建模。为了理解 ASAM ODS 物理存储的优化效果，需要掌握物理存储的基本知识。这些基本知识在第 6 章有详细说明。

在信息保存在一个应用元素的情况下（因此，所需的应用属性数量），根据不同的使用情况会有所不同。这些应用属性将分配给两个或两个以上的应用元素。其中包含了任何使用情况都需要的应用属性的一个元素将是超类应用元素。其他的元素是子类应用元素。它们与超类应用元素有关联。

当定义应用模型时，超类应用元素将被赋予所有继承子类应用元素的公共属性。子类应用元素只被赋予它们特有的应用元素。它们不会被赋予共同的应用属性，但是，它们会从超类应用元素继承共同应用属性。

当使用应用模型时，必须区分下列情况：

- a) 如果用例只要求存储超类应用元素包含的信息，那么只创建并存储该元素的一个实例（通

过 API 方法);

- b) 如果用例要求存储超类应用元素及其其中一个子类应用元素包含的信息,那么只创建并存储该子类应用元素的一个实例(通过 API 方法)。由于子类应用元素的实例也提供公共属性值(在超类中定义),因此自动存储了超类应用元素中定义的属性信息。

示例: 具有继承机制的应用模型见图 28。

在不同的试验中应使用应用元素“发动机”来描述发动机的特征,并为测量数据提供说明信息。在大多数情况下,用基本应用属性“EID”、“ENAME”以及扩展应用属性“EBore”、“EStroke”就能充分描述“发动机”。然而在某些情况下,汽缸 1 和 2 的内径可能与其他缸的内径不同。因此要通过扩展应用属性“EBore1”和“EBore2”说明附加信息,此外还要通过基本应用属性“EVersion”增加版本信息。

创建两个应用元素“EngineSuper”和“EngineSub”,它们都从 AoUnitUnderTest 中派生。EngineSub 的一个实例将包含 EID、ENAME、EBore、EStroke、Eversion、EBore1 和 EBore2 的值。EngineSuper 的一个实例将只包含 EID、ENAME、EBore 和 EStroke 的值。附加的基本应用属性 EObjType,提供关于是否存取超类或子类应用元素的信息。

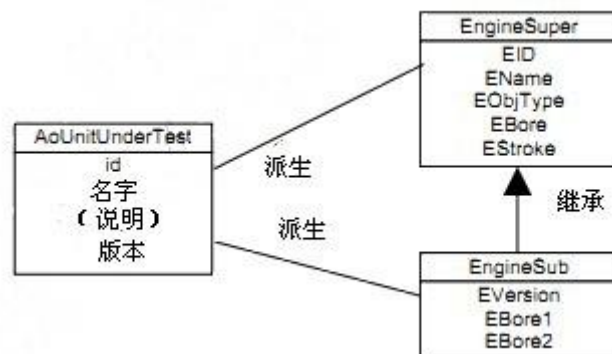


图28 具有继承机制的应用模型

在创建一个应用模型并包括继承机制时,必须遵循以下规则:

- 超类应用元素和子类应用元素必须从相同的基础元素中派生;
- 继承限制到一个级别。一个子类应用元素不可能是另外一个应用元素的超类元素;
- 只支持单一继承。一个子类应用元素不可能从两个或两个以上的其他应用元素中继承,即使它们是从相同的基础元素中派生出来的;
- 一个子类应用元素不可能定义任何基本应用关系。那些需要的基本应用关系必须在超类应用元素中定义;
- 一个子类可以增加扩展应用关系;
- 一个子类可以增加基本和扩展应用属性;
- 在一个子类应用元素及其超类应用元素之内,应用属性和关系的名称必须唯一;
- 禁止通过子类应用属性使超类应用属性重载。因此,一个子类的应用属性与其超类的一个应用属性不可能拥有相同的名字。此外,超类的应用属性不可能在子类内重命名;
- 只有数据类型与 DT_BLOB 相应的一个属性才有可能包含在一个实例中。因此,超类应用元素或其子类应用元素都可能有一个 blob 类型属性;

——虽然可能不是非常有用，但是，不要求一个子类指定自身的应用属性（除了超类已经赋予的那些应用属性之外）。这个不能视为 **MIME** 的一个备选类型（这个类型应该用来描述一个实例类型的特征）。

注1：超类应用元素不能被声明为抽象；任何应用都可以通过 API 方式创建它们的实例。

注2：在本语境的单一派生（一些基础元素只能派生一个应用元素，见第 5.3.2 节）指一个基础元素只能派生一个超类应用元素。该基础元素可以派生扩展子类应用元素。它们都是继承一个超类应用元素。

ASAM ODS 规范提供了关于通过 RPC-API 和 OO-API 技术，在物理存储、在 ATF/CLA、ATF/XML、还有服务器存取中实施继承机制概念的具体的信息。在这些章节中有专门的小节说明这些信息。

注：在物理存储中，需要基本属性 ‘objecttype’，因此，使用 APIs 时，也需要基本属性 ‘objecttype’。在 ATF 存储和客户端应用中都不需要它。因此，包含继承元素的应用模型可能稍微不同，根据第 6 章，这取决于它们是否只在物理存储环境中使用。

这可能产生一些特定的情况：

——服务器的数据应该写入一个 ATF 文件：由于它保存在服务器中，所以，‘objecttype’ 属性可用，并且可以在模型阶段之内写入 ATF 文件。实例也可能包含 ‘objecttype’ 的一个值。

——在 ATF 文件的数据应该存储在一台服务器中。

在这种情况下，应用模型可能省去 ‘objecttype’。不过，将这类数据导入一台服务器的客户必须提供 ‘objecttype’，并提供几个 ‘objecttype’ 在实例中的值。服务器将会使这些值适应内部已知的正确的 AIDs。

5.7.3 继承和派生

虽然继承和派生听起来似乎它们说明的是同一件事，但是 ASAM ODS 明确对两者作出区分，注意到这一点是重要的。

派生是创建一个应用元素并指定其基本类型时发生的一个过程。从一个基础元素中派生该应用元素。它被赋予该基础元素的类型。

通过派生，应用元素与一个基础元素联系起来。

继承是一个应用元素应继承另外一个应用元素所有的应用属性（并取得其自身的扩展属性）时发生的一个过程。它是从其他应用元素中继承的。

通过继承，一个应用元素与相同应用模型中的另外一个应用元素联系起来。

5.8 ATF 在 ODS 体系架构中的应用

关于 ASAM ODS 中如何使用 ASAM 传输格式（ATF）详见第 9 章 ATF/CLA 和第 10 章 ATF/XML。

5.9 ASAM ODS 的安全概念

ASAM ODS 为在研究、开发及生产环境中的各类应用领域提供了信息存储的方法。在很多情况下，这些信息是保密的，不允许任何人访问。因此，ASAM ODS 为其数据存储提供了信息访问控制机制。它们包含在 APIs（OO-API 和 RPC-API）中，并由 ASAM ODS 服务器实施。

ASAM ODS 的安全由两个概念组成：

——访问 ASAM ODS 服务器和该服务器内的具体信息的一组全局规则；

——基于 ACL 的安全概念，一台服务器内大多数信息的安全管理模式。

5.9.1 节描述了全局规则集合，其它后续各小节说明了基于 ACL 的安全概念。第 5.9.1 节的全局规则优于基于 ACL 的安全概念。

关于如何在物理存储中保存访问控制信息的更多信息，请参见第 6 章。安全对 ASAM 传输格式(ATF)的影响在第 9 章 (ATF/CLA) 和第 10 章 (ATF/XML) 中说明。分别在 API 第 12 章和第 13 章说明如何通过 APIs 存取信息。

5.9.1 存取 ASAM ODS 信息的全球规则

5.9.1.1 用户端登录和服务器的身份验证

只有在已经创建一个会话之后，存取 ASAM ODS 服务器内的信息才向客户端应用开放。将要创建一个会话时，客户通过提交用户名和密码登录。用户名和密码区分大小写。

ASAM ODS 服务器验证信息的有效性。为此，服务器将会存取基础元素 AoUser 派生的应用元素的实例，并确定是否有名字（就是：从基本属性‘名字’派生的属性值）与所提交的用户名一致。

如果相同，服务器存取从基本属性‘密码’派生的应用属性值，并与所提交的密码值进行比较。如果两者匹配，那么，登陆信息就是有效的，随后，为该用户创建一个会话。客户端在成功打开会话期间的进一步请求不再要求提供用户名和密码。

如果所提交的信息无效，将返回一个错误消息，不授权访问 ODS 服务器的内容。

目前，ASAM ODS 不支持外部身份验证服务。

5.9.1.2 超级用户

在一台 ASAM ODS 服务器之内，通过变为用户组会员来组织用户。由客户端应用决定对用户组和会员的定义，然而，必须总是存在一个“超级用户”用户组。应用下列规则：

——“超级用户”用户组由一个从基本属性“superuser_flag”派生的必选应用属性判定。必须刚好只有一个用户组实例拥有此标志；

——系统设置必须配置（在 ODS 外面）“超级用户”用户组；无法通过 ASAM ODS APIs 修改“superuser_flag”；

——只允许该用户组会员增加扩展用户或从“超级用户”用户组删除用户；

——至少有一个用户必须注册该“超级用户”用户组。服务器不允许该组存在的最后一个用户。

一个超级用户（就是：“超级用户”用户组的一个会员）拥有完全存取所有数据的权限。不需要任何 ACL-entries，ASAM ODS 服务器有可能忽略了存在的会员。然而，本节下面段落给出的一些“存取限制”同样适用超级用户；这在那些段落有明确指出。

5.9.1.3 一般存取限制

一台 ASAM ODS 服务器必须限制存储中一些信息的存取。

(1) 存取密码

从基本属性 AoUser 的‘密码’派生的（必选）应用属性不得通过 API 向用户传输。

a) 用户通过任何 API 存取模型时，不得让密码在应用模型中可视，也不得根据请求将其值传输给用户。此限制同样适用“超级用户”用户组。

b) 用户正在登录，并创建一个会话时，该用户可以变更其自己的密码，取得一个新值。其他用户

的密码只能由一个超级用户来改变。

- c) 密码将加密存储，除了它们在第一次初始化时。改变第一次密码后，改变后的密码必须加密存储。

一台 ASAM ODS 服务器必须按如下步骤进行身份验证：首先，它必须验证服务器存储中的条目的空白密码。如果验证失败，它必须验证服务器存储中的条目的加密密码。如果此次验证也失败，登录信息无效，服务器将拒绝创建一个会话。

(2) 存取应用模型

只有一个超级用户可以改变应用元素和应用模型（例如，添加新的应用属性、添加应用元素）。

然而，请注意，如果已经从应用元素中创建了实例，那么将无法删除这些应用元素。

只有一个超级用户可以配置安全等级（元素、属性及实例安全），并设置 ACL 引用。

5.9.1.4 自动程序

对于后台处于批处理模型下工作的 ODS 用户，适用常规的存取规则：用户需要通过提交其用户名和密码登录到所需的服务器，然后，才被授予相应的数据存取权限。

这些用户同样适用产生将要被创建的数据对象的 ACLs 规则。

实施建议：在自动操作中，数据对象通常为某人创建，而不是由某人创建。因此，自动用户将作为有特权的用户在启动时登录 ODS 服务器。由用户实施和系统配置决定向用户提供一旦启动后所需的权限。

5.9.2 基于 ACL 的安全概念概述

图 29 表明了 ASAM ODS 基于 ACL 的安全概念的主要组成部分。

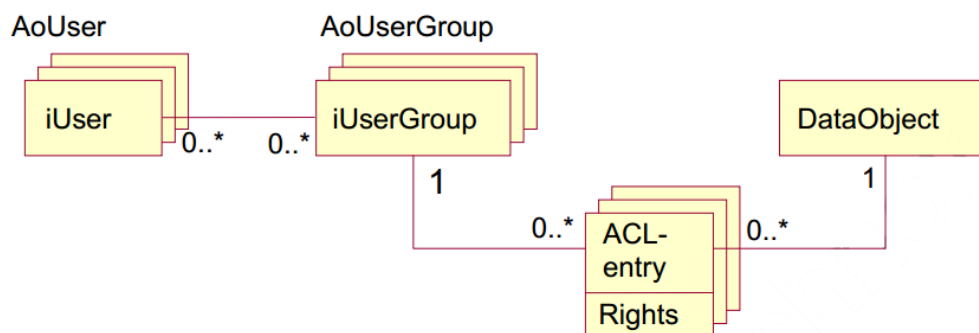


图29 ASAM ODS 基于 ACL 的安全概念

它们在 ASAM ODS 安全的角色是：

(1) iUser

- a) iUser 是基础元素 AoUser 派生的一个应用元素的一个实例。它表示一个单独用户，通过用户名字和密码鉴别。此信息由用户通过打开的会话提交给服务器。在整个打开的会话中，iUser 保持不变。通过 ODS 接口存取数据时，其与 iUserGroup 实例的关系决定了存取权

限。

(2) *iUserGroup*

- a) *iUserGroup* 是基础元素 *AoUserGroup* 派生的一个应用元素的一个实例。它表示共享相同存取权限的一组用户组；
 - b) 一个用户可以属于一个或多个用户组；
 - c) 一个用户存取数据对象的权限由其用户组的会员资格定义；
 - d) 一个用户组是一个抽象名词，从其概念看，该名词并不是指特定的组织单位；
 - e) 该概念不提供任何用户组的扩展结构（比如，没有子用户组）。然而，为具体的，自定义目的，它有可能使用子用户组；
- (3) 根据公司惯例，一个用户组可以假设为不同的含义。例如：“项目”，“部门”，“角色”。
- (4) 根据公司惯例，组织结构可以通过使用用户组的命名惯例反映。例如：“X 部门_工程师”，“Y 项目_操作员”。

(3) *DataObject*

- a) 一个数据对象在本语境中是一个应用元素，一个应用属性或一个实例；
 - b) 例外，基础元素 *AoLocalColumn* 及其应用属性和实例派生的应用元素不能作为存取控制的数据对象。相反，相对于的测量量及它们的实例要用来指定存取权限。这意味着不能为基础元素 *AoLocalColumn* 派生的应用元素设置属性安全；
 - c) 通过使用 *ACL-entries* 来定义用户组对数据对象的存取权限。
- (3) *ACL*（存取控制列表）-*entries*
- a) *ASAM ODS* 储存器内的每个数据对象可以有一个或多个的单独 *ACL-entries*。然而，没有 *ACL-entries* 的也可能是数据对象；
 - b) 数据对象间不共享 *ACL-entries*，即使它们指定了相同的权限；
 - c) 一个 *ACL-entries* 确定了授权给一个特定用户组对一个数据对象的存取权限；
 - d) *ACL-entries* 存储在 *ASAM ODS* 服务器内的列表中。更多关于 *ACL-entries* 如何在物理存储器中存储的信息，见第 6 章。

5.9.3 基于 *ACL* 的安全等级

可以通过本节说明的 *ACL-entries* 保护上述提到的三种类型的数据对象。必须在 *ASAM ODS* 数据存储系统中配置应该保护这些数据对象中的哪些数据对象。这可以为单个应用元素决定，也可以为它们的组合决定。

5.9.3.1 元素安全：保护一个应用元素的所有实例

在这种情况下，一个或多个的 *ACL-entries* 链接到应用元素，根据在这些 *ACL-entries* 指定的存取权限，通过 *ASAM ODS* 服务器保护该应用元素的所有实例。（一般来说，链接到应用元素的 *ACL-entries* 应与存在存取某种应用元素的实例的用户组一样多。）

对于‘元素安全’，一个 *ACL-entries* 由以下项目组成：

- a) 根据存取权限授权存取实例的应用元素规范；
- b) 存取权限规范（见下节）；

c) 存取权限授权用户组规范。

如果为一个应用元素激活元素安全，只允许一个用户组存取应用元素及其实例，如果存在一个相应 ACL-entries 的话。（后一小节给出一些例外）。

根据第 6 章（物理存储），在实施物理存储的情况下，一个应用元素的所有实例存储在一张表格中（每个实例一行，每个应用属性存储在一列），并对应用元素整张表格设置保护（见图 30 中标示）。

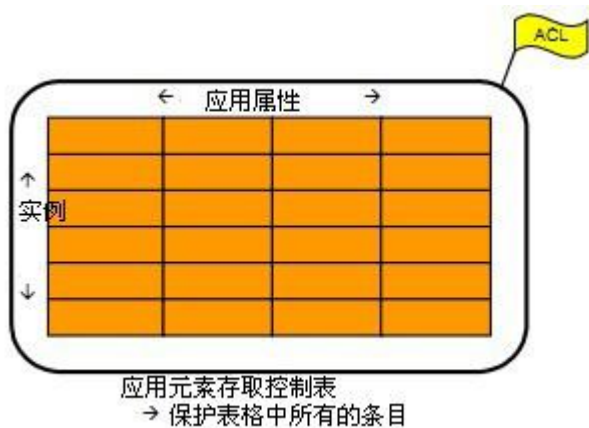


图30 元素安全

示例：元素安全。

假设一个应用元素命名为“aeEngine”，设置了元素安全，一个 ACL-entries 链接到“aeEngine”，{读取 | 更新} 组合存取权限授予一个特定的用户组。然后，该用户组的所有用户被允许读取“eaEngine”的所有实例，修改它们的属性值和关系。

5.9.3.2 实例安全：保护单个实例

在这种情况下，ACL-entries 链接到一个应用元素的单个实例，根据在这些 ACL-entries 指定的存取权限，通过 ASAM ODS 服务器，应用元素的那个实例将受到保护。（一般来说，链接到该实例的 ACL-entries 应该与存在存取该实例的用户组一样多。）

对于‘实例安全’，一个 ACL-entries 由以下项目组成：

- 根据存取权限授权存取实例规范；
- 该实例所属的应用元素规范；
- 存取权限规范（见下节）；
- 存取权限授权用户组规范。

如果为一个应用元素激活实例安全，只允许一个用户组存取该应用元素的任何实例，如果存在一个相应 ACL-entries 的话。

根据第 6 章（物理存储），在实施物理存储的情况下，一个应用元素的所有实例存储在一张表格中（每个实例一行，每个应用属性存储在一列），并对应用元素表格中的一行设置保护（在图 31 中标示）。

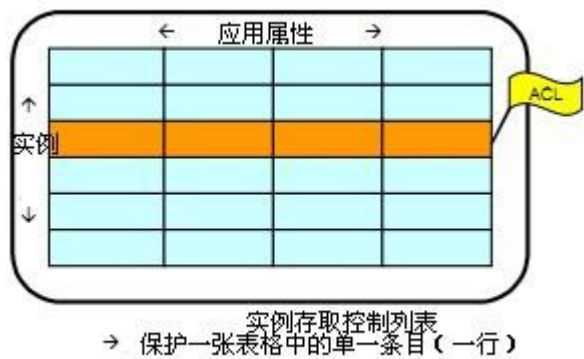


图31 实例安全

示例：实例安全。

假设一个应用元素命名为“aeTest”，它的其中一个实例命名为“iXK4.2_tests”，一个 ACL-entries 链接到“iXK4.2_tests”，{读取|插入}组合存取权限授予一个特定的用户组。然后，该用户组的所有用户被允许读取该单一实例（包括其实例的所有属性），创建其子实例。

5.9.3.3 属性安全：保护一个应用属性或一个应用元素所有实例中的一个 1：N 关系

在这种情况下，ACL-entries 链接到一个应用属性或应用元素的一个 1：n 应用关系，根据，这些在 ACL-entries 中指定的存取权限，只有该应用元素的这个应用属性（各自的关系）受到 ASAM ODS 服务器的保护。然而，该应用元素的所有实例的应用属性（各个关系中）也将受到保护。（一般来说，链接到该应用属性各自关系的 ACL-entries 应该与存在存取该应用属性各自关系的用户组一样多。）

对于‘属性安全’，一个 ACL-entries 由以下项目组成：

- 根据存取权限授权存取应用属性（各个关系中）的规范；
- 该应用属性（各个关系中）所属的应用元素规范；
- 存取权限规范（见下节）；
- 存取权限授权用户组规范。

如果为一个应用元素激活属性安全，只允许一个用户组存取该应用元素的任何应用属性及它们在实例中的值，如果存在一个相应 ACL-entries 的话（后一小节给出一些例外）。

根据第 6 章（物理存储），在实施物理存储的情况下，一个应用元素的所有实例存储在一张表格中（每个实例一行，每个应用属性存储在一列），并对应用元素表格中的一列设置保护（在图 32 中标示）。

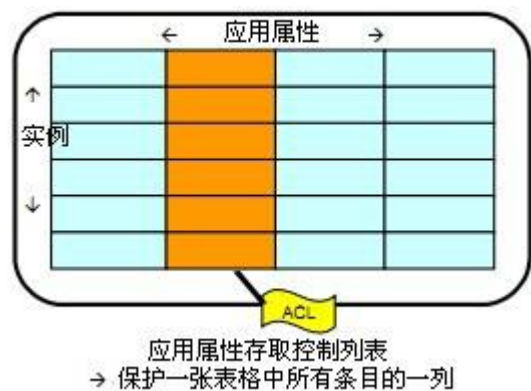


图32 属性安全

示例：属性安全。

假设一个应用元素命名为“aePress Transducer”，它的其中一个应用属性命名为“校准_日期”，一个 ACL-entries 链接到“校准_日期”，{读取}存取权限授予一个特定的用户组。然后，该用户组的所有用户被允许读取“aePress Transducer”的所有实例的该单一属性。

基于 ACL 的安全等级可以为每个应用元素单独设置。更多关于安全等级如何在 ASAM ODS 物理存储中编码的信息，请参加第 6 章（表 SVCENT）。由于它是位编码的，所以，甚至可以指定任何组合。然而，应该意识到，在一个相同的应用元素中集成安全等级可能导致非常复杂的、容易出错的存取计划，为清晰起见，建议不要为一个数据模型中的一个应用元素同时配置安全组合，像元素安全和实例安全组合。

在声明相同应用元素的多个安全等级的情况下，只能读取一个特定实例的一个特定属性的值，如果所声明的每个安全等级的 ACLs 允许的话。

注1：n：m关系不能设置任何属性安全。

注 2：不可能保护一个单一实例的一个单一应用属性的值。要么该实例的所有应用属性受到保护（实例安全），要么同一个应用元素的所有实例的该属性受到保护（属性安全）。

实施建议：要求元素安全（一个应用元素的所有实例的存取保护）控制插入不是分层结构的数据的权限。

5.9.3.4 存取实例属性

基于 ACL 的安全概念授予存取一个实例的所有实例属性的权限，如果也被授予了该实例的相应的存取权限的话。

5.9.3.5 规范安全管理的一般考虑

简单配置，容易管理，而选择性（复杂的）配置满足了选择性要求。因此，必须决定应该实施何种程度的安全。一些问题可能对此过程有帮助：

- 要求必须保护所有的应用元素吗？
- 只有对要求保护的那些元素设置一个安全等级。
- 哪些应用元素需要对不同实例的不同存取计划？

只对真正需要保护单一实例的应用元素设置‘实例安全’。否则，使用‘元素安全’或不设置安全等

级。

——有真正需要特别保护的应用属性吗？

只对真正要求保护的应用属性设置‘属性安全’。

——尝试避免在同一个应用元素中混合不同的安全等级，即使对于有经验的用户，确定哪个用户组可以存取什么实例的哪个属性，也可能变得十分复杂。

简单的安全配置示例：

注 1：应用元素 aeTestSeries（从 AoTest）：‘实例安全’（可以单独保护每个测试系列）；

注 2：应用元素 aeTest（从 AoSubTest）：‘实例安全’（可以单独保护每个测试）；

注 3：应用元素 aeMeasurement（从 AoMeasurement）及下级测量数据：不受保护（测量量的存取权限，及最终从 aeTest 中取得的本地列的存取权限）；

注 4：任何一个应用元素的属性：不允许单一保护；

注 5：其他所有的应用元素：‘元素安全’：存取权限同样适用于每个应用元素的所有实例。

5.9.4 基本存取权限

有五个不同的基本存取权限：读取、更新、插入、删除和授权。

（1）读取

- a) 读取权限与元素安全：用户可以读取该应用元素的所有实例；
- b) 读取权限与实例安全：用户可以读取该实例；
- c) 读取权限与属性安全：用户可以读取该一个应用元素的所有实例的该属性（各个关系中）的值。

（2）更新

- a) 更新权限与元素安全：用户可以更改该应用元素的所有实例；
- b) 更新权限与实例安全：用户可以更改该实例；
- c) 更新权限与属性安全：用户可以更改一个应用元素的所有实例的该属性（各个关系中）的值。

（3）插入

- a) 插入权限与元素安全：用户可以创建该应用元素新的实例；
 - b) 插入权限与实例安全：用户可以创建该（母）实例的子实例；
- 注 1：创建一个子实例意思是，创建一个新的实例或重新分配一个现有的子实例给一个不同的母实例。
- 注 2：一个应用元素在父_子类型的任何关系中不是一个母实例，那么，该应用元素的一个实例的插入权限没有意义，服务器将忽略该插入权限（任何管理工具都不支持）。
- c) 插入权限与属性安全：这没有意义，服务器将忽略该插入权限（任何管理工具都不支持）。

（4）删除

- a) 删除权限与元素安全：用户可以删除该应用元素的任何实例；
- b) 删除权限与实例安全：用户可以删除具体元实例；
- c) 删除权限与属性安全：这没有意义，服务器将忽略该插入权限（任何管理工具都不支持）。

（5）授权

- a) 授权权限与元素、实例或属性安全：存取权限可以传递。

该权限有服务器解释，如下：

——如果一个用户组对一个数据对象有授权权限，那么，该用户组的任何会员可以传递该组对该数据对象的授权权限给其他用户组；

——也可以设置该组对该数据对象的授权权限模板；

——从一个组传递授权权限到另外一个组的成员不必是接收组的会员。

要避免“盲目存储”，没有授权的所有的更新、插入和删除权限的组合中，读取权限被视为无效（管理工具应该不接受这些组合）。在出现这些组合的情况下，服务器将会拒绝访问并返回一个错误消息。

表 9 说明了存取权限组合，及这些组合当中哪些权限是有效的。

表9 存取权限组合

删除	插入	更新	读取	意义：
0	0	0	0	没有存取
0	1	0	0	（无效组合）
1	0	0	0	（无效组合）
1	1	0	0	（无效组合）
0	0	1	0	（无效组合）
0	1	1	0	（无效组合）
1	0	1	0	（无效组合）
1	1	1	0	（无效组合）
0	0	0	1	读取
0	1	0	1	读取和插入（例如，针对测试平台操作员）
1	0	0	1	读取和删除（例如，针对管理员）
1	1	0	1	读取、删除和插入（例如，针对管理员）
0	0	1	1	读取和更新（例如，针对工程师）
0	1	1	1	除了删除的其他存取（例如，针对工程师）
1	0	1	1	除了插入的其他存取 权限
1	1	1	1	所有存取权限

权限以位模式编码，由表 10 指定（0 未模式是最不重要的位模式）。

表10 存取权限的位编码

4位	3位	2位	1位	0
授权	删除	插入	更新	读取

注 1：每个所需的权限需要分别输入。在“权限 A 自动意味着权限 B”这个意义上，没有一个层次结构。

注 2：从一个实例到另一个实例设置一个 1: n 关系，在 n 这边的实例，一个读取权限是必需的，在 1 这边的实例，一个更新-（加上读取）权限是必需的。

注 3：从一个实例到另一个实例设置一个 n: m 关系，在两边的实例中，一个更新-（加上读取）权限都是必需的。

实施建议：避免复杂的、不必要的属性保护，应该在客户端应用拒绝一些特定属性的存取（IDs，

引用), 而不是通过属性保护。

服务器实施说明: 如果某个特定用户组的一些数据对象的权限设置为 0 (没有存取权限), 那么, 服务器应该从 ACL 中删除该 ACL-entries, 因为一个缺省的 ACL-entries 与“没有存取权限”有相同含义。

5.9.5 客户存取现有数据对象

本节说明一台 ODS 服务器确定一个用户 (由于其登录, 他属于一个特定的用户组) 是否可以在 ODS 存储器中存取信息所遵循的规则。

5.9.5.1 一般自由存取

基于 ACL 的安全概念授权自由读取存取权限 (读取权限)。

(1) 存取基础模型, 包括其基础元素, 及基础元素的基本属性和基本关系。此存取只能通过 OO-API 获取。RPC-API 不提供此存取。

(2) (只在通过 RPC-API 存取的情况下), 存取应用模型, 包括其应用元素, 及应用元素的应用属性和应用关系。

在通过 OO-API 存取的情况下, 存取应用元素取决于所定义的存取控制。如果元素安全被声明为是一个应用元素安全等级, 那么, 一个用户只能存取该元素, 如果该用户所属的任何用户组被一个恰当的 ACL-entries 授予读取存取权限的话。该规则的例外是在规则 3 中给出的列表。

建议: 强烈建议用户的应用模型 (取决于该用户的 ACL-entries) 视图产生一个一致的应用模型, 因此:

- 为一个用户组指定 **ACL-entries** 时, 必须注意, 该用户组有存取一个元素的权限时, 至少得有读取-存取该元素的父元素的所有元素的权限。否则, **OO-API** 可能会返回一个似乎不一致的一个应用模型 (因为子应用元素可视, 但不是该应用元素的父元素);
- 如果一个客户应用需要限制一个应用模型的应用元素的可视性, 那么, 该用户应该构建自己的过滤器, 而不是依赖 (因此滥用) 用户当前登录的安全设置隐式给出的过滤器机制。

(3) 存取从下列基础元素中派生的应用元素

存取从下列基础元素中派生的应用元素包括:

- a) AoUser;
- b) AoUserGroup;
- c) AoEnvironment;
- d) AoTest;
- e) AoSubTest;
- f) AoMeasurement;
- g) AoMeasurementQuantity;
- h) AoQuantity;
- i) AoUnit;
- j) AoPhysicalDimension;

- k) AoSubmatrix;
- l) AoLocalColumn;
- m) AoExternalComponent。

该自由访问只与应用模型有关。

(4) 存取从下列基本属性派生的应用属性

存取从下列基本属性派生的应用属性包括：

- a) 标识；
- b) 名字；
- c) 版本；
- d) 对象类型；
- e) 所有基本应用关系。

如果一个应用元素已经设置了属性安全，而不是元素安全或实例安全，那么，自由读取包括读取上述提到的属性在该应用元素的所有实例的值和关系

然而，请注意，由规则 3 和 4 指定的自由读取存取只是授权读取、修改、删除等，必须在相应的 ACL-entries 中定义恰当的权限。

管理工具应该遵守这些规则，应该不允许从上述提到的元素/属性中删除读取-权限。

示例：自由读取-存取权限。

假设一个应用元素命名为“aeUUT”（从 AoUnitUnder Test 中派生），它的一个基本应用属性为“uutName”（从基本属性“名字”中派生），及一个扩展应用属性为“uutAtt”。

给出 aeUUT 的两个实例：

一个实例的属性值 uutName= ‘myName1’ 和 uutAtt= ‘myAtt1’。

一个实例的属性值 uutName= ‘myName2’ 和 uutAtt= ‘myAtt2’。

假设一个应用元素命名为“aeTest”（从 AoTest 中派生），它的一个基本应用属性为“testName”（从基本属性“名字”中派生），及一个扩展应用属性为“testAtt”。

给出 aeTest 的两个实例：

一个实例的属性值 testName= ‘abc1’ 和 testAtt= ‘val1’。

一个实例的属性值 testName= ‘abc2’ 和 testAtt= ‘val2’。

表 11 所示为本示例中包含的各项。

表11 自由读取-存取权限

元素	属性	第 1 个实例值	第 2 个实例值
aeUUT	uutName	myName1	myName2
	uutAtt	myAtt1	myAtt2
aeTest	testName	abc1	abc2
	testAtt	val1	val2

下列情况说明了，一个用户可以在一些选定的安全方案中进行什么存取，如果没有 ACL-entries 授权该用户的用户组任何关于存取这些元素、属性和实例的特定权限的话。

第 1 种情况：两个应用元素已经设置元素安全（只有元素安全等级）

在这种情况下，该用户能够：

- 确定存在 **aeTest**（基于规则 3，因为从 **AoTest** 派生的应用元素不受保护）。
- 发现 **aeTest** 拥有属性“**testName**”和“**testAtt**”（基于规则 3，因为从 **AoTest** 派生的应用元素不受保护）。

注：在这种情况下，不能读取任何实例中的任何属性的值。

因此，该用户可以存取表 12 方框中的项目。

表12 自由读取-存取权限（情况 1）

元素	属性	第 1 个实例值	第 2 个实例值
aeUUT	uutName	myName1	myName2
	uutAtt	myAtt1	myAtt2
aeTest	testName	abc1	abc2
	testAtt	val1	val2

第 2 种情况：两个应用元素已经设置属性安全（只有元素安全等级），见表 13。

在这种情况下，该用户能够：

- 确定存在 **aeTest**（基于规则 3，因为从 **AoTest** 派生的应用元素不受保护）；
- 发现 **aeUUT** 拥有属性“**uutName**”，**aeTest** 拥有属性“**testName**”（基于规则 4，因为从基本属性“名字”派生的应用属性不受保护）；

——读取 **aeUUT** 的两个实例中的应用属性 **uutName** 的值和 **aeTest** 两个实例中应用属性 **testName** 的值（基于规则 4，因为从基本属性“名字”派生的应用属性不受保护）。

表13 自由读取-存取权限（情况 2）

元素	属性	第 1 个实例值	第 2 个实例值
aeUUT	uutName	myName1	myName2
	uutAtt	myAtt1	myAtt2
aeTest	testName	abc1	abc2
	testAtt	val1	val2

第 3 种情况：两个应用元素已经设置实例安全（只有实例安全等级），见表 16。

在这种情况下，该用户能够

- 确定存在 **aeUUT** 和 **aeTest**（由于实例安全整体上不保护应用元素）；
- 发现 **aeUUT** 拥有属性“**uutName**”和“**uutAtt**”，**aeTest** 拥有属性“**testName**”，“**testAtt**”（由于实例安全整体上不保护应用元素）。

注：在这种情况下，不能读取任何实例中的任何属性的值，见表14。

表14 自由读取-存取权限（情况 3）

元素	属性	第 1 个实例值	第 2 个实例值

aeUUT	uutName	myName1	myName2
	uutAtt	myAtt1	myAtt2
aeTest	testName	abc1	abc2
	testAtt	val1	val2

在设置了多个安全等级（组合级别）的情况下，只能存取相应的交结点。

第 4 种情况：两个应用元素已经设置了属性安全和实例安全等级（但没有元素安全等级），用户能够：

——确定存在 aeUUT 和 aeTest（由于实例安全整体上不保护应用元素）；

——发现 aeUUT 拥有属性“uutName”，aeTest 拥有属性“testName”（基于规则 4，因为从基本属性“名字”派生的应用属性不受保护）。

注：在这种情况下，不能读取任何实例中的任何属性的值，除了在规则4中列出的那些属性之外，用户无法看到其他属性。具体见表15。

表15 自由读取-存取权限（情况 4）

元素	属性	第 1 个实例值	第 2 个实例值
aeUUT	uutName	myName1	myName2
	uutAtt	myAtt1	myAtt2
aeTest	testName	abc1	abc2
	testAtt	val1	val2

5.9.5.2 存取控制模式

如果一个用户打算存取一个数据对象，而又没有授权的一般自由存取，根据上一段的规则，下列存取控制模式可能。这取决于在相应的应用元素，还有该元素的父元素中设置的安全等级。

(1) 直接存取控制

如果在应用元素设置了任何安全等级，那么，该数据对象属于直接存取控制。通过下列验证，确定存取权限。所有的验证必须成功：

a) 元素安全等级验证

如果在应用元素设置元素安全等级，应用元素，该应用元素的所有实例就受控于与应用元素链接的 ACL-entries。一个用户组将只能确定存在一个应用元素，也仅仅可以存取该应用元素的实例（包括这些实例的属性的值）。如果存在一个相应的 ACL-entries 为该应用元素提供恰当的权限的话。

b) 验证属性安全

如果在一个应用元素设置属性安全等级，该应用元素的所有属性就受控于与这些属性链接的 ACL-entries。一个用户组将只能确定存在一个应用属性，也仅仅可以存取该/一个应用属性的值（在该应用元素的所有实例中的属性的值），如果存在一个相应的 ACL-entries 为该应用属性提供恰当的权限的话。

c) 实例安全验证

如果在一个应用元素设置实例安全，该应用元素的所有实例就受控于与这些实例链接的 ACL-entries。一个用户组将只能确定存在一个实例，并能够存取该实例的属性的值，如果攒竹一个相应的 ACL-entries 为该实例提供恰当的权限的话。

(2) 间接存取控制

如果没有在一个应用元素设置安全等级，那么，间接存取控制可能生效。间接存取控制仅仅对存取实例有影响。

在间接存取控制中，父子关系是重要的。父子关系完全在 ASAM ODS 基础模型中定义，不能通过一个应用模型增加任何扩展父子关系。

对父元素及其先辈进行递归地验证，直到找到一个安全等级被激活的一个应用元素为止。任何元素安全或实例安全的 ACL-entries 都从该先辈中派生，并用于派生物。

注：属性安全不能从一个先辈元素转移给一个派生物，因为它们没有共同属性。

(3) 存取不受控制

如果没有对一个应用元素给定一个安全等级（没有元素、属性和实例安全），那么，它不受间接存取控制，所有用户组都被授权存取所有属性及这些属性在该应用元素所有实例中的值。

5.9.6 模板

应该创建一个应用元素的一个实例时，要对它的存取权限进行定义。在实例安全等级的情况下，初始并没有可用的 ACL-entries 对新创建的实例进行控制。为此目的，ASAM ODS 提供了 ACL-模板机制。该机制扩展了上述说明的安全概念，如图 33 所示。

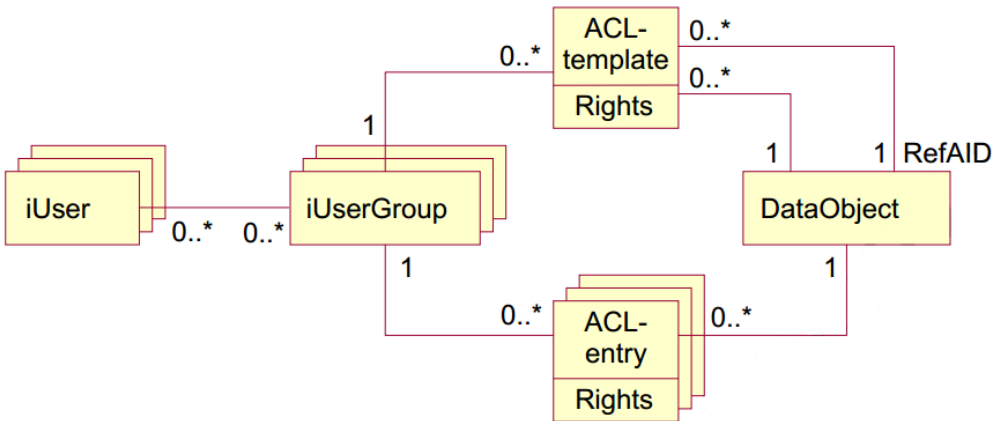


图33 基于 ACL 安全概念里的模板

ACL 模板包括：

——一个 ACL 模板被用来定义 ASAM ODS 服务器不提供但在未来创建的实例的存取权限。它是一个 ACL 条目的一个模型，只能在新创建的实例所属的应用元素已经说明了实例安全的情况下使用；

——一个 ACL 模板允许指定一个特定的应用元素的一个新创建的实例（引用应用元素，RefAID）应该为一个特定的用户组接收一组存取权限。这些权限当前链接一个已经存在的实例或应用元素（被引用的数据对象）；

——一个 ACL 模板存储在一个应用元素中或一个现有的实例中。

一个 ACL 模板由下列项目组成：

——应用元素规范（‘引用应用元素’，RefAID），该应用元素的（新创建）实例应该接收指定的存取权限；

——如果 RefAID=0，那么，该模板将被所有的引用应用元素用来创建新实例。因此，0 是一种通配

符，匹配所有的应用元素。

以下规范：

——应用元素；

——（可选）实例，在 ACL 模板应该存储在一个实例中而不是一个应用元素中的情况下。

ACL 模板存储的位置为（被引用的数据对象）。

RefAID 新创建的实例的存取权限规范（详见上一节）。

应该接收存取权限的用户组规范。

ACL 模板同样存储在 ACL 条目中，不同的是，对于 ACL 模板，它存储在说明实例，而 ACL 条目，它是将要受到 ACL 条目保护的 ACL 条目。更多关于 ACL 条目和 ACL 模板如何在物理存储器中存储的信息，请见第 10 章。

在一个 ASAM ODS 存储器中，可能存在大量 ACL 模板。如果应该创建一个应用元素的一个新实例，且为该应用元素指定了实例安全，那么，ASAM ODS 服务器必须确定使用 ACL 模板创建该实例（或甚至决定不使用他们中的任何一个）。

ACL 模板可能存储在应用元素或现有的实例中。为将要创建的新实例创建 ACL 条目时，它们可能被用来作为一个的模板。可以在两个类型的位置找到将要创建的新实例的 ACL 模板：

——可以在另外一个（已经存在的）实例中找到一个 **ACL** 模板，该实例是将要创建的新实例通过一个应用关系引用的实例。在这种情况下，要求标记该应用关系为 **ACL** 引用。更多关于如何将一个应用关系标记为 **ACL** 引用的信息，请见第 10 章；

——可以在将要创建的新实例所属的应用元素中找到一个 **ACL** 模板。

对于找到的每个 ACL 模板，ODS 服务器必须确定该模板是否可以用于将要创建的新实例。如果模板的 RefAID 与将要创建的新实例所属的应用元素相匹配，情况就如此。如果已经找到匹配的 ACL-模板，将从每个匹配的 ACL 模板中复制用户组及其权限到一个新的 ACL 条目，然后，链接该 ACL 条目到新的实例。

ODS 服务器说明以怎样的序列来搜索这些位置以寻找 ACL 模板的规则，及如何创建一组用于新实例的 ACL 条目的规则，将在下一节说明。

只有超级用户组会员才能创建和存储模板。

5.9.7 确定用于新数据对象的 ACL 条目

在大多数情况下，创建数据对象不要求特定测量时，在设置了实例安全等级的情况下，创建新实例有几个规则。

在很多情况下，需要自动设置一个新实例的存取权限，以满足不同终端用户和不同应用领域的不同要求。本节余下部分说明了处理这种情况的规则，并引进这些规则，满足所有这些要求。

在相应的应用元素设置了实例安全时，确定新创建的实例的存取权限（通过创建 ACL-entries），ASAM ODS 服务器应用了下列说明的规则，序列正好按照下文的说明序列。凡有一个准则匹配时，将应用相应的规则，而不评价任何其他规则。

（1）规则 1

如果写入端用户在插入实例之前明确指定一个或多个会话-具体模板，那么，在任何情况下，服务器都将会使用这些供应的会话-具体模板。

会话-具体模板是前面说明的模板,除了那些临时会话模板及没用任何一个被引用数据对象的模板。

(2) 规则 2

如果新实例所属的应用元素有被标记为 ACL-引用的关系,那么,ASAM ODS 服务器将找到该适用规则。

在这种情况下,服务器将跟踪该新实例的相应关系,并调查被引用的实例。凡是在这样一个被引用的实例中找到的一个 ACL 模板,并且,该 ACL 模板包含了一个与将要创建的新实例的应用元素相互匹配的 RefAID,那么,将会复制 ACL-模板中的权限和用户组到一个新的 ACL 条目,然后,链接该 ACL-entries 到新实例。

如果在被引用的实例中找到一个以上的带有匹配 RefAID 的 ACL 模板,那么,为这些 ACL-模板中的每个 ACL 模板创建一个 ACL 条目。

如果有一个以上的应用关系型被标记为 ACL-引用,那么,将对该应用关系进行调查,以调查所有这些应用关系的被引用的实例,不管这些应用关系是否提供匹配的 ACL-模板,如果提供,将会创建相应的 ACL-entries。

(3) 规则 3

如果至少一个 ACL 模板链接新实例所属的应用元素,那么,ASAM ODS 服务器会找到该适用规则。

如果匹配 ACL 模板在它们之中(RefAID 匹配应用元素本身),那么,ODS 服务器从这些匹配的 ACL-模板复杂权限和用户组到一个相应量的 ACL 条目,并链接这些 ACL 条目到新实例中。

(4) 规则 4

如果新创建的实例有一个父实例,并且,该父实例的应用元素已经设置了实例安全等级,那么,ASAM ODS 服务器将会找到该适用规则。

ODS 服务器将复制父实例中任何可用的 ACL-entries 到新 ACL-entries,并链接这些新 ACL-entries 到新(子)实例中。

(5) 规则 5

在上面规则都不适用的情况下,没有任何 ACL-entries 链接将要创建的新实例(不授予任何存取权限)。稍后,超级用户可增加 ACL-entries。

一般说明:一个 ASAM ODS 数据存储器的基于 ACL 的安全概念可能会变得非常复杂,强烈建议精心设计。在被引用的实例和/或应用元素中创建有用的 ACL-模板时也应该设置 ACL-引用。

下面几页枚举了几个例子,说明如何使用 ACL-模板自动创建新创建实例的 ACL-entries(在设置了实例安全等级的情况下)。

示例:针对规则 2

假设一个用户组有两个实例 i1UG 和 i2UG。Paul Simon 只是 i1UG 的会员, Ray Charles 只是 i2UG 的会员。

假设有两个应用元素:“aeTest”从 AoSubTest 派生,“aeMeasurement”从 AoMeasurement 派生,两个应用元素存取级别指定为‘实例安全等级’。

aeMeasurement 有一个应用关系“arTest”,它从基本关系“test”中派生的,该关系已经被指定为 ACL-引用(ACL REF)。

aeTest 已经存在一个实例 i1Test,该实例拥有(除了 ACL-entries 之外)一个 ACL-模板:RefAID 是 aeMeasurement,用户组是 i1UG,权限是{读取 | 更新 | 授权}。

aeTest的第二个实例i2Test也存在,该实例拥有(除了ACL-entries之外)一个ACL-模板:RefAID是aeMeasurement,用户组是i2UG,权限是{读取}。

图 34 说明了这种情况:

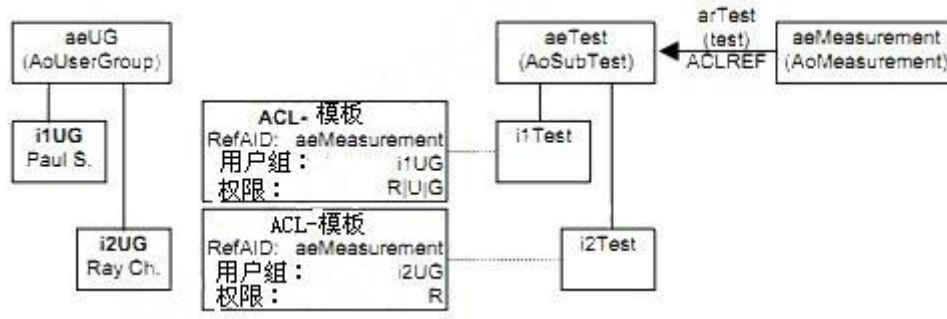


图34 规则 2 示例

现在,应该在 ASAM ODS 服务器中插入 aeMeasurement 的一个新实例 i1M; 用户不用明确指定一个会话特定模板 (不适用规则 1), 但, 设置为 i1Test 的一个关系 arTest 的值。

在这种情况下, ODS 服务器将确定 arTest 为一个 ACL-引用关系 (规则 2: 适用), 并跟踪 arTest, 在 i1Test 停止跟踪。服务器在那儿找到一个 ACL-模板, 模板中, RefAID 是 aeMeasurement, 因此, 服务器复制 i1UG 作为用户组、{读取 | 更新 | 授权} 作为权限到 i1M 的一个 ACL-entries 中。

从该观点出发, Paul Simon 可以读取和修改 i1M, Ray Charles 不可以。

现在,应该在 ASAM ODS 服务器中插入 aeMeasurement 的另一个新实例 i2M; 用户不用明确指定一个会话特定模板, 但, 设置为 i2Test 的一个关系 arTest 的值。

在这种情况下, ODS 服务器将再次确定 arTest 为一个 ACL-引用关系, 并跟踪 arTest, 在 i2Test 停止跟踪。服务器在那儿找到一个匹配 ACL-模板 (模板中, RefAID 是 aeMeasurement), 因此, 服务器复制 i2UG 作为用户组、{读取} 作为权限到 i2M 的一个 ACL-entries 中。

从该观点出发, Ray Charles 可以读取 i2M, Paul Simon 不可以。

示例: 针对规则 3

假设一个用户组有两个实例 i1UG 和 i2UG。Paul Simon 只是 i1UG 的会员, Ray Charles 只是 i2UG 的会员。

假设有两个应用元素: “aeTest” 从 AoSubTest 派生, “aeMeasurement” 从 AoMeasurement 派生, 两个应用元素存取级别指定为 ‘实例安全等级’。

aeMeasurement 的任何应用关系都不被指定为 ACL-引用 (ACL REF)。然而, aeMeasurement 拥有一个 ACL-模板, 模板中, RefAID 是 aeMeasurement, 用户组是 i2UG, 权限是 {读取 | 更新 | 授权}。

图 35 说明了这种情况。

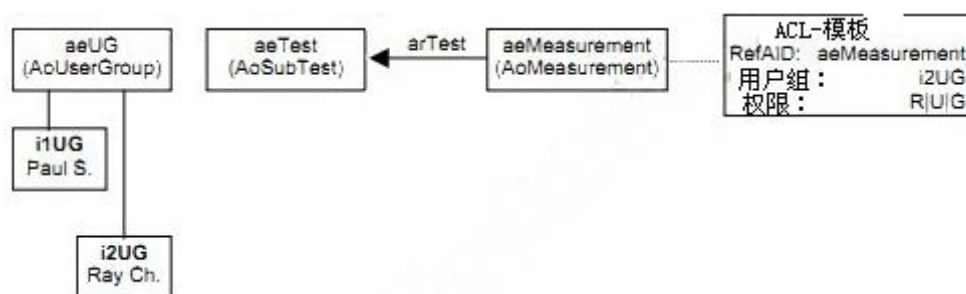


图35 规则 3 示例

现在，应该在 ASAM ODS 服务器中插入 aeMeasurement 的一个新实例 i1M；用户不用明确指定一个会话特定模板（不适用规则 1）。

此外，ODS 服务器将确定 i1M 是 aeMeasurement 的一个实例，aeMeasurement 没有任何关系被标记为 ACL-引用关系（规则 2：不适用）。最后，服务器找到一个 ACL-模板，模板中，RefAID 是 aeMeasurement，该模板直接链接到应用元素 aeMeasurement（规则 3：适用）。因此，服务器复制 i2UG 作为用户组、{读取 | 更新 | 授权}作为权限到 i1M 的一个 ACL-entries 中，并链接该 ACL-entries 到 i1M。

从该观点出发，Ray Charles 可以读取和修改 i1M，Paul Simon 不可以。

对于在父子关系中没有分层组织的应用元素，连接一个 ACL-模板到该应用元素，是有用的。因此，适用规则 3 而不是规则 4。对于在父子关系中分层组织的结构，规则 4 是非常适用的，如下例所示。

示例：针对规则 4

假设一个用户组有两个实例 i1UG 和 i2UG。Paul Simon 只是 i1UG 的会员，Ray Charles 只是 i2UG 的会员。

假设有两个应用元素：“aeTest”从 AoSubTest 派生，“aeMeasurement”从 AoMeasurement 派生，两个应用元素存取级别指定为‘实例安全等级’。

在其他应用关系中，aeMeasurement 有一个应用关系“arTest”，它从基本关系“test”中派生的，然而，其关系没有一个关系已经被指定为 ACL-引用。此外，aeMeasurement 不拥有 ACL-模板。

aeTest 已经存在一个实例 i1Test，该实例拥有一个 ACL-entries，用户组是 i2UG，权限是{读取 | 更新 | 授权}。

图 36 说明了这种情况：

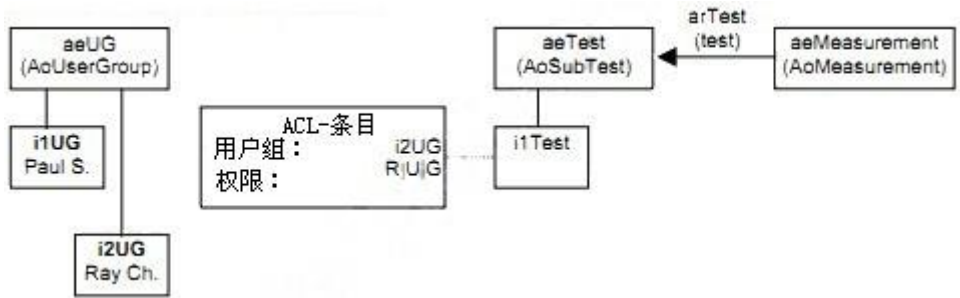


图36 规则 4 示例

现在，应该在 ASAM ODS 服务器中插入 aeMeasurement 的一个新实例 i1M；用户不用明确指定一个会话特定模板（不适用规则 1），但，设置为 i1Test 的一个关系 arTest 的值。

在这种情况下，ODS 服务器将确定 aeMeasurement 没有任何关系被标记为 ACL-引用关系（规则 2：不适用），i1M 所属的应用元素没有任何 ACL-模板（规则 3：不适用）。此外，服务器确定 arTest 从基本关系“test”派生，该基本关系是一个父子关系，服务器视 i1Test 为 i1M 的父实例。服务器找到一个链接 i1Test 的 ACL-entries（规则 4：适用）。因此，ODS 服务器复制该 ACL-entries 到一个新的 ACL-entries 中，并链接该新 ACL-entries 到 i1M。

从该观点出发，Ray Charles 可以读取和修改 i1M，Paul Simon 不可以。

5.9.8 继承概念安全

一些特定规则适用于实施继承机制概念的应用元素，及这些应用元素的实例。

(1) 安全等级

超类的安全等级必须与它所有子类的安全等级一致。因此，凡通过 API 方式为一个超类设置安全等级时，ODS 服务器复制该超类的值到所有的子类；凡通过 API 方式为一个子类设置安全等级时，ODS 服务器复制该子类的值到相应的超类及所有的其他子类。

建议只为超类应用元素设置安全等级。

(2) ACL 条目

ACL 条目可以链接到应用元素、应用属性和实例。

如果（在‘元素安全等级’的情况下）一个 ACL-entries 链接到一个子类应用元素，那么，ODS 服务器将会自动拷贝该 ACL-entries，并链接到相应的超类应用元素。这将提高查询需要评价的超类应用元素的性能。

与超类应用元素链接的 ACL-entries 不会自动被拷贝到子类应用元素。

如果（在‘实例安全等级’的情况下）一个 ACL-entries 链接到一个子类应用元素的一个实例，那么，ODS 服务器将会自动拷贝该 ACL-entries，并链接到相应的超类应用元素的实例。这将确保了共同属性值（那些由超类定义的共同属性值）在设置相同的权限下也可以存取。

与超类应用元素的实例链接的 ACL-entries 不会自动被拷贝到子类应用元素的实例。

因此，存在一些用户组，这些用户组被允许存取子类实例的共同的（超类）但不特定的（子类）的属性值。

在设置了‘属性安全等级’的情况下，对 ACL-entries 没有特定的测量，因为它们链接到一个相应的应用属性，该应用属性可以在超类定义，也可以在子类定义。

(3) ACL 模板

一个 ACL 模板只可以指定一个超类应用元素作为 RefAID（引用应用元素）。

凡将要创建一个超类或一个子类的一个新例子时，对于遵循 ACL-引用关系（参见第 5.9.7 节规则 2）而找到的实例，将审查它们的 RefAID 是否匹配超类应用元素。

如果将要创建一个超类或一个子类的一个新例子，服务器在子类应用元素中寻找 ACL-模板（参见第 5.9.7 节规则 3）适用同样的规则：如果 ACL-模板的 RefAID 匹配相对应的子类应用元素，就说这个 ACL-模板是匹配的。

5.9.9 基于 ACL 的安全概念备注

(1) 拥有者

ASAM ODS 没有给出关于一个数据对象拥有者的任何特定概念，这被认为超出了本标准的范围。应用模型可以引进它们自己的拥有者的概念。

(2) 有/没有存取控制的系统

一个应用模型可以创建存取控制，也可以不创建存取控制。

如果一台服务器组织一个应用模型并为该应用模型定义存取控制（就是：至少有一个应用模型有一个不等于 0 的安全等级），那么，该服务器将在该应用模型内采用相关的存取保护规则。

(3) 创建/修改应用模型时的安全管理建议

a) 创建了一个新的应用元素时，应该设置其安全等级（元素、属性、实例安全等级）。

由于在那一刻还没有实例，所以，不必要为实例设置 **ACL-entries**。

应用元素和/或应用属性的 **ACL-entries** 必须在创建该应用元素的过程中创建，如果已经声明了一个相应的安全等级。

在设置了实例安全等级的情况下，可以在该点上决定，对于将来创建的该应用元素的实例，是否应该通过 **ACL-模板** 机制自动给出它们的 **ACL**。

如果是，那么，可以指定 **ACL-模板**，并将它们链接到新的应用元素。

此外，还可以标记一个或多个的应用关系为 **ACL-引用**，并应该指定一组 **ACL-模板**，并链接到在未来将会变为该被标记的关系的关系端点的实例。

b) 创建一个新应用属性或一个 **1:n** 应用关系时，它的表现必须符合其应用元素的安全水平。

如果设置了元素安全等级或实例安全等级，那么，该应用属性（各个关系中）在多个单一实例中的值将自动受控于其应用元素各个相应实例的 **ACL-模板**。不要求任何特定测量。

如果设置了属性安全等级，必须在属性（各个关系中）创建过程中，为该应用属性（各个关系中）创建一组新的 **ACL-entries**，以允许未来可以在实例中存取该应用属性（各个关系中）的值。

c) 创建一个新的实例，而安全等级没有设置到实例安全等级时，不要求任何特定测量，而且存取新实例的权限与该应用元素的那些已经存在的实例相同：

1) 如果设置了元素安全等级，那么将采用其应用元素的 **ACL-entries**；

2) 如果设置了属性安全等级，那么与其应用属性的 **ACL-entries** 相关。

d) 此外，如果新创建了一个 **n:m** 应用关系或一个实例属性，那么，不需要做关于 **ACL-entries** 方面的任何工作。因为 **n:m** 应用关系和实例属性不受保护，并且可以自由存取，只要可以存取相应的实例。

5.10 其它因素

本节提供了一组有关使用 ASAM ODS 实现数据存储和恢复的因素、提议及建议。

5.10.1 绝对和相对日期与时间

ASAM ODS 数据类型 T-DATE 不提供关于全球绝对时间的信息。

然而，在全球企业，该信息是必要的。这可以用基础元素 AoEnvironment 里的基本属性‘时区’派生的应用属性的值来创建。该属性值最终确定了整个 ASAM ODS 的 UTC（世界协调时间）环境的时区。在对 ASAM ODS 服务器进行读取和写入访问时都必须考虑该属性值。

如果全球绝对时间信息是重要的，那么，赋予一个数据类型 T-DATE 的所有信息必须与‘时区’相关。在这种情况下，如果用户与 ODS 环境不在同一个时区，必须：

——在提交给用户之前，重新计算当地时区从 ODS 服务器读取的所有的日期/时间相关信息；

——在提交给 ODS 服务器之前，重新计算在 ODS 服务器环境的时区，需要从当地时区写入服务器的所有的日期/时间相关信息。

应该注意，根据其‘时区’设置，ODS 服务器不提供设施来调整时间信息。用户负责服务器时间、世界时间、及当地时间之间的转换。

关于基本属性‘时区’的工作，必须小心。

不管是谁在 ASAM ODS 服务器中初始设置一个新的环境,都应该设置一个合理的‘时区’属性的值。这可以是物理服务器位置的时区,或是公司的主要网站的时区。

不管是谁打算改变环境的‘时区’属性的值,都必须保证对一个数据类型 T_DATE 提供的信息在整个完整的 ODS 服务器存储器中作相应的调整。由于这可能是一个耗时的任务,所以,不建议改变一个现有的 ASAM ODS 数据存储的‘时区’值。

一台 ASAM ODS 服务器内的相对时间通常与一个相关的相应单位(例如“秒”,“小时”,“微秒”)一起存储。该信息独立于任何时区,不必要在用户和服务器之间调整。

存储在本地列的大量数据可能通过相当方便和节省空间的方式与绝对时间相关。不是为存储绝对时间戳定义数据类型 T_DATE 的一个测量量(相关本地列),而是,一个应用模型可能考虑在数值数据类型(例如, T_LONG)中存储时间戳信息,并从相应的测量量实例引用一个基于时间的单位(例如,毫米)。在这种情况下,通过将它们值添加到从相应的测量实例的基本属性‘measurement_begin’派生的应用属性的值,时间戳(在本例子中,时间戳的意思为‘毫秒的数倍’)的本地列的所有值可能转化为绝对时间戳(通过使用‘时区’值也有可能弥补当地用户与 ODS 环境时间之间的差别)。T_DATE 是一种字符串类型,并要求非常大的存储空间,而数值类型更短,更易于处理。

然而,应该注意,一次测量内的所有本地列指‘measurement_begin’的相同的值。

此外,还应该注意,这种时间戳列可能出现负值。

5.10.2 参数和参数集概念

ASAM-ODS 已经定义了存储任何类型的测量数据的几个规则和可能性(例如, MIME 类型, 隐式通道, 独立通道, …)。

特别是“正常”数据,如 X-Y-值的方式作出了非常明确的定义。因此,很有可能加载和理解之前另一个兼容产品写入的每个 ASAM ODS 兼容客户的数据。

但是,还有大量其他数据,如,来源于分析(如统计)的数据或其他算法(FFT, 分类, …)。ASAM ODS 已经实现存取任何数据的目标,但是,“理解”由另一个用户写入的数据的要求是什么呢?这些数据通常必须由其他信息和关于语义、参数和特定属性的隐性知识来说明和处理,

ASAM ODS 用一种方式定义,有趣的公司如何能够处理这些数据。通过使用符合 ODS 规则和可能性的方法,它们必须能够在 ASAM ODS 处理这些类型的数据,而没有出现任何矛盾。

用户的要求是明确的:用独立于产品的方式存储的应该不仅仅是 X-Y-数据。分析数据的任何结果也应该不仅仅是可存取的(ASAM ODS 已经完成这类数据的存储),还应该能够让了解该数据类型的每个产品容易理解。

示例: (雨流矩阵)。

假设在一个 ODS 数据库中有一些测试数据及测量。测量数据是一些测量通道的时间序列(例如,汽车行业的耐久性测试)。

特别是在该环境中,通过计算一个非常特别的统计矩阵,所谓的“雨流矩阵”,减少数据量,这是很正常的。

市场上有几种支持该算法和数据的产品,但是,每种产品格式/方式不同。只有将雨流矩阵存储在一个 ODS 数据库内才能产生可存取性,但不一定产生正确的语义解释。只有通过一个基于 ODS 定义的配套标准,才能实现正确的语义解释。

因此，通过任何参数，ASAM ODS 都必须支持一种处理该配套标准并说明任何一种类型的数据的统一的方法，包括语义。

为此，ASAM ODS 支持基础元素 “AoParameter” 和 “AoParameterSet”。

5.10.2.1 AoParameter 的说明

AoParameter 存储任何 ODS-兼容类型的一个参数。它有一个名称（参数的名称），一个值（任何类型的）。基本属性不是标准的 ODS 基本属性。

- 1) **pvalue**: 实例的值，以字符串表示形式存储；
- 2) **parameter_datatype**: p 值的数据类型，作为数据类型枚举的可能值中的枚举值，用来正确解释 p 值。

5.10.2.2 AoParameterSet 的说明

AoParameterSet 是一个参数的“分组对象”。它存储引用它的 AoParameter 的集合。基本属性是标准的 ODS 基本属性。

5.10.2.3 基础模型中的关系

AoParameterSet 有以下关系：

任何量的 **AoParameter**（因此，它知道在其中的哪些 **AoParameter** 被分组）。

AoParameter 有以下关系：

- 刚好一个 **AoParameterSet**（它所属的那个 **AoParameterSet**）；
- 提供一种方法计算为 **AoParameter** 存储的值信息的物理值的一个 **AoUnit**。

5.10.2.4 Parameters 在应用模型中的用法

Parameters 被认为要链接到任何其他元素，因此，更能正确地说明它们的特征。然而，基础模型不包含一个可以用于该链接的基本关系。（如果包含，人们就可以找到几乎所有的其他基础元素的关系。在此方面，它的表现与 AoAny 类似。）

相反，一个应用模型通常会构建 AoParameter 类型的应用元素，并增加它们的应用关系，这是根据它们携带的信息进行的，这些信息对其他应用元素是重要的。

有可能有：

- 一个 **AoParameterSet** 类型的应用元素（被一个 **AoParameterSet** 类型的应用元素集合引用）；
- 或一个 **AoParameter** 类型的应用元素。

应用任何其他应用元素。

图 37 说明了这种情况，在该种情况下，AoParameter 类型的一个应用元素引用另外一个 ODS 应用元素。这通常用于少量参数。

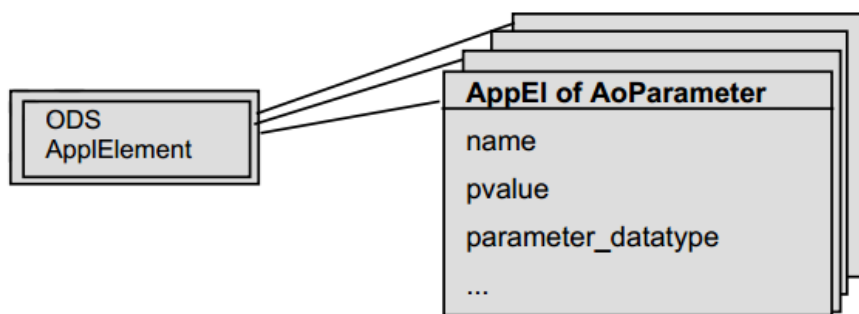


图37 AoParameter 引用一个应用元素

图 38 说明了这种情况，在该种情况下，AoParameterSet 类型的一个应用元素引用另外一个 ODS 应用元素，而 AoParameter 类型的一个应用元素集合引用它。

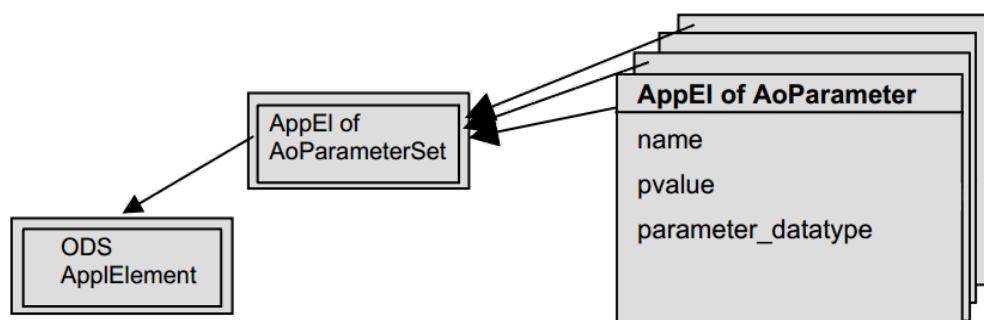


图38 AoParameter 通过 AoParameterSet 引用一个应用元素

5.10.2.5 与替代方法的比较

原则上有三种替代方法可以添加特定信息到应用元素：

1) 可能需要添加携带了所需的其他信息的应用属性。

在这种情况下，属性名称与它们的数据类型必须在定义应用模型的时候指定。

另外，每个实例必须携带该属性的值（甚至在相同的属性只与几个实例相关的情况下）。

此外，在必须添加进一步的信息的情况下，需要改变应用模型。这个在 ASAM ODS 参数方法里不需要改变。

2) 可能一点也不需要在应用模型内添加任何其他信息，需要通过实例属性的方法，添加其他信息到需要信息的所有这些实例。

每个需要信息的实例必须携带该属性的值（甚至在相同的信息只与几个实例相关的情况下）。

3) 最后，需要选择对 AoAny 类型的应用元素进行定义，并链接它们到那些需要进一步说明性信息的元素。这个是最不具体的方法。它不提供一个统一的方法进行分组，也不提供携带了属性值和数据类型的、链接到一个单位的属性。必须在定义应用模型的时候添加它们。

通过支持/使用基础元素 AoParameter 和 AoParameterSet，ASAM ODS 可以集成几乎每个结果类型，包括用灵活但统一的方法说明参数。

示例：雨流数据。

表 16 中参数是解释雨流分类所必需的。它们被定义为一个雨流分类类型的 AoParameter 类型的应用元素的实例。

所有这些实例必须与 AoParameterSet 类型的一个应用元素的一个实例相关

如果信息是必选的（M）或可选的（O），那么，列、“标志”显示，无论如何，都要保留名称。

表16 雨流数据

名称	标志	范围/类型	例外	评注
类别 最小值	M	-00...+00 浮动	最小值>最大值 →错误极端	分类范围的最小值
类别 最大值	M	-00...+00 浮动	最小值>最大值 →错误极端	分类范围的最小值
NClasses	M	1...100 短整型	<1 → 无 效 的 nClasses	类别数目， 代替 BinSize； $Nclasses = (ClassMaximum - ClassMinimum) / BinSize$
BinSize	O	-00...+00 浮动	binSize> (Max-Min) → 无 效 的 BinSize	类别数目， 代替 Nclasses； $BinSize = (ClassMaximum - ClassMinimum) / Nclasses$
振幅抑制	M	1.0... nClasses 浮动	AS<binSize →无效的振幅抑 制	BinSize (1.0=1 BinSize) 中过滤器的范围 默认值=1.0
对称的	O	真的，假的 布尔	-	如果设置，它是一个对称的矩阵 默认，如果不设置=真的

5. 10. 3 量和单位的用法

无论何时存储取值，它们的相应的单位也需要保存。否则，它们无法被正确解释。ASAM ODS 已经建立了处理这种信息的一个概念。该概念在本节说明，而涉及到基础元素的细节可见第 7 章。

五个 ASAM ODS 基础元素被用来完整地，有效地存储单位信息到任何给定的值：

- a) AoQuantity;
- b) AoUnit;
- c) AoPhysicalDimension;
- d) AoQuantituGroup;
- e) AoUnitGroup。

图 39 说明了它们如何相互联系，如何与其他 ASAM ODS 基础元素联系。

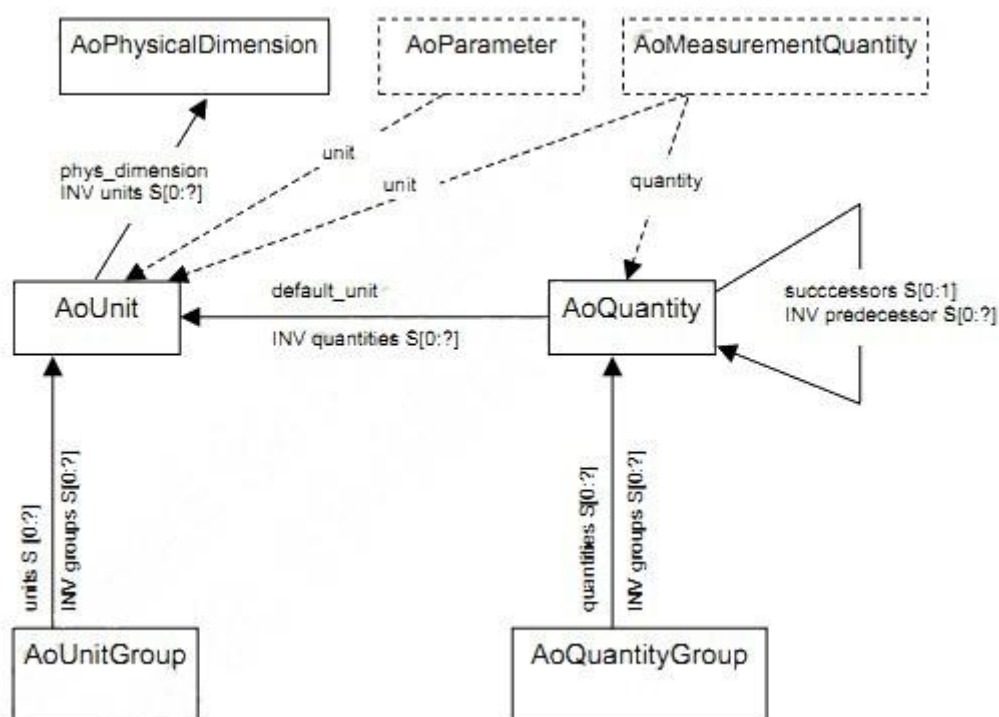


图39 单位、量和物理维度之间的关系

请注意符号：

e) $S[0: ?]$ = 一个集合，元素从 0 开始，集合里元素的次序互不相关。

5.10.3.1 量和它们的名称的概述

在今天所使用的测量和测试系统中，各种各样的信息编码在量名称中，此外，这往往受限于例 8 的字符。

这种信息可能是：

- a) 物理量本身的名称（强度、时间、浓度）；
- b) 通常情况下这种量的相关位置（铰接轴在前方、在左边）；
- c) 从其他量中（平均的、过滤的）确定该量通常使用的处理模式。

这种信息编码使它几乎不可能自动让计算机根据部分信息处理量（例如，列出了对在左前方的一个对象的特征进行说明的所有量）。另外，一个量的说明与测量这样一个量时所建立的配置、所取得的结果往往没有区别。

另一个特征，量所要求的，是在整个环境中，对标准量使用相同名称的要求。为此，应该在一个清单中列出这些名称（作为标准量名称目录）。

另一方面，为一个只测量一次的量定义一个全球名称没有意义。

5.10.3.2 ASAM ODS 内的量的层次结构

ASAM ODS 内的量形成了一系列需要测量的事情，它们排列在一个层次结构关系中。

每个“量”的使用可能有两种方法：作为一个属于一次“测量”中的“测量量”，另一个是，“量”的层次结构作为先秦。请注意，ASAM ODS 区分一个量本身（作为一个中性说明）与一次测量的主题。无论何时初始化一个量的测量，如：力、压力，……，必须首先实例化一个“测量量”，并连接到（通过设置一个引用）一个“量”。“测量”只了解其“测量量”。每个“测量量”指向一个“量”，因此，提供了当前所测量的信息；它引用一个量并接收该量的几个参数，如：该量的物理维度和一个默认单位。研究“AoMeasurement”，“AoMeasurementQuantity”和“AoQuantity”的属性和关系将进一步清楚它们的相互依赖性。

在层次结构中最高量的量，例如，那些没有任何先秦的量，一定是物理量，可以附加一个单位。在正常的情况下，这将是一个物理特性（如，力），而不是一个位置、一种物质或诸如此类（例如，FL 或 CO）。因此，这保证了每个“量”拥有一个单位，并可能用在“测量量”中（在这个意义上，“无量纲”或“[-]”也被视为一个单位）。

应该注意到量层次结构不依赖命名惯例，而是依赖“量”的实例彼此之间的引用。这些引用（后辈、先秦）嵌在 ASAM ODS 基础模型中。

为了快速识别的目的，每个量可以附上缩写，并且必须唯一。

5.10.3.3 ASAM ODS 内的量的用法

“量”可以结合在“量组”中，一个“量”可以是很多量组的一个成员。这允许可以根据应用特定方面（“量组”的例子：设定值、温度、废气值）对“量”进行分组。

每个“量”在一次测量中可以被使用一次或一次以上。如果“量”在一次单一测量中使用一次，这是通过测量和连接的测量量名称来确定的。可以定义一个其他的“测量”-特定名称。应用使用“量”的“默认测量量名称”（属性“默认_mq_名称”）作为“测量量名称”的默认名称。如果一个“量”在一次单一测量中被重复使用多次，那么，必须给每个“测量量”命名一个不同的“测量量名称”，名称上附加上“量”。

以这种方式可以定义本地量（关于每次“测量”）。例如，如果应该测量三个温度，而这些温度对其他测量不重要，那么，被引用的“量”是 T，“测量量名称”可以创建为，如：T1, T2 和 T3。“量”T 有一个量类型的功能，例如，它提供单位。

5.10.3.4 单位转换

每个本地单位可以通过 SI 单位的乘积 $\prod_{i=1}^7 SI - BU_i^{\text{exp}_i}$ 来表示，这里多提供的 SI 基本单位(SI-BU) {长度单位为 m，质量单位为 kg，时间单位为 S，电流单位为 A，温度单位为 K，摩尔量单位为 mol，发光强度单位为 cd} 带有一个带符号（在大多数情况下是整数）的指数（Expn），例如，加速度的单位为 m s⁻²。如果不需要表示一个 SI 基本单位，那么，其指数等于 0。

为了转换所给出的用本地单位表示的值为 SI 单位，通常要求乘以一个转换因子。例如，加速度的单位 g 转换为 m/s²，导致乘以转换因子 9.81（1g=9.81 m s⁻²）。至少在处理温度的情况下，必须额外处理一个相关单位的偏移，因为温度的基本单位为 Kelvin，而它在零点的摄氏温度是-273.15℃。

示例：单位转换见图 40。

$$\begin{aligned}\text{Kelvin} &= \text{Centigrade} \times 1.0 \frac{\text{K}}{^{\circ}\text{C}} - 273.15\text{K} \\ \text{Kelvin} &= \text{Fahrenheit} \times 0.556 \frac{\text{K}}{^{\circ}\text{F}} + 255.37\text{K} \\ \text{degree} &= \text{radian} \times \frac{360^{\circ}}{2 \cdot \pi}\end{aligned}$$

图40 单位转换

在第一个例子中，因子是 1.0，偏移是 273.15；

在第二个例子中，因子是 0.556，偏移是 255.37K

在第三个例子中，因子是 $2\pi/360^{\circ}$ ，偏移是 0（请注意，radian 是 SI 单位！）。

总结上面所述，转换用“本地单位”表示的本地值“值”到用“SI 单位”表示为“SI 值”的方程如下：

$$\text{SIValue} \left[\prod_{i=1}^7 \text{SI-BU}_i^{\text{exp}_i} \right] = \text{LocValue}[\text{LocalUnit}] \times \text{factor} \left[\frac{\prod_{i=1}^7 \text{SI-BU}_i^{\text{exp}_i}}{\text{LocalUnit}} \right] + \text{offset} \left[\prod_{i=1}^7 \text{SI-BU}_i^{\text{exp}_i} \right]$$

必须存储以下信息：

- 本地单位的名称（ASCII 文本）；
- 国际单位制（SI）基本单位的所有的七个带符号的整数指数（还有在分数指数情况下它们的整数分母）；
- 因子，和；
- 偏移（两者都是实数）。

国际单位制（SI）单位表见表 17。

表17 国际单位制（SI）单位表

索引	数量类型	单位名称	符号
1	长度	米	m
2	质量	千克	kg
3	时间	秒	s
4	电流	安培	A
5	温度	开尔文	K
6	摩尔量	摩尔	mol
7	发光强度	坎德拉	cd

所有七个国际单位制(SI)基本单位维度的指数必须作为 “AoPhysicalDimension” 的属性存储。如果相应的国际单位制 (SI) 基本单位无助于物理维度, 那么, 0 存储在该属性中)。

注: 没有单位的量和无量纲量的单位:

“量” 有一定的“值格式”, 例如, 字符或位没有任何“单位”。

不同于存在的“无量纲”量, 这些“无量纲”量拥有“单位”, 并且所有的指数=0 (例如, [-], [%], 或其它)

5.10.4 别名用法

ASAM ODS 已经提供了一些为应用元素、应用属性和关系的名称指定别名的方法。这些方法可以被用来指定一组语言的名称, 还有其它使用情况。然后, 客户端应用可以在原名及任何别名之间替换 (每个/种替代语言/使用情况一个别名)。

在基础元素 AoEnvironment 可以找到初始起点。在那里, 该基础元素只有一个应用元素, 也正好一个实例。

基本属性‘别名_的意义’派生的应用属性的值可以存储在几个说明中 (作为一个有序列表), 每个说明解释别名表示什么。本节的其余部分设 N 是该列表的大小。

这些说明还可以被用来指定一个别名集合的语言。在说明结束, 由一部分字符串指定语言, 而由一个逗号与说明分开, 也可选择在后面留一个或多个空格 (作为空白字符)。根据 ISO 639-1, 这部分字符串必须服从语法 “language=<langcode>”, 以<langcode>作为一个代码。

示例 1: 语言代码说明示例: “language=de”; “testbench, language=en”; “test, analysis, language=en”。

示例 2: 不用语言代码的说明示例: “testbench”; “test, analysis”; “Test, language”; “language=en”。

从基本关系‘实体_映像’派生的应用关系的值涉及到从 AoNameMap 派生的 (单一) 应用元素的实例的一个集合。该元素存在的实例应该与应用模型中的应用元素一样多。

可通过基础元素 AoNameMap 和 AoAttributeMap 找到应用元素及它们的属性、关系的别名。一个支持多语言环境的应用元素必须正好包含这些基础元素中每个基础元素的一个应用元素。

在一个应用模型内, 多语言支持的每个应用元素必须:

——刚好有一个基础元素 AoNameMap 派生的应用元素的相应的实例, 及;

——刚好有一个基础元素 AoAttributeMap 派生的应用元素的每个属性和关系相应的实例。

基础元素 AoNameMap 派生的应用元素的一个实例对应于一个应用元素, 如果基本属性‘实体_名称’派生的属性的值与该应用元素的原名相同的话。

基础元素 AoAttributeMap 派生的应用元素的一个实例对应于一个应用属性或关系, 如果基本属性‘属性_名称’派生的属性的值与该应用属性各个关系的原名相同的话。

基础元素 AoNameMap 派生的应用元素的实例包含一个基本属性‘别名_名称’派生的属性。其值是一个列表, 该列表应该刚好由 N 个条目组成。其第 n 个条目对应于由基本属性‘别名_的意义’ (参见上面所述) 派生的应用属性的列表-值中的第 n 个条目指定的语言。它提供了第 n 个替代语言的相应的应用元素的别名。请注意, 列表内的次序很重要。

基础元素 AoAttributeMap 派生的应用元素的实例包含一个基本属性‘别名_名称’派生的属性。其值是一个列表, 该列表应该刚好由 N 个条目组成。其第 n 个条目对应于由基本属性‘别名_的意义’ (参见上面所述) 派生的应用属性的列表-值中的第 n 个条目指定的语言。它提供了第 n 个替代语言的相应

的应用元素的别名。请注意，列表内的次序很重要。

语言选择、相关别名的检索和表示由客户端应用独立负责。ASAM ODS 服务器只作为一个别名名称的知识库；它不将原名转换成别名。

使用 APIs 存储 ASAM ODS 服务器的数据时，提供给服务器（例如，查询）的所有名称、服务器返回的所有名称都是原名。由客户端应用负责将原名转换成别名。

写入 ATF 文件时，用于应用元素、属性或关系的所有名称必须是原名。如果别名也是由 ATF 传输，那么，基础元素 AoNameMap 和 AoAttributeMap 派生的应用元素的实例必须从 ATF 文件导出；这些实例包含了别名。

如果应用模型中的应用元素没有 AoNameMap 派生的应用元素的相应的实例，这些应用元素和他们的属性、关系就只能用它们的原名来表示。在这种情况下，一个客户端应用可以提示用户这些是没有别名的。

一个应用元素有 AoNameMap 派生的应用元素的相应的实例，但它的一些应用属性和/或关系没有 AoAttributeMap 派生的应用元素的一个相应的实例，在这种情况下，这些应用属性各个关系只能使用它们的原名表示。在这种情况下，一个客户端应用可以提示用户这些缺省的别名。

如果基本属性‘别名_名称’派生的属性列表大小小于基本属性‘别名_的意义’派生的属性列表的大小，可能就会为没有可用别名名称的那张列表选择语言。在这种情况下，必须显示原名。在这种情况下，一个客户端应用可以提示用户这些缺省的别名。

如果基本属性‘别名_名称’派生的属性列表大小大于基本属性‘别名_的意义’派生的属性列表的大小，该列表的别名名称就比可选取的语言多。在这种情况下，必须显示原名。在这种情况下，一个客户端应用可以提示用户这些过剩的别名。

简单模型的例子见图 41。

示例：一个简单的模型 42。

假设一个应用模型有几个应用元素，被用来说明一组汽车。基础元素、属性、关系的名称在括号中给出；为清晰起见，逆关系已被省略。

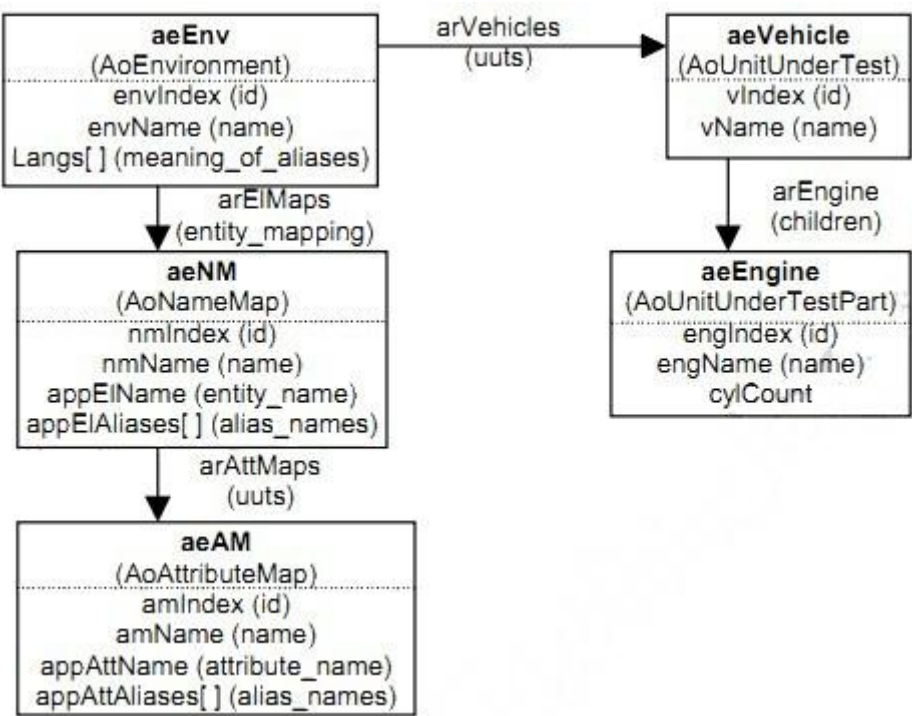


图41 简单模型的例子

如果要用其他语言来表示该应用模型，这些应用元素的示例的一个样本集合见图 42（省略了车辆和引擎的实例）。

aeEnv的实例: envIndex: 1 envName: "iEnv" Langs[]: ".language=de" ".language=fr" ".language=en"		aeNM的实例, 它们都与iEnv 相关: nmIndex: 1 2 3 4 5 nmName: "iNM1" "iNM2" "iNM3" "iNM4" "iNM5" appElName: "aeEnv" "aeNM" "aeAM" "aeVehicle" "aeEngine" appElAliases[]: "Umng" "ElNamen" "AttNamen" "KFZ" "Motor" "Env" "ElNoms" "AttNoms" "vehicule" "moteur" "aeEnv" "aeNM" "aeAM" "aeVehicle" "aeEngine"				
aeAM的实例, 它们都与iNM1相关: amIndex: 1 2 3 4 5 amName: "iAM1" "iAM2" "iAM3" "iAM4" "iAM5" appAttName: "envIndex" "envName" "Langs" "arElMaps" "arVehicles" appAttAliases[]: "UmngID" "UmngName" "Sprachen" "ElNamen" "Fahrzeuge" "envID" "envNom" "langues" "elNoms" "vehicules" "envIndex" "envName" "Langs" "arElMaps" "arVehicles"		aeAM的实例, 它们都与iNM2相关: amIndex: 6 7 8 9 10 amName: "iAM6" "iAM7" "iAM8" "iAM9" "iAM10" appAttName: "nmIndex" "nmName" "appElName" "appElAliases" "arAttMaps" appAttAliases[]: "NMID" "NMName" "appElName" "appElPseudos" "AttNamen" "nmID" "nmNom" "appElNom" "appElPseudos" "attNoms" "nmIndex" "nmName" "appElName" "appElAliases" "arAttMaps"				
aeAM的实例, 它们都与iNM3相关: amIndex: 11 12 13 14 amName: "iAM11" "iAM12" "iAM13" "iAM14" appAttName: "amIndex" "amName" "appAttName" "appAttAliases" appAttAliases[]: "AMID" "AMName" "appAttName" "appAttPseudos" "amID" "amNom" "appAttNom" "appAttPseudos" "amIndex" "amName" "appAttName" "appAttAliases"		aeAM的实例, 它们都与iNM4相关: amIndex: 15 16 17 amName: "iAM15" "iAM16" "iAM17" appAttName: "vIndex" "vName" "arEngine" appAttAliases[]: "FzID" "FzName" "Motor" "vehID" "vehNom" "moteur" "vIndex" "vName" "arEngine"				
aeAM的实例, 它们都与iNM5相关: amIndex: 18 19 20 amName: "iAM18" "iAM19" "iAM20" appAttName: "engIndex" "engName" "cylCount" appAttAliases[]: "MotID" "MotName" "ZylZahl" "motID" "motNom" "nombCyl" "engIndex" "engName" "cylCount"						

图42 应用模型中的应用元素

这允许, 例如切换到法语 (“fr”), 因此, 也允许用 “nombCyl” 表示应用属性 “cylCount”。请注意, 与元素和属性名称的原名与语言无关。为此, 它们还需要用别名表示, 如这个例子那样, 使它们与英语关联起来。

5.10.5 ASAM ODS 中的枚举

一个枚举是一种数据类型, 它由一个名称和一个枚举项目集合组成, 这些项目是名称-值-对, 每个包含一个名称, 一个非负整数。在下列三种情况时, 一个枚举数据类型用途很大: 需要从一个有限的项目集合中选择一个项目是, 需要在一台计算机或接口使用全面表示 (被选项目的编号) 时, 人类用户需要使用一个解释性表示 (被选项目的名称) 时。

ASAM ODS 本身在基础模型中指定了几个枚举 (datatype_enum, interpolation_enum, seq_rep_enum, typespec_enum; 请参见第 7 章)。这些是预定义枚举; 在未来会出现另外的预定义枚举。

一个应用模型可以添加任何数量的特定枚举。

所有枚举的名称及它们的项目的名称区分大小写。

在一个应用模型内，所有枚举的名称必须唯一。

在一个枚举中，所有枚举项目的名称必须唯一。

本节说明如何在 ASAM ODS 中管理和使用枚举。

(1) 枚举的存储

1) 在 ASAM ODS 服务器中，每个枚举的定义通过枚举类型的名称、一个枚举标识符来存储，而每个枚举项目的存储则通过项目名称和项目编号。枚举名称、项目名称的长度是受限的。详情请参见第 6.2.6 节；

2) 在 ATF/CLA 中，一个枚举由其项目名称的有序列表定义。不存储项目编号。项目编号从 0 开始，随后递增。假设预定义的枚举已知，因此，不必要在 ATF 文件内定义。可以通过项目的名称或编号来引用项目。详情请参见第 9 章；

3) 在 ATF/XML 中，一个枚举由枚举名称和一个名称-值-对集合定义（一个枚举类型项目一个名称-值-对）。项目编号必须从 0 开始，随后递增。预定义的枚举不得在 ATF/XML 中定义，以避免定义模棱两可。只能通过项目的名称来引用项目。详情请参见第 10 章。

(2) 通过 APIs 存取枚举

1) 设置或检索每个枚举类型属性的属性值时，OO-API 要求使用数据类型 DT_ENUM，因此，用枚举项目编号指定枚举项目；

2) OO-API 提供处理枚举的接口‘EnumerationDefinition’。其每个实例代表一个枚举。

可以在一个应用模型内创建任何数量的它的实例。可以从 ODS 服务器中取得一个应用模型中内的所有现有的枚举的名称的一个列表。现有的枚举可以通过它们的枚举名称存取，也可以从应用模型中删除。此外，通过 API 可以取得基本属性的枚举。

3) 可以添加项目到 EnumerationDefinition，现有项目（与枚举本身）可以重命名，然而，不能删除。一个项目的相关编号不能设置也不能改变。由 ODS 服务器给添加到 EnumeraionDefinition 的第一个项目设置编号 0，随后，再添加的每个项目的编号递增。

如果一个应用属性是枚举类型，那么，它可能被赋予一个 EnumerationDefinition 实例，指定特定的枚举。

上述中的一些操作只限制于一个超级用户。详情请参见第 13 章。

4) RPC-API 不支持枚举的使用。预定义枚举在服务器内编码，并使用得当。不能添加特定枚举，属性有枚举数据类型的应用模型不能与 RPC-API 一起使用。

(3) 处理枚举的建议

1) 一个客户端应用应该为其使用的每个枚举定义创建一个本地拷贝。

这可避免调用大量的方法只是为了用户接口之目的而分解项目编号为项目名称。

2) 枚举项目一旦存储在不同的地方就不应该重命名。

否则，会产生不一致的数据。此外，如果多个用户共用一台服务器，客户可能产生错误或甚至卡住。

5.10.6 在一台 ASAM ODS 服务器存取文件

5.10.6.1 起因和概念

ASAM ODS 旨在管理测量数据连同说明性数据。这些数据通常存储在一个数据库中(在第 6 章说明), 由一个客户端应用通过 OO-API 或 RPC-API 存取。

然而, 在几种情况下, 这些数据(虽然它们由 ASAM ODS 服务器管理)存储在文件中而不是数据库中。这是有用的, 例如, 在下列情况中:

1) 附加信息, 如包含在单独文件的图片、视频、音频文件、或其他大容量数据对象应该添加到一个应用元素的一个实例。

添加这些信息可以使用基本应用属性‘外部_引用’, 而不是通过定义数据类型 **T_BYTESTR** 或 **T_BLOB** 的扩展应用属性。该属性允许添加任何数量的外部引用到一个实例中。在一台 ASAM ODS 服务器(由一个 ASAM 路径定义)或位置可以由一个统一资源定位器(URL)说明的任何资源内, 一个外部引用可以是另一个实例的一个引用。后者允许引用单独文件。

2) 海量数据(例如, 由测量装置产生)应该存储在与数据库分开的二进制文件中。

这些海量数据可以存储在一个或多个组件文件中, 而不是存储在数据库(根据第 6.2.4, 存储在表 **SVCVAL**)的本地列中。这被称为‘混合-模式存储’, 并在第 6.4 节进一步说明。这些文件的结构和位置在 ASAM ODS 服务器(由 **AoExternalComponent** 类型的应用元素的实例说明)内说明, 而海量数据(确切地说, 本地列实例的‘值’和‘标志’)包含在单独的二进制文件中。

这些情况可发生在 ASAM ODS 内, 并使用三个项目指定这些文件的位置:

1) **location**: 这个是(结构)数据类型 **T_ExternalReference** 的‘位置’部分: 在引用一个文件的情况下, 它指定该外部引用文件的统一资源定位器(URL);

2) **filename_url**: 这个是基础元素 **AoExternalComponent** 的基本属性‘文件名_url’: 它指定包含了相应的本地列的‘值’的外部组件文件的统一资源定位器(URL);

3) **flags_filename_url**: 这个是基础元素 **AoExternalComponent** 的基本属性‘标志_文件名_url’: 它指定包含了相应的本地列的‘标志’的外部组件文件的统一资源定位器(URL)。

这三个项目指定一个外部文件的位置的方法非常相似。其方法及它们的用法在本节说明。

请注意, 本节内使用词组‘外部组件文件’或‘外部引用文件’时, 指的正是这种类型的文件。如果为说明两个类型, 那么, 将使用词组‘外部文件’。

这三个项目既可以给出任何外部文件的位置的绝对路径, 也可以给出相对路径。

一个绝对路径一个文件位置的唯一说明, 信息完备。

一个绝对路径的例子有“file : //data.asam.net/test.jpg”, “//plato/user_root/data/test.jpg”等。这些路径通向相同的文件, 不管从什么机器存取它们(假设 DNS 的设置正确, 网络存取可用)。

一个位置规范从一台计算机来看是绝对路径的另一个例子是“D: /user_root/data/test.jpg”。然而, 不同的计算机通常会结束存取不同的文件。

一个相对路径要求(不明确地或明确地)额外指定一个根路径; 根路径与相对路径组成一个绝对路径。

一个相对路径的一个例子是“data/test.jpg”。如果根路径是“//plato/d/user_root/”, 那么, 绝对路径的组成是“//plato/d/user_root/ data/test.jpg”(注意: 对于一台特定的计算机, 这个路

径仍然只是绝对路径)。

在 ASAM ODS 内, 外部文件的位置可以指定为绝对路径或相对路径。

绝对路径应该只能在一台 ASAM ODS 服务器安装内使用, 如果外部文件在其使用期间肯定不会被移动(例如, 被系统管理员移动)的话。

否则, 需要调整存储在 ASAM ODS 服务器的所有路径到新的位置。

相对路径是首选, 如果, 比如, 外部文件位于 ASAM ODS 服务器的一个(真实的或虚拟的)驱动器中, 而系统管理员常常决定重新排列文件存储系统。在这种情况下, 常常只需要改变根路径。

ASAM ODS 提供从语境变量确定根路径的方法。因此, 在移动外部文件的情况下, 唯一可做的是调整这些语境变量的内容(应该由系统管理员在移动文件时完成)。

此外, ASAM ODS 区分小规模解决方案(Single_Volume 模式)和大规模解决方案(Multi_Volume 模式), 在小规模解决方案中, 所有的外部文件所在的位置基于相同根路径, 而在大规模解决方案中, 所有的外部文件所在的位置可以有不同的根路径(例如, 位于不同的驱动器)。

语境变量是说明一台 ASAM ODS 服务器在具体方面的行为的变量。它们可以由客户端通过 APIs 进行存取, 在第 13.3.22 节中列出, 说明了如何通过 OO-API 存取, 在第 13.7 节列出, 说明了如何通过 RPC-API 存取。客户端可以设置并读取其中一些这样的变量, 其他变量只能是只读的(那些只能在 ODS 服务器中设置, 例如, 在启动服务器期间)。

下列语境变量控制外部文件的位置。通过 APIs 存取这些外部文件时, 它们都是只读的, 并且只能在启动服务器期间设置和/或修改。因此, 在整个会话过程, 客户端可以放心, 它们是恒定值。

(1) FILE_NOTATION

该语境变量定义了语法(包括一些语义), 在这些语法语义中, 指定了位置。其值可以是这些字符串值之一: 'UNC_WIN', 'UNC_UNIX', 'URL' (UNC=通用命名约定)。

在取值为'UNC_WIN'的情况下, 字符串的个别部分指定一个文件的位置, 这些个别部分由一个反斜杠字符('\')隔开, 同它在 Windows 操作系统的处理一样。一般语法是'\\<hostname>\<path>', 此处, <hostname>指定该文件位置的主机的名称或 IP 地址, 而<path>指定了目录路径和文件本身的文件名称。前部分'\\hostname\'是可选的, 可以省略, 如果外部文件位于 ODS 服务器本身的一个(真实的或虚拟的)驱动器的话。

例如: '\\plato\d: \data\test.dat', 'x: \tests\data.csv'。

在取值为'UNC_UNIX'的情况下, 字符串的个别部分指定一个文件的位置, 这些个别部分由一个斜杠字符('/')隔开, 同它在类 UNIX 操作系统的处理一样。一般语法是'//<hostname>/<path>', 此处, <hostname>指定了该文件位置的主机的名称或 IP 地址, 而<path>指定目录路径和文件本身的文件名称。前部分'//hostname/'是可选的, 可以省略, 如果外部文件位于 ODS 服务器本身的一个(真实的或虚拟的)驱动器的话。

例如: '//plato/d: data/test.dat', 'x: /tests/data.csv'。

在取值为'URL'的情况下, 字符串的个别部分指定一个文件的位置, 这些个别部分由一个斜杠字符('/')隔开, 前面是用来存取文件的协议, 同它在互联网的处理一样。一般语法是'<protocol>://<hostname>/<path>', 此处, <protocol>指定存取文件的协议, <hostname>指定该文件位置的主机的名称或 IP 地址, 而<path>指定了目录路径和文件本身的文件名称。因此, 目录路径是相对于主机上协

议服务端的根目录的。在目前的版本上，<protocol>只可能是‘文件’。

例如：‘file: //177.2.3.4/d: /data/test.dat’，‘file: //www.plato.asam.net/tests/data.csv’。

FILE_NOTATION 的设置 ASAM ODS 服务器内对所有绝对或相对位置规范都有效。它已经被引入，以缓解项目‘位置’、‘文件名_url’，和‘标志_文件名_url’的解析过程：如果客户端确定其自己的表示法对应于文件_表示法，在访问该服务器时，一般不需要用‘\’代替‘/’，也不需要‘/’代替‘\’，如果提供字符串，可以使用字符串。

(2) FILE_MODE

该语境变量指定服务器管理外部文件位置的方法。其值可以是这些字符串值之一：‘ABSOLUTE’，‘SINGLE_VOLUME’和‘MULTI_VOLUME’。

在取值为‘ABSOLUTE’的情况下，每个文件位置存储为绝对路径。

在取值‘SINGLE_VOLUME’的情况下，每个文件位置存储为相对路径。必须通过一个根路径串联该相对路径的方式创建文件的绝对路径。外部组件文件的所有相对路径基于一个共同根路径。外部引用的所有相对路径也是基于（也可能是另一个）一个共同根路径。

在取值为‘MULTI_VOLUME’的情况下，每个文件位置也存储为相对路径，但是，与该相对路径相关的根路径带有附加信息。必须通过一个相应的根路径串联相对路径的方式创建文件的绝对路径。服务器可以有各种外部文件根路径，其中一个作为外部引用文件的默认根路径，（可能是另一个）其中一个作为外部组件文件的默认根路径。

(3) FILE_SYMBOLS

该语境变量提供 ODS 服务器已知的所有文件符号名称列表。其值是一个逗号分隔的字符串列表，每个字符串是一个文件符号名称。

一个文件符号名称一个指定相对 URLs 的一个根路径的语境变量自身的名称。一个文件符号名称由字母-数字字符组成，包括下划线，不允许其他字符，也不可以包含空白。

在 FILE_MODE=ABSOLUTE 的情况下，该语境变量不可用。

在 FILE_MODE=SINGLE_VOLUME 的情况下，该语境变量刚好包含两个文件符号名称：FILE_ROOT 和 FILE_ROOT_EXTREF；它们指定外部组件文件各个外部引用文件的根路径。

在 FILE_MODE=MULTI_VOLUME 的情况下，该语境变量包含服务器已知的所有文件符号名称（请注意，FILE_ROOT，FILE_EXTREF 不在它们之列，因为，在这种情况下，它们发挥特殊作用）。

(4) FILE_ROOT

该语境变量指定外部组件文件的根路径。

在 FILE_MODE=ABSOLUTE 的情况下，该语境变量不可用。

在 FILE_MODE=SINGLE_VOLUME 的情况下，该语境变量的值是一个字符串，该字符串赋予所有外部组件文件的根路径。

在 FILE_MODE=MULTI_VOLUME 的情况下，该语境变量的值是一个字符串，该字符串赋予该语境变量的名称，而该语境变量指定外部组件文件的默认根路径。默认根路径是存储外部组件文件的首选。

(5) FILE_ROOT_EXTREF

该语境变量指定外部引用文件的根路径。

在 FILE_MODE=ABSOLUTE 的情况下，该语境变量不可用。

在 `FILE_MODE=SINGLE_VOLUME` 的情况下，该语境变量的值是一个字符串，该字符串赋予所有外部引用文件的根路径。

在 `FILE_MODE=MULTI_VOLUME` 的情况下，该语境变量的值是一个字符串，该字符串赋予该语境变量的名称，而该语境变量指定外部引用文件的默认根路径。默认根路径是存储外部引用文件的首选。

(6) 扩展文件符号名称

这些语境变量中的每个变量提供 ASAM ODS 服务器已知的根路径中的一个根路径。这些语境变量的值是否指定外部组件文件或外部引用文件的根路径不是由它们的名称或任何其他特征预先确定的。这取决于它们是否用于 ‘location’ 的或 ‘filename_url’ 各个 ‘flags_filename_url’ 的取值。语境变量 ‘FILE_SYMBOLS’ 的取值也提供了它们的名称。

在 `FILE_MODE=ABSOLUTE` 或者 `FILE_MODE=SINGLE_VOLUME` 的情况下，扩展文件符号名称不可用。

表 18 提供了语境变量和它们出现的信息。

表18 不同文件模式出现的语境变量

file mode	ABSOLUTE	SINGLE_VOLUME	MULTI_VOLUME
context variable			
FILE_NOTATION	X	X	X
FILE_MODE	X	X	X
FILE_ROOT		X	X
FILE_ROOT_EXTREF		X	X
FILE_SYMBOLS		X	X
Further file symbol names			X

用这些语境变量，可以找到一个外部文件的绝对路径，如下：

1) 在 `FILE_MODE=ABSOLUTE` 的情况下，‘location’ 或 ‘filename_url’ 或 ‘flags_filename_url’ 给出的信息直接指定外部文件的绝对路径；

例如（针对 `UNC_UNIX`）：如果对于一个本地列的一个给定的实例，文件名_url= ‘//homer/d:/mydata/test.gif’，包含本地列的值的文件所在的主机名为 ‘homer’，驱动器是 ‘d’，所在目录是 ‘/mydata/’，文件名为 ‘test.gif’。

2) 在 `FILE_MODE=SINGLE_VOLUME` 的情况下：通过一个根路径串联 ‘location’ 或 ‘filename_url’ 或 ‘flags_filename_url’ 给出的信息的方式创建绝对路径。由语境变量 `FILE_ROOT`（针对外部组件文件的），`FILE_ROOT_EXTREF`（针对外部引用文件）的值赋予根路径。

例如（针对 `UNC_WIN`）：如果，对于一个本地列的一个给定的实例，标志_文件名_url= ‘test7\data_fl.bin’，文件_根_扩展引用= ‘\\platox: \tests\’，文件的绝对路径是 ‘\\platox: \tests\test7\data_fl.bin’，包含本地列标志的文件所在的主机的名为 ‘plato’，驱动器为 ‘x:’，所在目录为 ‘\tests\test7\’，文件名为 ‘data_fl.bin’。

3) 在 `FILE_MODE=MULTI_VOLUME` 的情况下，通过一个根路径串联相对路径的方式创建绝对路径。两部分取决于 ‘location’ 或 ‘filename_url’ 或 ‘flags_filename_url’ 给出的信息。在该种情况

下，其语法 ‘\$(<root_path_symbol>) <remainder>’，此处，<root_path_symbol>是保存根路径的语境变量的名称，<remainder>是相对路径。

因为，在这种情况下，服务器可以有几个根路径，由语境变量赋予默认路径（被用来创建新外部文件的那个路径），该语境变量的名称由 FILE_ROOT（用于创建新外部组件文件）及 FILE_ROOT_EXTREF（用于创建外部引用文件）提供。

例如（针对 UNC_WIN）：如果，对于一个本地列的一个给定的实例，文件名_url= ‘\$(Disk_3) \test7\data.bin’，语境变量 Disk_3= ‘\\plato\x : \tests’，文件的绝对路径是 ‘\\plato\x : \tests\test7\data.bin’，包含了本地列的值的文件所在的主机的名为 ‘plato’，所在驱动是 ‘x:’，目录是 ‘\tests\test7\’，文件名是 ‘data.bin’。

5.10.6.2 典型的用例

本节以下各段内容通过几个用例描述了语境变量及‘location’，‘filename_url’，‘flags_filename_url’取值的典型应用。

（a）在 ODS 服务器只允许绝对路径的情况下：

1）语境变量文件_模式的值设置为‘ABSOLUTE’，读取该语境变量的客户端因而被通知只支持绝对路径。

2）如果客户端通过使用 APIs 将一个本地列的值和/或标志存入一台服务器中，并且，服务器在写入_模式=文件（write_mode=file）下工作，那么，服务器将创建一个恰当的 AoExternalComponent 的实例，构建本地列的一个关系，自己确定存储这些值和标志的文件的位置（及内部结构）。保存取值的文件的位置将作为绝对路径存储在‘AoExternalComponent 的实例的‘filename_url’中。保存标志的文件的位置将作为绝对路径存储在‘AoExternalComponent 的实例的‘flags_filename_url’中。

在这种使用情况下，客户端不需要与 AoExternalComponent 的实例交互，也不需要与外部组件文件交互。

3）如果客户端通过使用 APIs 从一台服务器中读取一个本地列的值和/或标志，那么，服务器将（从基本属性‘序列_表示’中）确定它们是否存储在外部组件文件中。如果是，服务器将存取 AoExternalComponent 相应的实例，并从‘文件名_url’中确定（除了内部结构之外，还有…）保存取值的文件的绝对路径，从‘标志_文件名_url’中确定保存标志的文件的绝对路径。服务器将读取文件并向客户端返回取值和/或标志。

在这种使用情况下，客户端不需要与 AoExternalComponent 的实例交互，也不需要与外部组件文件交互。

如果客户端通过提供 ODS-兼容的外部组件文件将一个本地列的值和/或标志写入服务器中，那么，客户端将创建一个恰当的 AoExternalComponent 实例，构建本地列的一个关系，并相应地设置 AoExternalComponent 的实例的属性，以便它们能够反映内部结构。客户端将拷贝外部组件文件到一台服务器能够访问的位置，并将保存取值的文件的位置作为绝对路径存储在 AoExternalComponent 的实例的‘文件名_url’中，将保存标志的文件的位置作为绝对路径存储在 AoExternalComponent 的实例的‘标志_文件名_url’中。

这不是一个平常的用例，因为，客户端需要具备关于服务器访问位置的专业知识，并且必须有写入这些位置的存取权限。

然而，在特殊情况下，这可以加速将数据存储入一台 ODS 服务器的过程，特别是需要处理大量数据的时候，因此，避免了海量数据通过 APIs 传输。

4) 如果客户端通过从一台 ODS 服务器的驱动器中拷贝外部组件文件读取一个本地列的取值和/或标志，那么，客户端首先必须（从基本属性‘序列_表示’中）确定它们是否真的存储在外部组件文件中。如果是，那么，客户端将会存取 **AoExternalComponent** 的相应的实例，并从‘文件名_url’中确定保存取值的文件的绝对路径，从‘标志_文件名_url’中确定保存标志的绝对路径。然后，客户端将会拷贝这些文件到其自己的存储域，并可以根据内部结构从 **AoExternalComponent** 的实例读取的方式处理它们。

这不是一个平常的用例，因为，客户端需要具备关于如何存取文件的专业知识（特别是如果文件_表示法不是‘URL’，绝对路径信息从服务器的角度给出，像“D: /root/data/t.dat”），必须有读取这些文件/位置的存取权限。然而，在特殊情况下，这可以加速从一台 ODS 服务器加载数据的过程，特别是需要处理大量已经存储在外部组件文件的数据的时候，因此，避免了海量数据通过 APIs 传输。

5) 如果客户端链接一个外部引用文件到 ODS 服务器中的一个实例（通过指定基本属性‘外部_引用’派生的一个应用属性的值），那么，客户端将会设置‘位置’的值到指定那个引用文件的绝对路径的一个字符串中。

虽然外部引用文件可能位于互联网的任何地方（显然，如果文件_表示法=URL），那么，建议客户端应该拷贝外部引用文件到 ODS 服务器控制下的任何位置。否则，可能无法保证可用性。这要求客户端具备服务器海量存储结构方面的专业知识，并且必须具备写入这些位置的存取权限。

ODS 服务器不会使用存储在‘位置’的信息，也不会存取外部引用文件。它只允许客户端设置‘位置’的值。

6) 如果客户端想要存取一个链接 ODS 服务器中的一个实例的外部引用文件，他将通过读取‘位置’确定该文件的绝对路径。

客户端必须具备读取这些位置的权限。

ODS 服务器不会使用存储在‘位置’的信息，也不会存取外部引用文件。它只向客户端提供关于‘位置’的信息。

(b) 在 **SINGEL_VOLUME** 模式下，ODS 服务器只允许相对路径的情况下：

1) 设置语境变量文件_模式(**FILE_MODE**)的值(在启动服务器时)为‘**SINGKE_VOLUME**’，读取该语境变量的客户端因而会被通知只支持相对路径。

2) 设置(在启动服务器时)语境变量文件_根(**FILE_ROOT**)的值为一个字符串值，该字符串值表达所有外部组件文件的根路径(当前和未来)。

3) 设置(在启动服务器时)语境变量文件_根_外部引用(**FILE_ROOT_EXTREF**)的值为一个字符串值，该字符串值表达所有外部引用文件的根路径(当前和未来)。

4) 设置语境变量(在启动服务器时)文件_符号(**FILE_SYMBOL**)的值为一个用逗号分隔的

字符串列表，该字符串列表包含两个文件符号名称：‘文件_根’和‘文件_根_外部引用’。

5) 如果客户端通过使用 APIs 将一个本地列的值和/或标志存入一台服务器中，并且，服务器在写入_模式=文件（write_mode=file）下工作，那么，服务器将创建一个恰当的 AoExternalComponent 的实例，构建本地列的一个关系，自己确定存储这些值和标志的文件的位置（及内部结构）。这些位置将基于文件_根（FILE_ROOT）指定的根路径。保存取值的文件的位置将被分解为根路径和剩余；该剩余作为相对路径存储在‘AoExternalComponent 的实例的‘文件名_url’（filename_url）中。保存标志的文件的位置将同样被分解为根路径和剩余；该剩余作为相对路径存储在‘AoExternalComponent 的实例的‘标志_文件名_url’（flags_filename_url）中。

在这种使用情况下，客户端不需要与 AoExternalComponent 的实例交互，也不需要与外部组件文件交互。

6) 如果客户端通过使用 APIs 从一台服务器中读取一个本地列的值和/或标志，那么，服务器将（从基本属性‘序列_表示’中）确定它们是否存储在外部组件文件中。如果是，服务器将存取 AoExternalComponent 相应的实例。服务器将串联根路径（从文件_根（FILE_ROOT）获取的）与相对路径（从文件名_url（filename_url）获取的），组成保存取值的文件的绝对路径。服务器串联根路径（从文件_根（FILE_ROOT）获取的）与相对路径（从文件名_url（filename_url）获取的），组成保存标志的文件的绝对路径。服务器将读取文件并向客户端返回取值和/或标志。

在这种使用情况下，客户端不需要与 AoExternalComponent 的实例交互，也不需要与外部组件文件交互。

例如（针对 UNC_WIN）：

如果文件_根（FILE_ROOT）=‘d:\data\externalcomponents\’，文件名_url（filename_url）= “ TestSerie1\ExternalFile1.dat ”，文件的绝对路径是“ d : \data\externalcomponents\TestSerie1\ExternalFile1.dat”。

7) 如果客户端通过提供 ODS-兼容的外部组件文件将一个本地列的值和/或标志写入服务器中，那么，客户端将创建一个恰当的 AoExternalComponent 实例，构建本地列的一个关系，并相应地设置 AoExternalComponent 的实例的属性，以便它们能够反映内部结构。客户端将拷贝外部组件文件到服务器驱动器上的根路径（由文件_根（FILE_ROOT）赋予的）内的一个位置上。客户端将分解保存取值的文件的位置为根路径和一个剩余，并将该剩余作为相对路径存储入‘AoExternalComponent 的实例的‘文件名_url’（filename_url）中。客户端将分解保存标志的文件的位置为根路径和剩余，并将该剩余作为相对路径存储入‘AoExternalComponent 的实例的‘标志_文件名_url’（flags_filename_url）中。

这不是一个平常的用例，因为，客户端需要具备关于如何访问根路径所在的服务器的驱动器的专业知识，并必须有驱动器的写入存取权限。然而，在特殊情况下，这可以加速将数据存入一台 ODS 服务器的过程，特别是需要处理大量数据的时候，因此，避免了海量数据通过 APIs 传输。

8) 如果客户端通过从一台 ODS 服务器的驱动器中拷贝外部组件文件读取一个本地列的取值和/或标志，那么，客户端首先必须（从基本属性‘序列_表示’中）确定它们是否真的存储在外部

组件文件中。如果是，那么，客户端将会存取 **AoExternalComponent** 的相应的实例，并串联根路径（从‘**FILE_ROOT**’中获取）与相对路径（从‘**filename_url**’中获取），获取保存取值的外部文件的绝对路径。此外，客户端将串联根路径（从‘**FILE_ROOT**’中获取）到相对路径（从‘**flags_filename_url**’中获取），获取保存标志的外部文件的绝对路径。然后，客户端将会拷贝这些文件到其自己的存储域，并可以根据内部结构从 **AoExternalComponent** 的实例读取的方式处理它们。

这不是一个平常的用例，因为，客户端需要具备关于如何访问保存根路径的服务器的驱动器的专业知识，并必须有读取该驱动器的存取权限。然而，在特殊情况下，这可以加速从一台 **ODS** 服务器加载数据的过程，特别是需要处理大量已经存储在外部组件文件中的数据的时候，因此，避免了海量数据通过 **APIs** 传输。

9) 如果客户端链接一个外部引用文件到 **ODS** 服务器中的一个实例（通过提供基本属性‘外部引用’（‘**external_references**’）派生的一个应用属性的值），那么，客户端将会拷贝该文件到 **ODS** 服务器的根路径（由‘**FILE_ROOT_EXTREF**’赋予）内的一个位置。然后，客户端将分解文件的绝对路径为根路径和一个剩余，并设置‘位置’的值为一个字符串，该字符串表示该剩余。

客户端必须具有在根路径内写入位置的存取权限。

ODS 服务器不会使用存储在‘位置’的信息，也不会存取外部引用文件。它只向客户端提供关于根路径的信息，并允许客户端设置‘位置’的值。

10) 如果客户端想要存取一个链接 **ODS** 服务器中的一个实例的外部引用文件，他将串联根路径（从 **FILE_ROOT_EXTREF** 获取）与‘位置’的值，确定该文件的绝对路径。

客户端必须具备读取这些位置的权限。

ODS 服务器不会使用存储在‘位置’的信息，也不会存取外部引用文件。它只向客户端提供关于根路径和‘位置’的信息。

(c) 在 **MULTI_VOLUME** 模式下，**ODS** 服务器只允许相对路径的情况下：

- 1) 设置语境变量文件_模式 (**FILE_MODE**) 的值（在启动服务器时）为‘**MULTI_VOLUME**’，读取该语境变量的客户端因而会被通知只支持相对路径。
- 2) 设置（在启动服务器时）语境变量文件_根 (**FILE_ROOT**) 的值为一个字符串值，该字符串值是指定外部组件文件的默认根路径的另一个语境变量的名称。
- 3) 设置（在启动服务器时）语境变量文件_根_外部引用 (**FILE_ROOT_EXTREF**) 的值为一个字符串值，该字符串值是指定外部引用文件的默认根路径的另一个语境变量的名称。。
- 4) 另外的语境变量和文件符号名称可以在服务器中指定（取决于组织 **ODS** 服务器的存储器的事前措施）。它们全部结合成一个用逗号分隔的文件符号名称列表，并存储在语境变量 **FILE_SYMBOL** 中。
- 5) 如果客户端通过使用 **APIs** 将一个本地列的值和/或标志存入一台服务器中，并且，服务器在 **write_mode=file** 下工作，那么，服务器将创建一个恰当的 **AoExternalComponent** 的实例，构建本地列的一个关系，自己确定存储这些值和标志的文件的位置（及内部结构）。这些位置

将基于默认根路径（由语境变量指定，该语境变量的名称由 FILE_ROOT 赋予）。

保存取值的文件的位置将被分解为默认根路径和一个剩余。该剩余位于一个默认路径信息的前面，该根路径被指定为字符串 “\$ (<root_path_symbol>)”。此处，<root_path_symbol> 是 FILE_ROOT 提供的字符串（注意：这是语境变量的名称，而不是根路径本身）。根路径的信息与剩余串联在一起，并作为相对路径存储在 ‘AoExternalComponent 的实例的 ‘filename_url’ 中。

保存标志的文件的位置将同样被分解为默认根路径和剩余；该剩余以同样的方式位于一个默认路径信息的前面，该根路径被指定为字符串 “\$ (<root_path_symbol>)”。此处，<root_path_symbol> 是 FILE_ROOT 提供的字符串。根路径的信息与剩余串联在一起，并作为相对路径存储在 ‘AoExternalComponent 的实例的 ‘flags_filename_url’ 中。

在这种使用情况下，客户端不需要与 AoExternalComponent 的实例交互，也不需要与外部组件文件交互。

6) 如果客户端通过使用 APIs 从一台服务器中读取一个本地列的值和/或标志，那么，服务器将（从基本属性 ‘序列_表示’ 中）确定它们是否存储在外部组件文件中。如果是，服务器将存取 AoExternalComponent 相应的实例。

服务器将根据语法 “\$ (<root_path_symbol>) <remainder>” 分解 ‘文件名_url’ (filename_url) 的值，该值是一个字符串。分解成两个字符串：root_path_symbol 和剩余。字符串 <root_path_symbol> 是一个语境变量的名称。其值是该外部组件文件的根路径。因此，服务器将串联根路径与剩余，组成保存取值的文件的绝对路径。

同样，服务器也根据语法 “\$ (<root_path_symbol>) <remainder>” 分解 ‘标志_文件名_url’ (flags_filename_url) 的值，该值也是一个字符串。分解成两个字符串：root_path_symbol 和剩余。字符串 <root_path_symbol> 是一个语境变量的名称。其值是该外部组件文件的根路径。因此，服务器将串联根路径与剩余，组成保存取值的文件的绝对路径。

服务器读取文件并向用户返回取值和/或标志。

在这种使用情况下，客户端不需要与 AoExternalComponent 的实例交互，也不需要与外部组件文件交互。

例如（针对 URL）：

如果语境变量 DISK_1= ‘file: //HOMER/C: /Files’，filename_url= ‘\$ (DISK_1) /TDATA0.dat’，文件的绝对路径是 ‘file: //HOMER/C: /Files TDATA0.dat’，这表示，文件 TDATA0.dat 位于主机 HOMER 的驱动器 C: 的目录/Files/中。

7) 如果客户端通过提供 ODS-兼容的外部组件文件将一个本地列的值和/或标志写入服务器中，那么，客户端将创建一个恰当的 AoExternalComponent 实例，构建本地列的一个关系，并相应地设置 AoExternalComponent 的实例的属性，以便它们能够反映内部结构。客户端将拷贝外部组件文件到服务器驱动器上的默认根路径（由语境变量指定，该语境变量的名称由文件_根 (FILE_ROOT) 赋予）内的一个位置上。

客户端将分解保存取值的文件的位置为默认根路径和一个剩余。该剩余位于一个路径

信息的前面，该路径被指定为字符串“\$(<root_path_symbol>)”。此处，<root_path_symbol>是 FILE_ROOT 提供的字符串（注意：这是语境变量的名称，而不是根路径本身）。根路径的信息与剩余串联在一起，并作为相对路径存储在‘AoExternalComponent 的实例的‘文件名_url’（filename_url）中。

客户端同样分解保存标志的文件的位置为默认根路径和剩余；该剩余以同样的方式位于一个默认路径信息的前面，该默认路径被指定为字符串“\$(<root_path_symbol>)”。此处，<root_path_symbol>是 FILE_ROOT 提供的字符串。根路径的信息与剩余串联在一起，并作为相对路径存储在‘AoExternalComponent 的实例的‘标志_文件名_url’（flags_filename_url）中。

这不是一个平常的用例，因为，客户端需要具备关于如何访问默认根路径所在的服务器的驱动器的专业知识，并必须有驱动器的写入存取权限。然而，在特殊情况下，这可以加速将数据存入一台 ODS 服务器的过程，特别是需要处理大量数据的时候，因此，避免了海量数据通过 APIs 传输。

8) 如果客户端通过从一台 ODS 服务器的驱动器中拷贝外部组件文件读取一个本地列的取值和/或标志，那么，客户端首先必须（从基本属性‘序列_表示’中）确定它们是否真的存储在外部组件文件中。如果是，那么，客户端将会存取 AoExternalComponent 的相应的实例。

客户端将根据语法“\$(<root_path_symbol>) <remainder>”分解‘文件名_url’（filename_url）的值，该值是一个字符串。分解成两个字符串：root_path_symbol 和剩余。字符串<root_path_symbol>是一个语境变量的名称。其值是该外部组件文件的根路径。因此，客户端将串联根路径与剩余，组成保存取值的文件的绝对路径。

同样，客户端也根据语法“\$(<root_path_symbol>) <remainder>”分解‘标志_文件名_url’（flags_filename_url）的值，该值也是一个字符串。分解成两个字符串：root_path_symbol 和剩余。字符串<root_path_symbol>是一个语境变量的名称。其值是该外部组件文件的根路径。因此，客户端将串联根路径与剩余，组成保存标志的文件的绝对路径。

然后，客户端将会拷贝这些文件到其自己的存储域，并可以根据内部结构从 AoExternalComponent 的实例读取的方式处理它们。

这也不是一个平常的用例，因为，客户端需要具备关于如何访问保存确定的根路径的服务器的驱动器的专业知识，并必须有读取该驱动器的存取权限。然而，在特殊情况下，这可以加速从一台 ODS 服务器加载数据的过程，特别是需要处理大量已经存储在外部组件文件中的数据的时候，因此，避免了海量数据通过 APIs 传输。

9) 如果客户端链接一个外部引用文件到 ODS 服务器中的一个实例（通过指定基本属性‘外部_引用’（‘external_references’）派生的一个应用属性的值），那么，客户端将会拷贝该文件到 ODS 服务器的默认根路径（由语境变量指定，而该语境变量的名称由‘文件_根_外部引用’（‘FILE_ROOT_EXTREF’）指定）内的一个位置。

然后，客户端将分解外部应用文件的位置为默认根路径和一个剩余，该剩余位于一个根路径信息的前面，该根路径被指定为字符串“\$(<root_path_symbol>)”。此处，<root_path_symbol>是 FILE_ROOT_EXTREF 提供的字符串（注意：这是语境变量的名称，而不是根路径本身）。根路径的信息与剩余串联在一起，并作为相对路径存储在‘位置’中。

客户端必须具有在默认根路径内写入位置的存取权限。

ODS 服务器不会使用存储在‘位置’的信息，也不会存取外部引用文件。它只向客户端提供关于默认根路径的信息，并允许客户端设置‘位置’的值。

10) 如果客户端想要存取一个链接 ODS 服务器中的一个实例的外部引用文件，它将根据语法“\$(<root_path_symbol>)<remainder>”分解‘位置’的值，该值是一个字符串。分解为两个字符串：root_path_symbol 和剩余。字符串<root_path_symbol><remainder>是一个语境变量的名称。其值是该外部引用文件的根路径。因此，客户端将串联该根路径与剩余，组成外部引用文件的绝对路径。

客户端必须具有读取该位置的存取权限。

ODS 服务器不会使用存储在‘位置’的信息，也不会存取外部引用文件。它只向客户端提供关于根路径和‘位置’的信息。

例如（针对 UNC_WIN）：

在服务器包含以下符号名称列表的情况下：‘DISK_1’，‘DISK_2’，‘DISK_3’，DISK_2=‘S: \RootDir’，服务器返回的一个外部引用的位置值为“\$(DISK_2)\TestData\File1.dat”，外部引用的最终位置将是“S: \RootDir\TestData\File1.dat”。

可以在客户端使用下列 Java 示例代码确定外部引用文件的绝对路径：

```
/**
 * Get the UNC absolute filename for an external reference.
 * @param location the location as stored in the ODS server
 * @param aoSess the session to the server
 * @return the UNC filename.
 * @throw AoException any occuring ASAM ODS exception
 */
public String getUNCFilename (String location, AoSession aoSess) throws
    AoException {
    String filename = location;
    // Get the begin of the symbol, must be 0
    int indexStart = location.indexOf("$ (");
    if (indexStart == 0) {
        //Get the end of the symbol
        int indexEnd = location.indexOf(")");
        if (indexEnd > indexStart) {
            //Get the symbol.
```

```

        String sym = location.substring(indexStart+2, indexEnd);
        //Get the context variable of the symbol
        String rootpath = aoSess.getContectByName(sym).value.u.stringVal
    ( );

        filename = rootpath + location.substring(indexEnd+1);
    }
}
return (filename);
}

```

5.11 与其他 ASAM 标准的兼容性

在 ASAM 标准体系中, ASAM ODS 规范适用于长期保存汽车领域自动化及测量系统的所有类型的信息。

然而由于各种原因, 该领域的永久存储标准也被 ASAM 内部和外部的其他几个机构制定。本节说明 ASAM ODS 与这些相关标准的关系。

5.11.1 经协调一致的 ASAM 数据类型

ASAM 已确定了一个所有 ASAM 标准共同使用的数据类型集合。这会便于由一个以上的 ASAM 标准集成的系统的实施, 在数据一系列的 ASAM 接口传递时, 保持数据丢失的可能性尽量小。

统一数据类型在一个单独的 ASAM 标准中规定, 可以向 ASAM 办公室获取。

5.11.2 VSIM 车辆安全测试标准

VSIM (车辆安全信息模型) 规范源自 ASAM ODS 工作组。由于其对文档产品开发的重要性, 创建和维护已经转到一个国际标准化组织 (ISO) 工作组 (ISO TC22/SC12/WG3), 并已经发布为 ISO/TS 22240。它符合 ASAM ODS, 虽然不是每发布一次新的 ODS 就发布一次新的 VSIM。规范和工具版本必须匹配存储兼容性。

5.11.3 MDF 存储 ECU 标定系统的测量数据

MDF (测量数据格式) 规范从传动系统开发领域演化而来。它被用来存储来自传感器和电子控制单元 (ECU) 的测量数据已久, 并经历了几次重大修改。从版本 4.0.0 开始, MDF 规范变成 ASAM 标准, 并与 ASAM ODS 合并开发。ASAM ODS 的参与保证了 MDF 和 ASAM ODS 尽量互相兼容。

本节将提供一组应该遵循的建议和规则, 以便于 MDF 文件能够在 ASAM ODS 环境使用。理解和遵从这些建议要求熟悉 ASAM ODS 和 MDF 规范。

在一个 ASAM ODS 环境中, 测量数据可以直接保存在 ODS 存储器中。另外, 也可以保存在 ODS 存储器引用的外部文件中。第二个可能性被称为“混合模式”, 建议使用混合模式的方式存储大量数据。

在混合模式中, ODS 存储器可以为每个数据项目引用一个或多个含义测量数据的二进制文件。测量文件被称为“外部组件”。此外, ODS 存储器必须包含二进制文件中的记录布局的说明。

将测量数据导入 ODS 环境时, 测量的相关元信息将会直接保存在 ODS 存储器中。外部组件也有可能包含相同的元信息。这种冗余在信息交换中是接受的, 以更快地存储信息, 例如, 搜索操作。

为了在测量数据系统和 ODS 环境中交换测量数据, 通常使用 ASAM ATF/XML 文件格式。类似于 ODS

存储器，一个 ATF/XML 文件可以在相同的文件（XML 格式）内直接存储纯测量值，也可以引用外部组件文件。测量的元信息可以存储在 ATF/XML 文件的标题。

将一个 ATF/XML 文件导入 ODS 环境时，要将元信息和直接存储的测量值拷贝到 ODS 存储器中。可以提取外部组件传输的测量数据存储在 ODS 存储器中，或者，对于混合模式服务器，外部组件也可以存储在文件系统并由 ODS 存储器引用。

除了 ATF/XML 文件之外，测量数据也可以存储在一个 MDF 文件中。有两种备选方案可以将一个 MDF 文件导入 ODS 环境：

- a) 直接导入 MDF 文件，或；
- b) 创建一个引用 MDF 文件作为外部组件的 ATF/XML 文件，并导入该 ATF/XML 文件。

在两种情况下，创建 ATF/XML 文件或将 ATF/XML 文件导入 ODS 服务器的应用必须识别 MDF 格式，而且必须能够从 MDF 文件中提取 ODS 相关元信息。

在任一情况下，需要保存 MDF 文件，并直接将 MDF 文件作为外部组件使用（而不是将测量数据传输到其他某种二进制格式，因而，有可能丢失包含在 MDF 文件中的附加信息）。

建议：使用一个引用 MDF 文件作为外部组件的 ATF/XML 文件时，MDF 文件也可以通过使用一个附件块（ATBLOCK）引用 ATF/XML。因此，基于 ATF/XML 的工具和基于 MDF 的工具都可以看到两个文件的联盟。

5.11.3.1 MDF 文件作为外部组件的集成规则

为了将一个 MDF 文件作为外部组件使用，必须满足一些要求：

- (1) MDF 文件必须排序；
- (2) MDF 文件中的所有通道必须使用线性转换，或全部不用转换；
- (3) MDF 文件中的所有通道的数据类型必须符合 ODS 支持的数据类型（括号里的是 MDF 数据类型编号）：
 - a) 无符号整数数据类型（cn_data_type=0，或 1；LEO，BEO）只可以使用 cn_bit_count=8，16，32 位；
 - b) 有符号整数数据类型（cn_data_type=2，或 3；LEO，BEO）只可以使用 cn_bit_count=8，16，32 或 64 位；
 - c) 浮点数据类型（cn_data_type=4，或 5；LEO，BEO）只可以使用 cn_bit_count=32 或 64 位；
 - d) 字符串数据必须是单字节字符（SBC）ISO/IEC 8859-1 Latin（cn_data_type=6）。
ODS 不支持 UTF-8 和 UTF-16 字符串类型。字符串数据必须存储在 MDF SD-Block（安全数字-块）。不支持没有 SD-Block（安全数字-块）的列表；
 - e) 一般来说，只有 SD-Block（安全数字-安全）支持复杂的数据类型（cn_data_type>=10）。
不支持 SD-Block（安全数字-安全）列表。复杂的数据类型总是被解释为一个字节流；
 - f) ODS 不支持位信号。
- (4) 所有通道的起始位（cn_bit_offset）必须是 0，例如，信号值必须是对齐的 1 字节；
- (5) ODS 不解释无效位；

(6) 数据块列表 (DTBLOCKs) 可以由外部组件处理, 但是, 如果有大量块, 将会导致外部组件表中的管理总开销, 影响性能。在后处理步骤, 可以压缩含义多个数据模型 (DTBLOCKs) 的一个列表成一个只含一个数据模型 (DTBLOCKs) 的列表;

(7) 结构分成子组件, 作为测量量处理。必须忽略表示整个结构的通道;

(8) n 维数组, 即存储的 en-block (CN-模板), 可以通过使用秩和维度属性映射到测量量。

如果原始 MDF 文件不符合这些规则, 有可能通过后处理的 MDF 文件来实现外部组件的集成。这个后处理的实施建议列出在本节的下一段。

注意, 在一些情况下, 后处理可能非常复杂或不可能实现, 例如, 如果在 MDF 文件中, 一个通道采用非线性转换规则, 或采用无法转换为 ODS 兼容的数据类型的一个数据类型。如果这样一个通道只有几个样本, 最简便的解决方案将是直接将其物理值保存在 ODS 存储器。

另外, 可以采用 ODS 兼容格式将物理值存储在一个二进制文件中。然后, ODS 存储器可以引用该二进制文件作为各个通道的外部组件。

5.11.3.2 MDF 文件的元信息

一个 MDF 文件可以传输自定义的元信息。这也可以用来存储 ODS 相关信息, 例如, 在一个测量数据块 (MDBLOCK), 采用 XML 自定义标签。应用负责导入, 然后, 必须识别如何读取已经存储在 MDF 文件的 ODS 相关元信息。

5.11.3.3 可能实现 ODS 兼容 MDF 文件的后处理步骤

对于已经存在的 MDF 文件, 或由不遵循上述规则的应用创建的 MDF 文件, 可以进行后处理, 实现 ODS 与 MDF 文件兼容。针对每个规则实现兼容而采取的措施列出如下。

注意: 对于 ODS 兼容, MDF 文件排序是必要的。下面列出的所有检查和后处理步骤可以在 MDF 排序后立即完成, 也可以以后再完成。

(1) 保证对 MDF 文件进行排序。(每个 DGBLOCK 包含一个单一的 CGBLOCK);

(2) 检查所有通道的转换规则;

如果一个通道采用非线性转换, 那么, 采用某个 ODS 兼容数据类型将该通道的转换的(物理)值存储在一个额外的二进制文件中。如果只有几个值, 那么, 考虑直接导入到 ODS 存储器中, 或, 创建一个 ATF/XML 文件, 将这几个值存储在相应的 XML 标签, 直接存入该文件。

(3) 检查所有通道的数据类型

对于采用几个不兼容 ODS 的位的整数或浮点数据类型, 检查是否可以增加位的数量至下一个允许的更高的数量。然而, 这要求通道组所有记录有一个新的布局, 并处理一次完整的数据块。由于数据块的规模会增加, 通常情况下, 有必要重新安排 MDF 块, 或至少移动数据块至文件结尾, 可能会留下一个缺口。

如果不需要改变记录布局, 或, 如果一个通道的一个数据类型无法加大或传输以实现 ODS 兼容, 那么, 其数据冗余地存储在 ODS 存储器中。

d.& e.: 检查每个带有可变长度信息值的通道

信号值必须无缝存储在安全数字块 (SDBLOCK) 中, 对于字符串, 终止空字符必须是最后个字节。如果不是这样, 可以删除不必要的字节, 压缩安全数字块 (SDBLOCK)。由于安全数

字块（SDBLOCK）的大小将会缩小，它可以存储在与之前相同的起始地址中。这同样要求调整引用该通道的所有普通记录的安全数字块（SDBLOCK）数据的结构的偏移和长度。如果有多个安全数字块（SDBLOCKS）的一个列表，必须压缩为只含一个安全数字块（SDBLOCK）的列表。

（4）检查所有通道的起始位

如果一个通道不采用 0 作为起始位，那么，可能要改变记录布局，类似第 3 步，或，可以将数据冗余地存储在 ODS 存储器中。

（5）如果使用了无效位，那么，必须创建一个附加文件，该附加文件包含相关的标志，并附在 ODS 存储器上。

（6）可以压缩含义多个数据块（DTBLOCKs）的列表至只含一个数据块（DTBLOCKs）的列表，以提高性能。

（7）没有具体的后处理操作。

6 物理存储

6.1 概述

6.1.1 简介

本章将描述如何在符合 ASAM ODS 规范的关系数据库中（物理）存储测量数据及结果。

为了对 ASAM ODS 信息进行物理存储，除了保存结果数据的表，还要使用几个表。这些表叫做 SVCxxx，其中 xxx 描述表的类型。这些表是：

- SVCENT（存储应用元素（元模型））；
- SVCATTR（存储应用属性（元模型））；
- SVCREF（存储引用（元模型））；
- SVCVAL（存储值（子矩阵代表局部列））；
- SVCINST（存储实例属性）；
- SVCENUM（存储枚举）；
- SVCACLI（存储实例保护安全数据）；
- SVCACLA（存储元素和属性保护安全数据）；
- SVCTPLI（存储 ACL 模版）；
- SVCVAL_SPS（存储单点值（子矩阵代表局部列））。

下几节将详细介绍这些表。这些节的脚本说明采用 Oracle 数据库系统 7.3 版和更高版本。可供其他数据库系统参考。

接下来一节将描述如何存储应用属性值，尤其是当它们不能立即放进相应表的时候。

最后一节将解释所谓“混合模式”下的存储机制，在混合模式下，大量的数据被存储在外部，而大多数应用模型信息被保存在关系数据库中。

6.1.2 常用的数据类型

ASAM 已经制定了标准化的 ASAM 数据类型，这些数据类型已经在 2002 年末公布。这些数据类型有标准化的名称，总是以 A_ 开头。ASAM ODS 使用的这些数据类型将与今后制定的所有新规范一致。

许多年前，基于关系数据库的物理存储已经开始开发和明确。那个时候不存在 ASAM 级别的标准数据类型。因此，ASAM ODS 决定制定满足 ASAM ODS 需要的数据类型。这些数据类型总是以 T_（如果是单值/结构类型）或 S_（如果是单值/结构类型的序列相关数组）开头。

本章描述的物理存储已经在 ODS 服务器上实施，并已经在大量公司计算机上进行了安装。用 A_ 类型替代 T_ 类型就必须得修改和改变已经安装的服务器和底层数据库设计，并且还要修改访问服务器的客户端。

这就是为什么 ASAM ODS 决定不在物理存储说明中引入标准 ASAM 数据类型（A_）。

相反，ASAM ODS 规范的第 6 章介绍了 T_ 类型和 A_ 类型的关系。应该注意到，数据类型的枚举以及它们的枚举名称和 ASAM 数据类型规范中确定完全相同，而且在大多数情况下，这些数据类型本身就是二进制兼容的。所以，如果需要的话，可以实现一对一的映射。

6.2 SVC 表的描述

6.2.1 SVCENT

本表存储不同应用元素的描述。

在 Oracle 数据库服务器中创建 SVCENT 表的语法是：

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

AID 列是应用元素的标识符；它在整个 ASAM ODS 物理存储中必须是唯一的，通常由创建应用程序模型的 ASAM ODS 服务器给出。Oracle 数据库服务器用一个 Oracle 序列号决定下一个可用 ID，序列号的名称（计算算法）是 DBTNAME+ID_SEQ。

ANAME 列是应用元素的名称，由应用模型的创建者确定。在一个应用模型中应用元素的名称必须是唯一的。该规范限制应用元素名称的长度和数据库表名称的长度最多为 30 个字符。当使用 Oracle 数据库服务器时，较灵活的 'varchar2(30)' 会代替 'char(30)'。为了和 ASAM ODS 早期版本兼容，ASAM ODS 服务器必须支持两种类型。

BID 列是基础元素的唯一标识符，基础元素派生出应用元素。ASAM ODS 已经将 BID 列标准化，如表 19 所示。

表19 基础元素 IDs (BID)

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

BTNAME 列指定了包括应用元素所有实例的表名。表名只有 ASAM ODS 服务器知道；它不会通过任何借口提供给客户端。一个完整的表名长度一定不能超过数据库允许的表名最大长度。

对于阵列表，命名规则为“<Table name>_ARRAY”。在这种情况下，意味着，例如，对于 Oracle 数据库，<表名>的长度不能超过 24 个字符，这是最长的表名长度（30 个字符）减去“_ARRAY”（6 个字符）后的结果。

SECURITY 列保存应用元素安全等级信息。可能的安全等级是“元素安全”，‘实例安全’和‘属性安全’或是它们的组合。它们中的每一个都通过一个位码来控制（0=无效，1 = 有效）。

ACL 类型位码见表 20。

表20 ACL 类型的位码

位元#	十进制	配置
0	1	元素安全：应用元素的 ACL 条目（通常为应用元素的所有实例设置访问权限）
1	2	实例安全：应用元素实例的 ACL 条目（通常为每个实例单独设置访问权限）
2	4	属性安全：应用元素属性的 ACL 条目（通常为每个生成单独设置访问权限）

在 SECURITY 列中存储的信息应该和在第四章中定义与描述的实体“安全等级”（基础模型）相符。
有关 ASAM ODS 的安全概念的更多信息在第 6 章中介绍。

SVCENT 表的示例如下。

示例：SVCENT 表。
表21申明三个应用元素和存储三个应用元素相应实例的表名。

表21 SVCENT 表

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

6.2.2 SVCATTR

该表保存对应用属性以及所有应用元素的应用关系描述。

在 Oracle 数据库服务器中创建 SVCATTR 表的语法是：

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

AID 列指定应用属性或关系所属应用元素的唯一标识符。它是 SVCENT 表中一个条目的引用（有相同的标识符 AID）。

ATTRNR 列包括应用属性/关系的编号，它是由创建应用模型的 ASAM ODS 服务器给出的。在相同应用元素的所有应用属性和关系中，这个编号必须唯一。它必须从 0 开始并且连续增加。

应用属性的最大数量受底层数据库限制。AANAME 列保存应用属性/关系的名称，它是由应用模型创

建者指定的。然而， $n:m$ 关系没有在这个表中列出，而是在 SVCREF 表中进行了描述。在一个应用元素中，应用属性/关系的名称必须唯一。本标准限制了应用属性/关系的名称长度最大为 30 个字符。其他的一些字符串长度也有限制。当使用 Oracle 数据库服务器时，较灵活的 'varchar2(30)' 会代替 'char(30)'。ASAM ODS 服务器必须支持和 ASAM ODS 早期版本兼容的两种类型。

BANAME 列包括基本属性/关系的名称，如果应用属性/关系派生自基本属性/关系（也就是一个基本的应用属性）；否则，这个数据库字段为空。具有相同应用元素的两个应用属性不能引用一样 d 的基本属性；然而，对于抽象的超类关系，允许一个以上应用关系引用相同的基本关系。

FAID 列和应用关系以及继承概念有关。对于应用关系，AID 列包含作为应用关系目标的应用元素的标识符。在 SVCENT 表中，AID 列是引用这些条目（它的 AID 和 FAID 值相同）中一个。

对于继承概念，当其他应用元素是超类应用元素时，FAID 列用于指定一个子类应用元素。为此目的，使用派生自基本属性的子类属性的 FAID 列的输入“ID”；

FAID 列将包括相应超类应用元素的 AID。第 6.3.7. 节给了一个例子。

FUNIT 列可能保存应用元素一个实例标识符的引用，该应用元素来派生自基础元素 AoUnit。

ADTYPE 列指定应用属性数据类型的枚举值。对于一个应用关系，ADTYPE 列包括了引用标识符的数据类型。DT_LONG 和 DT_LONGLONG 可能为 $1:n$ 关系而设，DS_LONG 或 DS_LONGLONG 可能为相反关系而设。ASAM ODS 数据类型和相应的枚举值列表将在本节给出。从这些列表来看，与 DT_UNKNOWN, DT_ID, 和 DS_ID 对应的数据类型不得使用。

AFLEN 列指定有字符串，日期，或数据流类型的应用属性的最大长度（以字节计算）（其他数据类型，该列可以为空）。该列最大长度依据数据库存储字符串能力可以设定。ASAM ODS 不支持无限长度字符串。

DBCNAME 列指定最终保存应用属性/关系值的列名。一个应用元素所有实例将一起存储在一个实例表中，这个实例表的名称在 SVCENT>DBTNAME 中给出。应用属性/关系的值（存在一些例外）将会存储在那个表的一个列中，该列的名字由 DBCNAME 指定。一个应用元素中，DBCNAME 的值必须唯一。

该规则还有一些例外，即：

——派生自 AoLocalColumn 基本属性 'values', 'flags' 和 'generation-parameters' 的属性将不会存储在这些实例表的列中，而是存储在单独的 SVCVAL 表中（见 6.2.4 节）或在混合模式服务器下，存储在外部文件中（见 6.4 节）。在这种情况下，DBCNAME 的相应字段所含字符串为“NULL”。

—— $1:n$ 类型关系只能存储在两个相关应用元素之一的实例表中，该实例只和伙伴实例有关系。在这种情况下，其他应用元素（相反关系）的 DBCNAME 列的相应字段所含字符串为“NULL”。

—— $n:m$ 类型关系不会存储在实例表的列中，而是存储在 SVCREF 指定的单独表中（见 6.2.3 节）（它们没有在 SVCATTR 中指定。）

——序列类型或其他的多元素类型的应用属性存储在一个单独的表中，进一步的介绍见 6.3 节。

列名 DBCNAME 只有 ASAM ODS 服务器知道；它不会通过任何接口提供给客户端。本标准限制了列名的最大长度为 30 个字符。

当应用元素声明“实例安全性”（通过 SVCENT 表中的 SECURITY 字段）时 ACLREF 列才能使用。它可能仅为了指定 $1:n$ 类型的应用关系（不是相反关系）的表的条目而设定，并且包括一个标记，该标记表明应用关系是 (=1) 否 (=0) 应该通过解析寻找 ACL 模版。在没有特定的 ACL 条目提供给实例的

情况下，服务器用 ACLREF 决定应该把什么样的 ACL 条目附加到一个即将创建的实例上。

存储在 ACLREF 列中的信息和实体'InitialRightsAt'是相对应，将在第 7 章（基础模型）对实体'InitialRightsAt'进行定义和描述。

有关 ASAM ODS 安全概念的更多信息可以在第 6 章找到。

INVNAME 列仅仅和应用程序关系有关，并且保存相关关系的名称。在 ODS 模型（基础模型和应用模型）中，所有的关系都有相反关系，所以可以采用相反名称。相反名称和相关应用元素的应用关系的名称（AANAME）相同；它可能和基础模型里的名称不同。针对应用属性，INVNAME 列保持为空。

FLAG 列包括三个控制应用属性/关系内容的标记，这三个标记，将在本节的后面对其进行详细解释。

当应用属性的数据类型 ADTYPE 是 DT_ENUM（=30）或 DS_ENUM（=31）时，ENUMNAME 列保存枚举的名称。本标准限制了枚举名称的最大长度为 30 个字符。

枚举是一堆的固定名称值对，它通常通过字符串表述，让某个数据项为人所理解，但计算机将会使用数值表述，因为其存储空间占用小、比较操作速度快。

所有枚举项都存储在 SVCENUM 中；属于相同枚举的项会有相同的 ENUMNAME 标识。（见 6.2.6 节）

6.2.2.1 应用属性的标记

在 ASAM ODS 模型中，应用属性都有标记，如 UNIQUE 或 OBLIGATORY。这些标记将以位模式存储在 FLAG 列中。

表 22 列出了产生列值的位以及它们各自代表的含义。

表22 应用属性标记的位码

位	含义
0	（UNIQUE） 这个标记告诉服务器这个属性的值必须唯一。 1表明该值必须唯一。服务器能够通过数据库约束来保证该值唯一。
1	（OBLIGATORY） 这个标记告诉服务器这个属性的值必填。 1表明该值必须设置。服务器能够通过数据库约束来确保此值必填。 （AUTOGENERATE） 这个标记告诉服务器这个属性值必须通过服务器产生，如Id。
2	1表明属性值将由服务器自动产生。对于每个自动生成的列，需要在数据库或服务器里手动定义触发器，并且必须指定一个合适的算法来生成值。此行为依赖于数据库，不受ASAM ODS客户端或服务器控制。

6.2.2.2 ADTYPE 列的数据类型编码

ADTYPE 列指定应用属性值在数据库里被编码的数据类型。与将数据类型指定为一个字符串相反，它被赋予数字（代表它的数据类型枚举值）。枚举和数据类型本身之间的联系将在下表中介绍，在表 23 中，T_XXX 是基本的数据类型，S_XXX 是基本的数据类型的序列，DT_XXX（DS_XXX 序列）是数据类型枚举的名称。

表23 ASAM ODS 数据类型

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

数据类型的定义和 ASAM ODS 4.1 是相符的。在 RPC-API 中也提供数据类型定义。该数据类型的值和 AoMeasurementQuantity 中的属性 “datatype” 以及 AoQuantity 中的属性 “default_datatype” 是相同的。

6.2.2.3 SVCATTR 表的举例

为了阐明 SVCATTR 表的规范，下面给了一个例子。

示例：SVCATTR 表的举例。

表 24 申明了 AID=1 的应用元素的三个应用属性和 AID=3 的应用元素的三个应用属性。

Id 属性的 HeadId 和 TestId 的值是由服务器产生的，而且值必须唯一，所以属性 HeadId 和 TestId 的 FLAG 值设为 5（0101）。

表24 SVCATTR 表的举例

AID	ATTRNR	AANAME	BANAME	FAID	FUNIT	ADTYPE	AFLEN	DBCNAME	ACLREF	INVNAME	FLAG	ENUM NAME
1	1	HeadId	id			8		HEAD_INDEX			5	
1	2	SerialNo	name			1	32	PART_ID				
1	3	Head_rel		3		8		TEST_INDEX		TestObj_rel		
3	1	TestId	id			8		TEST_INDEX			5	
3	2	TestObj_rel		1		8		HEAD_INDEX		Head_rel		
3	3	State				1	32	TEST_STATUS				

6.2.3 SVCREF

该表保存应用元素之间的 n: m 关系的描述，在 Oracle 数据库服务器中创建 SVCREF 表的语法是：

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

AID1 列包括第一个应用元素的标识符（应用关系的起点）。它是引用 SVCENT 表中一个条目（具有相同标识符 AID）。

AID2 列包括第二个应用元素的标识符（应用关系的终点）。它是引用 SVCENT 表中其中一个条目（具有相同标识符 AID）。

REFNAME 列包括由应用模型的创建者指定的应用关系名称。本标准限制了关系名称的最大长度为 80 个字符。此外，其他字符串的长度也有限制。使用 Oracle 数据库服务器时，较灵活的 'varchar2(80)' 会代替 'char(…)'。ASAM ODS 服务器必须支持和 ASAM ODS 早期版本兼容的两种类型。

两个相同应用元素之间的应用关系名称必须唯一。此外，应用关系名称和 SVCATTR 表中指定的由 AID1 引用的应用元素的应用属性/关系的名称不同。

DBTNAME 列指定了表的名称，而实例之间的关系最终存储在这些表中；在 SVCREF 表中上述表的名称必须唯一，而且不同于 ASAM ODS 物理存储中的其他表的名称。本标准限制了表的名称的最大长度为 30 个字符。表的结构将在本节的后面进行说明。只有 ASAM ODS 服务器知道表的名称；表的名称不通过任何接口提供给客户端。

INVNAME 列包括相反关系的名称。相反名称和表中相反的 n: m 关系的应用关系名称（REFNAME）相同（可能和基础模型的关系名称不同）。本标准限制了相反关系的名称的最大长度为 80 个字符。

如果应用关系派生自基本关系（即基本应用关系），BANAME 列保存基本关系的名称；否则，本数据

库字段为空。

如果应用关系派生自基本关系（即基本应用关系），INVBANAME 列保存相反基本关系的名称；否则，本数据库字段为空。

在该表中，AID1、AID2、REFNAME 的组合必须唯一。只要两个应用元素名字不同，允许相同的两个应用元素之间存在一个以上 n： m 关系。

在 SVCREF 表中，每个 m： n 关系只能由一个条目表示。

6.2.3.1 实例之间保存 N： M 关系的表

保存实例之间 N： M 关系（在 DBTNAME 中已经给出了它的名称）的表有三个列：

——因为元素基本属性“ID”由 SVCREF 表 AID1 列应用元素 ID 给出，第一列名称必须与 SVCATTR 表相应条目的 DBCNAME 列值一样；

——因为元素基本属性“ID”由 SVCREF 表 AID2 列应用元素 ID 给出，第二列名称必须与 SVCATTR 表相应条目的 DBCNAME 列值一样；

——第三个列的名称必须是 REFNAME。

示例：最终保存关系的表的举例。

假设表 SVCENT, SVCATTR 和 SVCREF 包含下表给出的条目，最后保存实例之间关系的这个表采用“UNITTOGROUPS”的名称，具体见表 25。

表25 最终保存关系的表的举例

SVCENT :												
AID		ANAME		BID		DBTNAME		SECURITY				
3		Unit		---		---		0				
4		UnitGroup		---		---		0				
---		---		---		---		0				
SVCATTR :												
AID	ATTRNR	AANAME	BANAME	FAID	FUNIT	ADTYPE	AFLN	DBCNAME	ACLREF	INVNAME	FLAG	ENUM NAME
3	1	Unitid	id			8		UID			5	
3	2	Unit	name			1	32	UNIT				
4	1	UnitGroupId	Id			8		UGID			5	
4	2	UnitGroup	name			1	32	UNITGROUP				
---	---	---	---	---	---	---	---	---	---	---	---	---
SVCREF :												
AID1	AID2	REFNAME		DBTNAME		INVNAME		BANAME		INVBANAME		
3	4	Unit_to_Groups		UNITTOGROUPS		---		---		---		
---	---	---		---		---		---		---		
UNITTOGROUPS :												
UID				UGID				REFNAME				
ID of instance of Unit				ID of instance of UnitGroup				Unit_to_Groups				
ID of instance of Unit				ID of instance of UnitGroup				Unit_to_Groups				
---				---				---				

6.2.4 SVCVAL

本表保存测量数据和数据的值标记。

在 Oracle 数据库服务器中创建 SVCVAL 表的语法如下：

```
create table svcval
(
  MEQID integer NOT NULL, /* ID of measurement quantity */
  PMATNUM integer NOT NULL, /* ID/number of submatrix */
  SEGNUM integer NOT NULL, /* Number of segment */
  VALINDEP integer NOT NULL, /* Independent flag */
  VALEXIMP integer NOT NULL, /* Implicit/explicit flag */
  VALBLOBLN integer NOT NULL, /* Length of the blob */
  VALBLOB long raw NOT NULL /* Data of the column */
);
```

该程序中的每一行都存储了一组值；它们最后被保存在 VALBLOB 字段中。

MEQID 列指定是测量数量的 id，测量数量值存储在 SVCVAL 表的行中，它是引用派生自基础元素 AoMeasurementQuantity 的应用元素的一个实例。

PMATNUM 列保存测量值所属的子矩阵的标识符。它是引用派生自基础元素 AoSubmatrix 的应用元素的一个实例。

SEGNUM 列表示分割段数。由于一些数据库的限制，VALBLOB 的最大尺寸被限制为 10000 字节。

如果将要被存储的一组测试值超过这个尺寸，它们将要被分割并存储在单独的 VALBLOBs 中。在这种情况下，SEGNUM 将从 1 开始，并随着以后的每次分割而增长。即使 VALBLOB 的尺寸限制没有达到，分割也可能会用于测量值。

VALINDEP 列保存单独的标记。它的值必须和派生自相应的局部列的基本属性 “independent” 的属性值相同，其定义如下：

- 1 = 局部列独立于其子矩阵（即设定值、范围、时间轴）；
- 0 = 局部列和其子矩阵相关。

VALEXIMP 列保存一个标记，该标记表明这些值是应该明确地存储还是自动由生成算法给出。它的值必须和派生自相应的局部列基本属性 “sequence_representation” 的属性值相同。其定义如下：

- 1 = 局部列有隐式定义；
- 0 = 本地列有明确定义。

隐式局部列的更多信息将会在本节给出。

VALBLOBLN 列指定存储在 VALBLOB 中的值的数量。值的数据类型将会在派生自相应的测量数量的基本属性 “datatype” 的属性中给出。

VALBLOB 列包含将和可选标记一起存储的值。VALBLOB 的大小取决于值的数据类型、VALBLOBLN 和标记的存在。VALBLOB 是 blob（二进制对象），结构见表 26。

表26 VALBLOB 的结构

值	差值（假定的）	标记（假定的）

在 VALBLOB 中只有值和标记会被存储。

VALUES 和 FLAGS 的字节排列顺序和服务器的字节排列顺序相同。

在 Oracle 数据库中，LENGTH 字段必须在 VALUES 字段之前插入。LENGTH 是一个 4 个字节的字段，该字段给出了 VALBLOB 中的字节实数（然而 VALBLOBLN 只给出了数据项的编号）；LENGTH 元素本身不能被计数。分割段的最大长度总是限制为 10000；而 $LENGTH \leq 10000$ 。

示例：

```
VALBLOBLN = 2500
Data Type= DT_FLOAT (4 byte)
→ LENGTH = 2500 * 4 = 10.000
```

ALBLOB 的这种结构也可用于数据类型字符串的值。字符串可以采用带分隔符“\0”的序列方式存储。例子：“Hello\0Peter\0Test\0”其中 LENGTH=17，VALBLOBLN=3。

如果 LENGTH 比值所需的字节数还大，那么标志被定义为局部列，并存储在 VALBLOB 中。LENGTH 和值所需的字节数之间的差是标志的字节数。一个标记由两个字节（一个短的整数值）表示。该标志用来储存单个数据值的状态信息；每个状态信息被编码为 16-位标记中的一个位，表 27 给出了各个位的意思。一般而言，所有的位都有设定（因此提供了一个 0x000F 值）。

表27 值标记位码

缩写	值	描述
AO_VF_VALID	0x0001	值有效
AO_VF_VISIBLE	0x0002	值必须直观化
AO_VF_UNMODIFIED	0x0004	值不能被修改
AO_VF_DEFINED	0x0008	值应该被定义.如果一个值矩阵中的值是不可用的,那么将不设制位

如果 LENGTH 比值和标记所需的字节数的总和还大，那将会在值和标记之间有一差值。值和标记的序列在元素值相关的标记是左对齐的。差值的内容是不相关的。

示例：2.500 带标记的浮点值。

第一个 blob：

1666 值 + 1666 标记 = 值为 6664 字节+ 标记为 3332 字节。

如果 LENGTH 设置为 10.000 字节，差值将是 4 个字节。

如果 LENGTH 设置为 9996 字节的实际长度，将没有差值。

第二个 blob：

834 值 + 834 标记（2500-1666）。。

LENGTH = 5004 字节：结果没有差值。

如果 LENGTH 是从 5006 字节到 10.000 字节：将有差值。。

ASAM ODS 服务器必须支持阅读带有 GAPs 的 blob，但是由服务器的制造商决定服务器是否选用带有 GAPs 的 blob 写入。

在本表中，MEQID、PMATNUM 和 SEGNUM 的组合必须唯一。

第 5 章会说明局部列，子矩阵和测量矩阵的联系。子矩阵的重要性是它不包括差值，并且它的每个局部列的 nth 值都彼此相关。因此，只要当它们有相同数目的值并彼此相关（例如，以相同的速度同时采样），那么在相同的子矩阵中测量量就会保持它们的值。

局部列的名称必须和相应的测量量的名称相同。在一个子矩阵中没有两个局部列可参照相同的测量量。因此，MEQID 和 PMATNUM 的组合也是对相应的局部列的间接唯一引用。所以，SVCVAL 表中的每一行都与局部列直接相关。

另一方面，局部列可以用 SVCVAL 中的多行表示。在局部列的值分割成若干段（因为 BLOB 的大小限制）的情况下，尤其如此。

当相同的测量量以不同的速度计算时，这种情况也会发生。

随后，它的值将会在不同的子矩阵之间被分割，每个子矩阵都有一个与相同的测量量相关的局部列；最终测量值将以每个测量速度存储在 SVCVAL 中的不同行中，通过 MEQID（测量量）和 PMATNUM（子矩阵）的组合区分。

6.2.4.1 显式和隐式局部列及生成算法

隐式标记 VALEXIMP 说明存储在 VALBLOB 中值的含义。

通常值是明确存储的（例如，每个样本都作为一个值被存储）。缺省状况下，VALEXIMP=0。这种情况下 VALBLOB 中的元素 VALUES 明确地包括测量量的（一个分割段中）所有值：

$VALUES = \{x_1, x_2, x_3, \dots, x_n, \dots, x_N\}$

派生自相应局部列基本属性“sequence_representation”的属性必须和这些信息相符，因此，应该设置‘explicit’（=0）。

对于一些数据（例如，时间驱动抽样上的时间戳），每一行的值可以被生成；在 VALEXIMP=1 的情况下，只有生成算法需要确定，这样能节省大量存储空间。

这种情况可以分为以下两种子情况：

a) 纯隐式生成：

在这种子情况下，VALBLOB 中的 VALUES 元素只包括生成算法所需的一组生成参数：

$VALUES = \{p_1, p_2, p_3, \dots, p_m, \dots, p_M\}$

所需的参数数目 M 取决于生成算法。参数 $p_1 \dots p_M$ 的数据类型由派生自相应测量量的基本属性“datatype”的属性指定。

表 28 列出了目前提供的隐式值生成的变量。被选中的变量由派生自相应的局部列的基本属性“sequence_representation”的属性指定；它也在表中被指定。终值 X_n 是根据表中给出的公式计算的；N 从 1 开始，以 VALBLOBLLEN 值结束。

表28 纯隐式值生成算法

sequence_representation	M	生成算法
implicit_constant (=1)	1	$x_n = p_1$
implicit_linear (=2)	2	$x_n = p_1 + (n-1) * p_2$

implicit_saw (=3)	3	$x_n = p_1 + ((n-1) \bmod (p_3 - p_1) / p_2) * p_2$

只有当测试数量数据类型是 DT_SHORT, DT_FLOAT, DT_BYTE, DT_LONG, DT_DOUBLE, DT_LONGLONG 时，才允许使用隐式存储。计算始终使用测量量定义的数据类型。然而，表达式 $(p_3 - p_1) / p_2$ 必须被截断为整数，在 P1 处开始每个锯齿曲线循环。

在这种情况下，VALBLOB 不存储任何标记。

b) 原始数据预处理:

在这种子情况下，原始数据值数目 N 和生成参数数目 M 一起被存储起来。元素 VALUES 的结构如下:

$$\text{VALUES} = \{p_1, p_2, p_3, \dots, p_m, \dots, p_M, r_1, r_2, r_3, \dots, r_n, \dots, r_N\}$$

原始数据值数目 N 等于结果值 X_n 的数目， X_n 本身等于派生自相应子矩阵应用元素的基本属性 “number_of_rows” 的属性值。生成参数数目 M 取决于生成算法。生成参数的数据类型和 DT_DOUBLE (每个生成参数使用 8 个字节) 相符。原始数据 r_n 的数据类型是由派生自相应局部列的基本属性 “raw_datatype” 的属性指定。

当所有的生成参数和原始数据可以归入一个分割段时，它们将被连续放入如上所示 VALBLOB 字段的 VALUES 字段中。

如果要存储生成参数和原始数据的所需分割段不只一个，那么第一个分割段代表生成参数 $p_1, p_2, \dots, p_m$ ，紧接着是第一个 N1 原始数据。第二个分割段代表下一组原始数据。结构设定如表 29 所示。

表29 结构设定

分割段 (SEGNUM)	生成参数和原始数据 (VALBLOB)		原始数据数目 (VALBLOBLN)
1	$p_1, p_2, \dots, p_m$	$r_1, r_2, \dots, r_{N1}$	N1
2	$r_{N1+1}, r_{N1+2}, \dots, r_{N2}$		$N2 - N1$
3	$r_{N2+1}, r_{N2+2}, \dots, r_{N3}$		$N3 - N2$
...	...		
k+1	$r_{Nk+1}, r_{N1+2}, \dots, r_N$		$N - N_k$

每个分割段都存储在 SVCVAL 表单独行中。VALBLOBLN 的相应条目是原始数据值各自数目；他们累计到 N。VALBLOBLN 不计算生成参数数目。

表 30 列出了目前为原始数据预处理提供的变量。选中变量由派生自相应局部列的基本属性 “sequence_representation” 的属性指定；选中变量在表中有说明。最终值 X_n 根据表中给出的公式计算；N 从 1 开始，以派生自基本属性 “number_of_rows” 的属性值结束。

表30 原始数据预处理的生成算法

sequence_representation	M	生成算法
raw_linear (=4)	2	$x_n = p_1 + p_2 * r_n$

raw_polynomial (=5)	$p1+2$	$x_n = p2 + p3 \cdot r_n + p4 \cdot r_n^2 + p2 + p1 \cdot r_n^{p1}$
raw_linear_calibrated (=10)	3	$x_n = (p1 + p2 \cdot r_n) \cdot p3$

对于 raw_polynomial, p1 给出了多项式的阶 (从而也间接地给出了生成参数数目)。多项式的阶受物理存储分割段最大值的限制。

只有当测试数量数据类型是 DT_SHORT, DT_FLOAT, DT_BYTE, DT_LONG, DT_DOUBLE, DT_LONGLONG 时, 才允许使用隐式存储。终值 Xn 是服务器提供, 数据类型由派生自相应测量量基本属性“datatype”的属性指定。

6.2.5 SVCINST

该表保存实例属性。实例属性是应用模型中未说明但赋予实例的属性。他们不能保存在保存实例的表中, 因为没有为他们准备相应的表列。相反, 任何应用元素实例的所有实例属性都能一起保存在同一个 SVCINST 表中。

在 Oracle 数据库服务器中创建 SVCINST 表的语法如下:

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

表 31 是 SVCINST 表。

表31 SVCINST 表

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

AID 列包括实例所属的应用元素的标识符, 该实例可以接收实例属性。它是引用 SVCINST 表中一个条目 (具有相同标识符 AID)。

IID 列保存实例本身的标识符, 可以接收实例属性。它是引用相应应用元素实例表中一个实例。

NAME 列保存实例属性的名称。本标准限制实例属性名称的最大长度为 30 个字符。当使用 Oracle 数据库服务器时, 较灵活的 'varchar2(30)' 会代替 'char(30)'。ASAM ODS 服务器必须支持和 ASAM ODS 早期版本兼容的两种类型。

AODT 列保存给出实例属性值数据类型的枚举值。它是依据 6.2.2 节给出的 ASAM ODS “datatype” - 枚举确定。实例属性不支持序列数据类型和复杂的数据类型。

UNITID 列存储实例属性单位的标识符；因此，它是指派生自基础元素 AoUni 的应用元素的一个实例。

如果数据类型是数值型，NUMVAL 列保存实例属性值。无论 AODT 指定哪种数据类型，NUMVAL 列会以最大精度数据类型保存在数据库系统中。因此，在物理存储中，AODT 不指定将要存储的实例属性的数据类型，但是指定必须返回发出请求客户端实例属性的数据类型。支持的数值型的数据类型和 DT_SHORT, DT_FLOAT, DT_BOOLEAN, DT_BYTE, DT_LONG, DT_DOUBLE, DT_LONGLONG（枚举值为 2,...,8）相一致。

如果数据类型是文本，那么 NUMVAL 列保存实例属性值。支持的文本数据类型是 DT_STRING 和 DT_DATE（枚举值是 1 和 10）。本标准限制了文本型实例属性值的最大长度为 255 个字符。

在本表中 AID、IID 和 NAME 的组合必须唯一。

6.2.6 SVCENUM

该表保存枚举数据。

在 Oracle 数据库服务器中创建 SVCENUM 表的语法如下：

```
create table svcenum
(
    ENUMID      integer      NOT NULL, /* Id of the enumeration. */
    ENUMNAME    char(30)    NOT NULL, /* Name of the enumeration. */
    ITEM        integer      NOT NULL, /* Value of the item. */
    ITEMNAME    char(128)   NOT NULL /* Name of the item. */
);
COMMENT ON TABLE svcenum IS 'ASAM ODS enumeration definitions';
```

ENUMID 列保存枚举的标识符，对于每个枚举其值必须唯一。如果一个枚举包括不只一个数据项，那么相同的 ENUMID 值将出现在 SVCENUM 表几行之中。

ENUMNAME 列包括枚举的名称，对于每个枚举它必须唯一。如果一个枚举包括不只一个项，那么相同的 ENUMNAME 值将出现在 SVCENUM 表几行之中。本标准限制了实例属性名称的最大长度为 30 个字符。当使用 Oracle 数据库服务器时，较灵活的'varchar2(..)'会代替'char(..)'。ASAM ODS 服务器必须支持和 ASAM ODS 早期版本兼容的两种类型。

ITEM 列保存枚举项的值。这样做可以相当灵活地指定枚举，尽管枚举项从 0 开始，连续不断增加是很好的做法。在一个枚举之中，ITEM 值必须唯一。

ITEMNAME 列保存便于阅读的枚举项的字符串值。本标准限制了枚举项名称的最大长度为 128 个字符。当使用 Oracle 数据库服务器时，较灵活的'varchar2(128)'会代替'char(128)'。ASAM ODS 服务器必须支持和 ASAM ODS 早期版本兼容的两种类型。在一个枚举之中，ITEMNAME 值必须唯一。

6.2.6.1 ASAMODS 枚举定义

ASAM ODS 数据模型定义了几种枚举，例如 datatype_enum 或 seq_rep_enum。这些枚举可以用于 ASAM ODS 服务器的物理存储。当指定了一个应用模型，才可以定义其他的与应用相关的枚举。

枚举是一堆固定的名称值对，它使用字符串表述让人可以理解当中的数据项。由于存储空间小和比较运算速度快，电脑将使用值表述。

直到现在，在 ASAM ODS 中，枚举由 T_LONG 值表述，但在 ATF 中，枚举以 T_STRING 值存储。下面的例子说明了基础模型给出的枚举 seq_rep_enum。

示例：在 SVCENUM 中枚举的定义见表 32。

表32 SVCENUM 中枚举的定义

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

在基础模型中，项的名称是小写，通常情况下，程序员将枚举项的名称写成大写，但它取决于模型创造者使用大写还是小写。重要的是，ITEMNAME 是区分大小写的。

6.2.6.2 早期版本的物理存储问题

关系数据库物理存储的定义是不向下兼容的。以前版本服务器可以在表 SVCATTR 的 ADTYPE 列中找到 DT_ENUM 或 DS_ENUM 值，但不知道如何处理它们。所以一旦物理存储升级到 ASAM ODS5.0 或更高版本，服务器必须进行转换。

如果服务器能够处理以前版本的物理存储中的枚举，但是无法找到 SVCENUM 表。在使用 SVCENUM 表时，推荐服务器先检查该表是否存在。如果不存在，那么用来存取枚举的 API 方法一旦被调用，必须抛出 AO_NOT_IMPLEMENTED 异常。服务器其余功能必须能正常工作。

如果服务器能够处理以前版本物理存储中的枚举，但是无法找到 SVCATTR 表中的 ENUMNAME 列。即使 SVCATTR 表中的 ENUMNAME 列不存在，根据推荐做法，符合这种条件的服务器应该能正常工作。通过 API 方法对枚举进行的任何读取都必须予以拒绝，并且必须抛出 AO_NOT_IMPLEMENTED 异常。

6.2.7 SVCACLI

该表保存实例保护的安全数据（ASAM ODS 安全概念和存取权限的更多信息见第 5 章）。

在 Oracle 数据库服务器中创建 SVCVALI 表的语法如下：

```
create table svcaccli
(
  USERGROUPID          integer NOT NULL, /* Reference to a UserGroup instance */

```

```

AID          integer NOT NULL, /* Application element id */
IID          integer NOT NULL, /* Application instance id */
RIGHTS       integer NOT NULL /* Rights value (bit masked) */
);

```

此表保存单个实例的访问控制列表（ACL）条目。本表中存储的每个条目（表行）都和第 7 章（基础模型）中定义和描述的实体“ACLI”相符。但是应该注意的，应用元素不是由它的名称指定（见第 4 章定义），而是由它的应用元素 ID（较短）指定。

存取权限只允许授权给用户组，而不是单个用户。存取权限是以许可而不是限制的方式授权。每个存取权限都在本表的行中设置。

USERGROUPID 列保存存取权限授予用户组的标识符。它是指派生自基础元素 AoUserGroup 的应用元素的一个实例。

AID 列保存实例所属应用元素的标识符。它是引用 SVCENT 表中一个条目（有相同的标识符 AID）。

IID 列保存受存取权限约束的实例的标识符。它是引用相应应用元素实例表中的一个实例。

RIGHTS 列保存五个基本的权限：读取、更新、插入、删除、授权的位屏蔽值。如果各个位屏蔽值都设为（=1），那意味着授予各个权限。表 33 显示了可用的权限。

表33 访问权限位编码

Bit4	Bit3	Bit2	Bit1	Bit0
Grant	Delete	Insert	Update	Read

SVCACLI 表见表 34。

表34 SVCACLI 表

Logical:	GroupID	Appl.ElementID	InstanceID	Rights
Database-column:	USERGROUPID	AID	IID	RIGHTS

6.2.8 SVCACLA

该表保存属性和元素安全的安全数据（ASAM ODS 安全性概念和存取权限的更详细介绍见第 5 章）在 Oracle 数据库服务器中创建 SVCACLA 表的语法如下：

```

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

```

此表保存应用元素和应用属性的访问控制列表的（ACL）条目。存储在该表中的每个条目（表行）都和实体'ACLA'相对应，'ACLA'在第 4 章（基础模型）有定义和描述。但值得注意的是，应用元素不是由它的名称指定（见第 4 章定义），而是由它的应用元素 ID（较短）指定。此外，因为已经引入了 TYPE 列，所以“元素安全性”或“属性安全性”的权限是否在一个特定表的行中指定，并不是根据 AANAME 字段是否为空来决定。

存取权限只允许授权给用户组，而不是单个用户。存取权限应该以授权而不是限制的方式授予。每个存取权限都在本表的行中指定。

当使用 Oracle 数据库服务器时，较灵活的'varchar2 (..)'代替'char (..)'。ASAM ODS 服务器必须支持和 ASAM ODS 早期版本兼容的两种类型。

USERGROUPID 列保存授予存取权限用户组的标识符，它是引用派生自基础元素 AoUserGroup 的应用元素的一个实例。

AID 列保存受存取授权约束或拥有受存取授权约束属性应用元素的标识符。它是引用 SVCENT 表中一个条目（有相同的标识符 AID）。

AANAME 列保存受存取授权约束应用属性的名称。它是引用 SVCATTR 表中一个条目（与 AID 和 AANAME 相匹配的条目）。当为应用元素整体指定存取权限时（由 TYPE="AE"表示），AANAME 值保持为空。

TYPE 列保存选择安全机制类型。对于元素安全性，TYPE 设为“AE”；对于属性安全性，TYPE 设为“AA”。对于“AE”，AANAME 是不相关的，AANAME 任何一个非空值将被服务器忽略。

RIGHTS 列保存五个基本的权限：读取、更新、插入、删除、授权的位屏蔽值。如果各个位屏蔽值都设为（=1），那意味着授予各个权限。表 35 显示了可用的权限。

表35 访问权限位编码

Bit4	Bit3	Bit2	Bit1	Bit0
Grant	Delete	Insert	Update	Read

SVCACLI 表见表 36。

表36 SVCACLI 表

Logical:	GroupID	Appl.ElementID	Attr.Name	Type	Rights
Database-column:	USERGROUPID	AID	AANAME	TYPE	RIGHTS

6.2.9 SVCTPLI

该表保存安全（ASAM ODS 安全性概念的更多详细介绍见第二章）的 ACL 模版。ACL 模版仅对于'实例安全'才是重要的。

在 Oracle 数据库服务器中创建 SVCTPLI 表的语法如下：

```
create table svctpli
(
    USERGROUPID      integer NOT NULL, /* Reference to a UserGroup instance */
```



```

AID          integer NOT NULL, /* Application element id */
IID          integer NOT NULL, /* Application instance id */
REFAID       integer NOT NULL, /* Referencing application element */
RIGHTS       integer NOT NULL /* Rights value (bit masked) */
);

```

该表保存存取控制列表（ACL）条目的模版。存储在该表中的每个条目（表行）都和实体 'ACLTemplate' 相对应，'ACLTemplate' 在第 7 章（基础模型）有定义和描述。但值得注意的是，应用元素和引用的应用元素不是由它们的名称指定（见第 7 章定义），而是由它们的应用元素 IDs（较短）指定。

存取权限只允许授权给用户组，而不是单个用户。存取权限应该以授权而不是限制的方式授予。每个存取权限都在本表的行中指定。

应该注意到，由 ACL 模版指定的存取权限不适用于应用元素的存取，或者 ACL 模版所附实例的存取。相反地，它们用于即将创建实例（通常属于其他应用元素）的 ACL 条目。提供 ACL 模版的使用机制，是为了在每个实例不需要提供自己的 ACL 条目的情况下，ASAM ODS 服务器可以自动产生新实例的 ACL 条目。第 5 章 ASAM ODS 安全当中有详细描述。

USERGROUPID 列保存授予存取权限用户组的标识符，它是引用派生自基础元素 AoUserGroup 的应用元素的一个实例。

AID 列保存 ACL 模版所附应用元素的标识符。它是引用 SVCENT 表中一个条目（有相同的标识符 AID）。ACL 模版是附加在应用元素上（IID=0）上还是它的一个实例上（IID>0），取决于 IID 的值。

IID 列保存 ACL 模版所附实例的标识符。它是引用相应应用元素实例表中一个条目。当模版附加到应用元素上时，IID 必须包括值 0。

REFAID 列保存引用实例的应用元素 ID。如果应用元素 RefAID 的新实例在创建时没有指定自己的 ACL 模版，那么服务器将会在 AID-IID 中找到相应的 ACL 模版，并且将取自 ACL 模版的 USERGROUPID 和 RIGHTS 拷贝到那个实例的 ACL 条目上。

SVCATTR 表中的 ACLREF 列指定创建新实例时，必须采用哪个应用关系进行解析才能找到 ACL 模版。第 5 章有本程序的详细描述。

RIGHTS 列保存五个基本的权限：读取、更新、插入、删除、授权的位屏蔽值。如果各个位屏蔽值都设为 (=1)，那意味着授予各个权限。下表显示了可用的权限：

SVCTPLI 表见表 37。

表37 SVCTPLI 表

Logical:	GroupID	Appl.ElementID	InstanceID	Ref.Appl-ID	Rights
column name:	USERGROUPID	AID	IID	REFAID	RIGHTS

6.2.10 SVCVAL_SPS

局部列的值和标记通常存储在 SVCVAL 表的 BLOB 字段中。虽然这是一个非常紧凑和节省存储空间的方法，但是，当需要选择局部列的单值并进行分析时也会出现一些性能问题。

在这种情况下，可能会用到局部列存储的 **SPS** 版本（单点存储）。它的好处是，测量值不需要塞入 **BLOB** 中，并且不需要对测量值进行分割。因此分割段的大小和数量对性能没有影响。

此外，**SVCVAL_SPS** 支持一些 **SVCVAL** 存储不能使用的数据类型。

与传统的 **SVCVAL** 表相反，**SVCVAL_SPS** 表可以直接引用局部列和相应的子矩阵。

注意，本节只适用于局部列，派生自基本属性“**sequence_representation**”的属性

值属于{**explicit**, **implicit_constant**, **implicit_linear**, **implicit_saw**, **raw_linear**, **raw_polynomial**, **raw_linear_calibrated**}子集在 Oracle 数据库服务器中创建 **SVCVAL_SPS** 表的语法如下：

```
create table svcval_sps
(
  SUBMATRIXID      NUMBER (20) NOT NULL, /* The id of the submatrix */
  LOCALCOLUMNID     NUMBER (20) NOT NULL, /* The id of the local column*/
  POINT             NUMBER (20) NOT NULL, /* Number of the measurement point
  */
  ORD               NUMBER (20) NOT NULL, /* Ordinal number */
  FLAG              NUMBER (5), /* Flag value */
  NUMBER_VALUE      NUMBER, /* Number value, for all numbers */
  STRING_VALUE      VARCHAR2 (4000), /* String value, for strings and dates
  */
  BLOB_VALUE        BLOB /* BLOB value, for BLOBs and byte arrays */
);
```

该表的每一行都是用来存储：

- 测量点一个值（如果是简单的数据类型），或；
- 测量点一个值的一部分（如果不是简单的值），或；
- 一个生成参数或一组生成参数的一部分。

SUBMATRIXID 列指定值所属子矩阵的 **id**。它是引用派生自基础元素 **AoSubmatrix** 的应用元素的一个实例。

LOCALCOLUMNID 列指定值所属的局部列的 **id**。它引用派生自基础元素 **AoLocalColumn** 的应用元素的一个实例。

POINT 列指定测量点，测量点的值是由所在行提供。它的变化范围是从 **1** 到 **N**（假设相应的子矩阵中有 **N** 个测量点）。值“**-1**”可能会出现，在这种情况下，此行不包括特定的测量值，而是包括生成参数的一部分（见下文）。

如果指定一个测量点（例如阵列类型）的值，或一组生成参数需要不止一行，**ORD** 列用于枚举所需行数。因此，它可以对由几部分构成的值进行唯一编号。如果不需要的话，将它设置为 **0**。

FLAG 列包括行所给出的值的标记。如果值的标记没有给出，或该行包括由几部分构成的值的后面部分，或一个参数或一个参数集的一部分存储在该行中，那么 **FLAG** 值就是 **NULL**。

NUMBER_VALUE 用来存储数值。

STRING_VALUE 用来存储字符串值。

BLOB_VALUE 用来存储字节流或 **BLOB** 值。

SVCVAL_SPS 的用途在很大程度上取决于表示的类型（由派生自局部列基本属性“sequence_representation”的属性值给出）和值的数据类型（由派生自于相应测量量基本属性'datatype'的属性值给出）。

以下各节将详细说明支持的类型。

6.2.10.1 显式局部列

这种情况可以从序列表示'explicit'判断。

所有的值都会明确地给出并存储在 SVCVAL_SPS 中。它们存储在表中的方式取决于它们的数据类型。

（1）简单数值

支持的数据类型：DT_SHORT、DT_FLOAT、DT_BOOLEAN、DT_BYTE、DT_LONG、DT_LONGLONG, DT_DOUBLE。

简单的数值存储在 NUMBER_VALUE 列中，这些值的存储精度都是相同的。在对值进行读和写的过程中，API 数据类型的精度在转换过程就决定。注意，当通过 API 访问这些值时，DT_LONGLONG 是两个长值（'high' 和 'low'）的结构。在检索出来，存储和分割成两个短值之前它们会被服务器转换成一个 64 位的整数。

ORD 将会是 0，STRING_VALUE 和 BLOB_VALUE 将包括 NULL。表中的一行只能存储一个值，它的结构见表 38。

表38 简单数值表

SUBMATRIX ID	LOCALCOLUMNID	POINT	ORD	FLAG	NUMBER_VALUE	STRING_VALUE	BLOB_VALUE
<sid>	<cid>	<点>	0	<标记>	<值>	NULL	NULL

表 38 和以后所有结构表中<sid>代表相应的子矩阵的标识符，<cid>代表值所属的局部列的标识符，<point>代表测量点（从 1 到 N），<flag>代表与值相关的标记（见 5.2.4 节），<value>代表将要存储的值。

（2）结构化数值

支持的数据类型：DT_COMPLEX、DT_DCOMPLEX。

结构化数值存储在 NUMBER_VALUE 列中，结构见表 39。但是，在 API 数据类型中，它们的表示是由结构给出的，要将它们存储在 SVCVAL_SPS 表中需要采取具体措施 DT_COMPLEX 和 DT_DCOMPLEX 中的每一个都是浮点类型的两个组件（'r','i'；实部和虚部）的一个结构。每个组件都存储在表的行中。

两种数据类型存储精度是相同的；在对值进行读和写的过程中，API 数据类型的精度在转换过程就确定了。STRING_VALUE 和 BLOB_VALUE 包括 NULL。表中的两行储一个值，ORD 可以区分这两行，结构是（<r-value>代表值的实部，<i-value>代表虚部）。

表39 结构化数值表

SUBMATRIXID	LOCALCOLUMNID	POINT	ORD	FLAG	NUMBER_VALUE	STRING_VALUE	BLOB_VALUE
<sid>	<cid>	<点>	0	<标记>	<值>	NULL	NULL
<sid>	<cid>	<点>	1	NULL	<i值>	NULL	NULL

<sid>, <cid>和<point>条目在这两行中是相同的。<flag>条目与复杂值有关，由实部指定。

(3) 简单的字符串值

支持的数据类型：DT_DATE、DT_STRING。

简单的字符串值存储在 STRING_VALUE 列中，结构见表 40。

ORD 将会是 0, NUMBER_VALUE 和 BLOB_VALUE 将包括 NULL。表中一行存储一个值，它的结构是：

表40 简单字符串值

SUBMATRIXID	LOCALCOLUMNID	POINT	ORD	FLAG	NUMBER_VALUE	STRING_VALUE	BLOB_VALUE
<sid>	<cid>	<点>	0	<标记>	NULL	<值>	NULL

(4) 结构化字符串值

支持的数据类型：DT_EXTERNALREFERENCE。

DT_EXTERNALREFERENCE 是三个组件（描述、 MIME 类型、位置）组成的字符串类型的结构。每个组件都存储在表的行中，结构见表 41。NUMBER_VALUE 和 BLOB_VALUE 将包括 NULL。其中表的三行存储一个值，ORD 可以区分这三行，它的结构是（<desc>代表描述，<mime>代表 MIME 类型，<loc>代表值的位置部分）。

表41 结构化字符串值

SUBMATRIXID	LOCALCOLUMNID	POINT	ORD	FLAG	NUMBER_VALUE	STRING_VALUE	BLOB_VALUE
<sid>	<cid>	<点>	0	<标记>	NULL	<desc>	NULL
<sid>	<cid>	<点>	1	NULL	NULL	<mime>	NULL
<sid>	<cid>	<点>	2	NULL	NULL	<loc>	NULL

<sid>、<cid>和<point>条目在这三行中都是相同的。<flag>条目与外部参考值有关，由描述部分指定。

(5) 二进制流值

支持的数据类型：DT_BYTESTR、DT_BLOB。

DT_BYTESTR 值存储在 BLOB_VALUE 列中。

ORD 将会是 0, NUMBER_VALUE 和 STRING_VALUE 将包括 NULL。表中的其中一行是存储一个值，它的结构是见表 42。

表42 二进制流值

SUBMATRIXID	LOCALCOLUMNID	POINT	ORD	FLAG	NUMBER_VALUE	STRING_VALUE	BLOB_VALUE
<SID>	<CID>	<点>	0	<标记>	NULL	NULL	<值>

DT_BLOB 值由标题和 BLOB 本身组成。标题存储在 STRING_VALUE 列中，而 BLOB 存储在 BLOB_VALUE 列中。

ORD 将会是 0, NUMBER_VALUE 将包括 NULL。表中一行是存储一个值，它的结构见表 43。

表43 二进制流值

SUBMATRIXID	LOCALCOLUMNID	POINT	ORD	FLAG	NUMBER_VALUE	STRING_VALUE	BLOB_VALUE
<SID>	<CID>	<point>	0	<flag>	NULL	<标题>	<值>

(6) 简单数值数组

支持的数据类型：DS_SHORT、DS_FLOAT、DS_BOOLEAN、DS_BYTE、DS_LONG、DS_LONGLONG、DS_DOUBLE。

每个阵列元素都存储在表中一行的 NUMBER_VALUE 列中。因此，M 个元素元素的阵列将使用 M 行，ORD 可以区分一行中的阵列元素（ORD 值在 0 和 M-1 之间变化）。STRING_VALUE 和 BLOB_VALUE 将会包括 NULL，它的结构是（<value[m]>代表阵列元素 mth 的值，m=0..M-1），见表 44。

表44 简单数值数组

SUBMATRIXID	LOCALCOLUMNID	POINT	ORD	FLAG	NUMBER_VALUE	STRING_VALUE	BLOB_VALUE
<SID>	<CID>	<点>	0	<标记>	<值[0]>	NULL	NULL
<SID>	<CID>	<点>	1	NULL	<值[1]>	NULL	NULL
...	...	...	...	...	...	...	...
<SID>	<CID>	<点>	M-1	NULL	<值[M-1]>	NULL	NULL

<SID>、<CID>和<point>条目在所有的行中都是相同的。<flag>条目与整个阵列值相关，由第一个阵列元素指定。

因为所有的值都存储在与数据类型无关的 NUMBER_VALUE 列中，所以在值进行读和写的过程中，API 数据类型的精度在转换过程必须通过服务器来完成。更加值得注意的是，当通过 API 访问这些值的时候，DS_LONGLONG 的每一个阵列元素都是由两个长值（'high'和'low'）组成的结构。在检索出来，存储和分割成两个长值之前它们会被服务器转换成一个 64 位的整数。

(7) 结构化数值数组

支持的数据类型：DS_COMPLEX、DS_DCOMPLEX。

结构化数值阵列也存储在 NUMBER_VALUE 列中。但是，因为一个值由许多组件组成，所以每个值所需要的行都不只一个。因此，在表中（假设有 M 个阵列元素）存储它们需要几个 M 行。

DS_COMPLEX 和 DS_DCOMPLEX 两者中的每一个都是有两个组分（'r', 'i'；实部和虚部）组成

的值阵列。每个组件都存储在表的一行中。这两个数据类型存储精度相同。在对值进行读和写的过程中，API 的数据类型的精度在转换过程就确定了。所以 M 个元素组成的阵列会使用 $2 \times M$ 行，ORD 可以区分组件及存储在一行中的阵列元素（ORD 值的变化范围在 0 和 $2 \times M - 1$ 之间）。。STRING_VALUE 和 BLOB_VALUE 将包括 NULL，结构是（<r[m]>代表阵列元素 mth 的实部，代表阵列元素 mth 的虚部； $m=0..M-1$ ），结构见表 45。

表45 结构化数值数组

SUBMATRIXID	LOCALCOLUMNID	POINT	ORD	FLAG	NUMBER_VALUE	STRING_VALUE	BLOB_VALUE
<SID>	<CID>	<点>	0	<标记>	<r[0]>	NULL	NULL
<SID>	<CID>	<点>	1	NULL	<i[0]>	NULL	NULL
<SID>	<CID>	<点>	2	NULL	<r[1]>	NULL	NULL
<SID>	<CID>	<点>	3	NULL	<i[1]>	NULL	NULL
...	...	...	...	...	...	...	...
<SID>	<CID>	<点>	2-M-1	NULL	<i[M-1]>	NULL	NULL

<SID>、<CID>和<point>条目在所有的行中是相同的。<flag>条目与复杂值的整个阵列相关，由第一个阵列元素的实部指定。

（8）简单字符串值数组

支持的数据类型： DS_DATE, DS_STRING。

每个阵列元素存储在表中一行的 STRING_VALUE 列中。所以 M 个元素构成的阵列会使用 M 行，ORD 可以区分存放在表一行当中的阵列元素(ORD 值的变化范围在 0 和 $M-1$ 之间)。NUMBER_VALUE 和 BLOB_VALUE 将包括 NULL，结构是（<value[m]>代表阵列元素 mth 的值； $m=0..M-1$ ），结构见表 46。

表46 简单字符串值数组

SUBMATRIXID	LOCALCOLUMNID	POINT	ORD	FLAG	NUMBER_VALUE	STRING_VALUE	BLOB_VALUE
<SID>	<CID>	<点>	0	<标记>	NULL	<值[0]>	NULL
<SID>	<CID>	<点>	1	NULL	NULL	<值[1]>	NULL
...	...	...	...	...	...	...	...
<SID>	<CID>	<点>	M-1	NULL	NULL	<值[M-1]>	NULL

<SID>、<CID>和<point>条目在所有的行中是相同的。<flag>条目与阵列值相关，由第一个阵列元素指定。

（9）结构化字符串值数组

支持的数据类型： DS_EXTERNALREFERENCE。

结构化的字符串值阵列也存储在 STRING_VALUE 列中。

但是，因为一个值由许多部分组成，所以每个值所需要的行都不只一个。因此，在表中（假设有 M 个阵列元素）存储它们需要几个 M 行。

DS_EXTERNALREFERENCE 的每个阵列元素都是由三个组件（描述、MIME 类型、位置）的字符串类型结构。每个组件都存储在表的一行中。所以，表中存储一个值需要 $3 \times M$ 行，ORD 可以区分这

三行, NUMBER_VALUE 和 BLOB_VALUE 将包括 NULL, 结构是(<desc[m]>代表阵列值 mth 的描述, <mime[m]>代表 MIME 类型, <loc[m]>代表位置), 结构见表 47。

表47 结构化字符串值数组

SUBMATRIXID	LOCALCOLUMNID	POINT	ORD	FLAG	NUMBER_VALUE	STRING_VALUE	BLOB_VALUE
<SID>	<CID>	<点>	0	<标记>	NULL	<desc[0]>	NULL
<SID>	<CID>	<点>	1	NULL	NULL	<mime[0]>	NULL
<SID>	<CID>	<点>	2	NULL	NULL	<loc[0]>	NULL
<SID>	<CID>	<点>	3	NULL	NULL	<desc[1]>	NULL
<SID>	<CID>	<点>	4	NULL	NULL	<mime[1]>	NULL
<SID>	<CID>	<点>	5	NULL	NULL	<loc[1]>	NULL
...	...	...	...	...	...	...	...
<SID>	<CID>	<点>	3-M-1	NULL	NULL	<loc[M-1]>	NULL

<SID>、<CID>和<point>条目在所有的行中是相同的。<flag>条目与外部引用值的整个阵列有关, 由第一个阵列元素的描述部分指定。

6.2.10.2 隐式局部列及其生成算法

这种情况是通过子集{implicit_constant, implicit_linear, implicit_saw, raw_linear, raw_polynomial, raw_linear_calibrated}的序列表示区分的。

值要么联机生成, 要么只给出原始值, 终值必须通过它们计算得出。两种情况都需要生成参数, 并存储在 SVCVAL_SPS 中。它们在表中的存储方式取决于序列表示和原始值的数据类型。

在任何一种情况下, 数值只能为局部列创建。支持的数据类型是 DT_SHORT、DT_FLOAT、DT_BOOLEAN、DT_BYTE、DT_LONG、DT_LONGLONG、DT_DOUBLE、DT_COMPLEX、DT_DCOMPLEX。

(1) 简单数值数据类型的隐式表达

如果序列表示是'implicit_constant', 'implicit_linear', 或'implicit_saw', 那么只有生成参数需要存储。生成参数的数量 and 意义取决于序列表示的类型, 与局部列产生的值的数量无关。

如果派生自相应测量量基本属性'datatype'的属性是 DT_SHORT、DT_FLOAT、DT_BOOLEAN、DT_BYTE、DT_LONG、DT_LONGLONG 或 DT_DOUBLE, 本段解释说明了 SVCVAL_SPS 类型。

对于'implicit_constant', 局部列的所有值 x_n 是不变的, 只需要存储一个为常量的生成参数 p_1 。 p_1 存储在 NUMBER_VALUE 列中。存储精确度与值 x_n 的数据类型无关; 在对值进行读和写的过程中, API 的数据类型的精度在转换过程就确定了。

POINT 将会是-1, ORD 将会是 1, STRING_VALUE 和 BLOB_VALUE 将包括 NULL。

表 50 的每一行存储局部列的完整规范, 结构见表 48。

表48 简单数值数据类型的隐式表达

SUBMATRIXID	LOCALCOLUMNID	POINT	ORD	FLAG	NUMBER_VALUE	STRING_VALUE	BLOB_VALUE
<SID>	<CID>	-1	1	NULL	<p ₁ >	NULL	NULL

对于 'implicit_constant', 局部列的所有值 x_n 都在直线 $x_n = p_1 + (n-1) * p_2$ 上, 并且只需要存储两个生成参数 p_1 和 p_2 。

它们存储在 NUMBER_VALUE 列中。存储精确度与值 x_n 的数据类型无关; 在对值进行读和写的过程中, API 的数据类型的精度在转换过程就确定了。

POINT 将会是-1, ORD 可以区分这两个参数。STRING_VALUE 和 BLOB_VALUE 将包括 NULL. 存储局部列的完整规范需要两行, 见表 49。

表49 POINT 的隐式表达 1

SUBMATRIXID	LOCALCOLUMNID	POINT	ORD	FLAG	NUMBER_VALUE	STRING_VALUE	BLOB_VALUE
<SID>	<CID>	-1	1	NULL	<p ₁ >	NULL	NULL
<SID>	<CID>	-1	2	NULL	<p ₂ >	NULL	NULL

<SID>和<CID>条目在这两行中是相同的。

对于'implicit_saw' , 局部列的所有值 x_n 根据公式 $x_n = p_1 + ((n-1) * \text{mod} (p_3 - p_1) / p_2) * p_2$ 建立了一个

锯齿曲线, 其中需要存储三个生成参数 p_1 , p_2 和 p_3 。它们存储在 NUMBER_VALUE 列中。存储精确度与值 x_n 的数据类型无关; 在对值进行读和写的过程中, API 的数据类型的精度在转换过程就确定了。

POINT 将会是-1, ORD 可以区分这三个参数。STRING_VALUE 和 BLOB_VALUE 将包括 NULL. 存储局部列的完整规范需要三行, 结构见表 50。

表50 POINT 的隐式表达 2

SUBMATRIXID	LOCALCOLUMNID	POINT	ORD	FLAG	NUMBER_VALUE	STRING_VALUE	BLOB_VALUE
<SID>	<CID>	-1	1	NULL	<p ₁ >	NULL	NULL
<SID>	<CID>	-1	2	NULL	<p ₂ >	NULL	NULL
<SID>	<CID>	-1	3	NULL	<p ₃ >	NULL	NULL

<SID>和<CID>条目在这三行中是相同的。

(2) 具有简单数值数据类型的原始数据预处理

如果序列表示是'raw_linear', 'raw_polynomial', 或'raw_linear_calibrated' , 那么原始数据和生成参数都需要存储。生成参数的数量 and 意义取决于原始数据预处理的类型, 而不取决于为局部列而产生的值的数量。原始数据的数量和局部列产生值的数量相匹配。

如果派生自相应测量量基本属性'datatype'的属性是 DT_SHORT, DT_FLOAT, DT_BOOLEAN, DT_BYTE, DT_LONG, DT_LONGLONG, or DT_DOUBLE 类型, 该段解释说明了 SVCVAL_SPS; 这点导致每个生成参数和每个原始值都是简单的数值。

对于 'raw_linear', 根据公式 $x_n = p_1 + p_2 * r_n$, 局部列的所有值 x_n 都是原始值 r_n 的线性映射, 除了

原始值之外，只需要存储两个生成参数 p_1 和 p_2 。参数和原始值都存储在 NUMBER_VALUE 列中。存储精度不取决于值 x_n 和 m 数据类型；在对值进行读和写的过程中，API 数据类型的精度在转换过程就确定了。

每个参数都存储在表的一行中；在这种情况下，POINT 是-1，ORD 可以区分这些参数。

每个原始值都存储在表的一行中，ORD 设为 0，POINT 用来区分测量点（变化范围是从 1 到 N），STRING_VALUE 和 BLOB_VALUE 将包括 NULL，POINT 的隐式表达 4 见表 51。

表51 POINT 的隐式表达 4

SUBMATRIXID	LOCALCOLUMNID	POINT	ORD	FLAG	NUMBER_VALUE	STRING_VALUE	BLOB_VALUE
<SID>	<CID>	-1	1	NULL	<p ₁ >	NULL	NULL
<SID>	<CID>	-1	2	NULL	<p ₂ >	NULL	NULL
<SID>	<CID>	n	0	<flag>	<r _n >	NULL	NULL

<SID>和<CID>条目在所有行中都是相同的。

对于 'raw_polynomial'，根据公式 $x_n = p_2 + p_3 \cdot r_n + p_4 \cdot r_n^2 + p_{2+k} \cdot r_n^k$ （其中 $k=p_1$ ），局部列的所有值 x_n 都是原始值 r_n 的 K 阶多项式的映射，除了原始值之外，只需要存储 $k+2$ 个生成参数 $p_1 \dots p_{2+k}$ 。参数和原始值都存储在 NUMBER_VALUE 列中。存储精度不取决于值 x_n 和 r_n 的数据类型；在对值进行读和写的过程中，API 的数据类型的精度在转换过程就确定了。

每个参数都存储在表的一个行中；在这种情况下，POINT 是-1，ORD 可以区分这些参数。

每个原始值也都存储在表的一个行中，ORD 设为 0，POINT 可以区分测量点（变化范围是从 1 到 N），STRING_VALUE 和 BLOB_VALUE 将包括 NULL，结构见表 52。

表52 POINT 隐式表达 5

SUBMATRIXID	LOCALCOLUMNID	POINT	ORD	FLAG	NUMBER_VALUE	STRING_VALUE	BLOB_VALUE
<SID>	<CID>	-1	1	NULL	<p ₁ >	NULL	NULL
<SID>	<CID>	-1	2	NULL	<p ₂ >	NULL	NULL
...	...	...	...	...	...	...	...
<SID>	<CID>	-1	K+2	NULL	<p _{k+2} >	NULL	NULL
<SID>	<CID>	n	0	<flag>	<r _n >	NULL	NULL

<SID>和<CID>条目在所有行中都是相同的。

对于 'raw_linear_calibrated'，根据公式 $x_n = (p_1 + p_2 \cdot r_n) \cdot p_3$ ，局部列的所有值 x_n 都是通过特定线性程序映射自原始值 r_n ，除了原始值只需要存储量三个生成参数 $p_1 \dots p_3$ 。

参数和原始值都存储在 NUMBER_VALUE 列中。存储精度不取决于值 x_n 和 r_n 的数据类型；在对值进行读和写的过程中，API 的数据类型的精度在转换过程就确定了。

每个参数都存储在表的一行中；在这种情况下，POINT 是-1，ORD 可以区分这些参数。

每个原始值也都存储在表的一行中，ORD 设为 0，POINT 可以区分测量点（变化范围是从 1 到 N），STRING_VALUE 和 BLOB_VALUE 将表示 NULL。结构见表 53。

表53 POINT 隐式表达 6

SUBMATRIXID	LOCALCOLUMNID	POINT	ORD	FLAG	NUMBER_VALUE	STRING_VALUE	BLOB_VALUE
<sID>	<cID>	-1	1	NULL	<p1>	NULL	NULL
<sID>	<cID>	-1	2	NULL	<p2>	NULL	NULL
<sID>	<cID>	-1	3	NULL	<pk+2>	NULL	NULL
<sID>	<cID>	n	0	<flag>	<rn>	NULL	NULL

<sID>和<cID>条目在所有行中都是相同的。

(3) 结构化的数值数据类型的原始数据预处理

如果序列表示是'raw_linear'、'raw_polynomial' 或 'raw_linear_calibrated', 并且派生自相应测量量基本属性'datatype'的属性是 DT_COMPLEX 或 DT_DCOMPLEX, 那么每一个原始值都是一个结构化、由两个浮点数 ('r', 'i'; 实部和虚部) 组成的数值。所以, 存储每个原始值需要使用两行。在表中这些局部列的存储方式取决于序列表示。

对于 'raw_linear', 根据公式 $x_n = p_1 + p_2 * r_n$, 局部列的所有值 x_n 都是原始值 r_n 的线性映射, 除了原始值之外, 只需要存储两个生成参数 p_1 和 p_2 。参数和原始值都存储在 NUMBER_VALUE 列中。存储精度不取决于值 x_n 和 r_n 的数据类型; 在对值进行读和写的过程中, API 数据类型的精度在转换过程就确定了。

每个参数都存储在表的一行中; 在这种情况下, POINT 是-1, ORD 可以区分这些参数。

每个原始值都存储在表的两行中, ORD 可以区分原始值的实部和虚部 (设为 0 表示实部, 1 表示虚部), POINT 可以区分测量点 (变化范围是从 1 到 N), STRING_VALUE 和 BLOB_VALUE 将包括 NULL。 所以结构 (<r[rn]>代表 nth 原始值 r_n 的实部, <i[rn]>代表虚部; $n=0..N-1$), 见表 54。

表54 结构化数值数据预处理

SUBMATRIXID	LOCALCOLUMNID	POINT	ORD	FLAG	NUMBER_VALUE	STRING_VALUE	BLOB_VALUE
<sID>	<cID>	-1	1	NULL	<p ₁ >	NULL	NULL
<sID>	<cID>	-1	2	NULL	<p ₂ >	NULL	NULL
<sID>	<cID>	1	0	<标记>	<r[r ₀]>	NULL	NULL
<sID>	<cID>	1	1	NULL	<i[r ₀]>	NULL	NULL
...	...	...	...	...	...	...	...
<sID>	<cID>	N	0	<标记>	<r[r _{N-1}]>	NULL	NULL
<sID>	<cID>	N	1	NULL	<i[r _{N-1}]>	NULL	NULL

<sID>和<cID>条目在所有的行中都是相同的。<flag>条目与每个复杂的原始值有关, 由它的实部指定。

对于 'raw_polynomial', 根据公式 $x_n = p_2 + p_3 * r_n + p_4 * r_n^2 + p_{2+k} * r_n^k$ (其中 $k=p_1$), 局部列的所有值 x_n 都是原始值 r_n 的 K 阶多项式映射, 除了原始值之外, 只需要存储 $k+2$ 个生成参数 $p_1 .. p_{2+k}$ 。参数和原始值都存储在 NUMBER_VALUE 列中。存储精确度不取决于值 x_n 和 r_n 的数据类型; 在对值进行读和写的过程中, API 的数据类型的精度在转换过程就确定了。

每个参数都存储在表的一行中；在这种情况下，POINT 是-1，ORD 可以区分这些参数。

每个原始值都存储在表的两行中，ORD 可以区分原始值的实部和虚部（设为 0 表示实部，设为 1 表示虚部），POINT 可以区分测量点（变化范围是从 1 到 N），STRING_VALUE 和 BLOB_VALUE 将包含 NULL。所以结构<r[rn]>代表 nth 原始值 rn 的实部，<i[rn]>代表虚部；n=0..N-1），见表 55。

表55 虚数的表达

SUBMATRIXID	LOCALCOLUMNID	POINT	ORD	FLAG	NUMBER_VALUE	STRING_VALUE	BLOB_VALUE
<SID>	<CID>	-1	1	NULL	<p ₁ >	NULL	NULL
<SID>	<CID>	-1	2	NULL	<p ₂ >	NULL	NULL
...	...	...	...	...	...	...	...
<SID>	<CID>	-1	k+2	NULL	<p _{k+2} >	NULL	NULL
<SID>	<CID>	1	0	<flag>	<r[r ₀]>	NULL	NULL
<SID>	<CID>	1	1	NULL	<i[r ₀]>	NULL	NULL
...	...	...	...	...	...	...	...
<SID>	<CID>	N	0	<flag>	<r[r _{N-1}]>	NULL	NULL
<SID>	<CID>	N	1	NULL	<i[r _{N-1}]>	NULL	NULL

<SID> and <CID>条目在所有的行中是相同的。<flag>条目与每个复杂的原始值有关，由它的实部指定。

对于'raw_linear_calibrated'，根据公式 $x_n = (p_1 + p_2 \cdot r_n) \cdot p_3$ 局部列的所有值 x_n 都是通过特定线性程序映射自原始值 r_n ，除了原始值之外，只需要存储量三个生成参数 $p_1 \dots p_3$ 。参数和原始值都存储在 NUMBER_VALUE 列中。存储精度不取决于值 x_n 和 r_n 的数据类型；在对值进行读和写的过程中，API 的数据类型的精度在转换过程就确定了。

每个参数都存储在表的一行中；在这种情况下，POINT 是-1，ORD 可以区分这些参数。

每个原始值也都存储在表的两行中，ORD 可以区分原始值的实部和虚部（设为 0 表示实部，设为 1 表示虚部），POINT 可以区分测量点（变化范围是从 1 到 N），STRING_VALUE 和 BLOB_VALUE 将包括 NULL。所以结构（<r[rn]>代表 nth 原始值 rn 的实部，<i[rn]>代表虚部；n=0..N-1），见表 56。

表56 两个虚数的表达

SUBMATRIXID	LOCALCOLUMNID	POINT	ORD	FLAG	NUMBER_VALUE	STRING_VALUE	BLOB_VALUE
<SID>	<CID>	-1	1	NULL	<p ₁ >	NULL	NULL
<SID>	<CID>	-1	2	NULL	<p ₂ >	NULL	NULL
<SID>	<CID>	-1	2	NULL	<p ₃ >	NULL	NULL
<SID>	<CID>	1	0	<标记>	<r[r ₀]>	NULL	NULL
<SID>	<CID>	1	1	NULL	<i[r ₀]>	NULL	NULL
...	...	...	...	...	...	...	...
<SID>	<CID>	N	0	<flag>	<r[r _{N-1}]>	NULL	NULL
<SID>	<CID>	N	1	NULL	<i[r _{N-1}]>	NULL	NULL

<SID>和<CID>条目在所有的行中是相同的。<flag>条目与每个复杂的原始值有关，由它的实部指

定。

6.3 属性值的存储

6.3.1 单值的存储

大部分属性是单值属性，其中单值属性的数据类型枚举值在 1 ... 8 之间。在 SVCATTR 表中，这些值直接存储在 DBCNAME 给出的列中。ASAM ODS 的数据类型必须和数据库引擎的数据类型相符。

数据类型枚举 DT_DATE (= 10) 的属性存储将在下面的第 6.3.4 节中进行描述。

其他的数据类型不会作为单值存储（见第 5 章）。

6.3.2 字节流值的存储

数据类型枚举 DT_BYTESTR (= 11) 的属性值存储在相应实体表的二进制原始数据类型列（例如，Oracle 数据库服务器的‘long raw’）中。字节流的长度和‘long raw’字段的长度一致。Oracle 数据库服务器限制每个表只能有一个‘long raw’列，所以，每个应用元素只能允许存在一个具有数据类型枚举类型 DT_BYTESTR 或 DT_BLOB 的应用属性。

6.3.3 BLOB 值的存储

具有数据类型枚举 DT_BLOB (=12) 的属性值存储在表的两列中。

BLOB 的标题存储在一个列，这个列的名称由 SVCATTR 表的 DBCNAME 给出。该列的数据库数据类型是“VARCHAR”。

BLOB 属性体存储在一个单独的列中，它有一个固定的名字，叫做‘Blob’。该列是一个二进制的原始数据类型（例如，Oracle 数据库服务器的‘long raw’）。

本标准限制具有 BLOB 数据类型的应用属性的数量最大为每个应用元素一个。

此外，对于 Oracle 数据库服务器，每个应用元素最大一个应用属性，应用属性可以赋予数据类型枚举 DT_BLOB 或 DT_BYTESTR。

6.3.4 日期和时间的存储

在 ASAM ODS 中，存储具有数据类型枚举 DT_DATE 的应用属性值的标准方法是使用字符串。T_DATE 值的语法描述见 5.5.2 节。

这种存储格式可以选用无限制解决方案；然而对于所需要的存储空间来说，这不是最佳的。因此，一个服务器可能在内部使用数据库使用的日期和时间类型（如 Oracle 的‘date’和‘timestamp’类型更适于在 Oracle 数据库中使用）。

由于这影响了相应的应用元素表中的列的数据类型，又因为改变列（该列已经有值，而要在转换时不丢失它们）的数据类型不是一件小事，所以当指定应用模型时，就应该决定是存储日期和时间的数据类型。。

注意：内部存储使用的任何数据类型都必须对用户隐藏。ASAM ODS 服务器必须以日期值形式提供日期和时间，并按照如上所述表示为字符串，这些字符串值总能为客户端应用程序所接收。

这就要求 ASAM ODS 服务器必须能够处理所有与特定数据库相关的日期和时间类型，以及 ASAM ODS 字符串表示，还能够在 ASAM ODS 字符串表示和特定数据库相关的日期和时间类型之间进行转

换（怎么转换是由包括 DT_DATE 型的应用属性的列的数据类型隐式给出；这个数据类型可能已经由管理者手动设置了）。

6.3.5 属性阵列值和对象值的存储

在 ASAM ODS 中，属性值可能是值或对象（例如，DS_BYTE、DT_EXTERNALREFERENCE）的一个阵列，每个属性值提供的元素都不只一个。

这种应用属性值存储在一个单独的表中。所以，当一个应用元素包括单元素应用属性和多元素应用属性时，实例存储在两个表中。

其中一个表通过 SVCENT 表的 DBTNAME 列指定的名称命名；它包括后来保存每个实例值的单元元素应用属性的列。

另外一个表通过 SVCENT 表的 DBTNAME 列指定的名称命名，加上“_ARRAY”后缀；（注意：这更进一步限制了所有包括多个应用属性值的应用元素的表名长度，）；它存储所有多元素应用属性值。

- 然而第一个表没有任何特别之处，第二个表的特别之处在于它的结构。它具有：
- 表示实例 id 的列（叫做 IID，并且和表 DBTNAME 给出的 ID 列有相同的类型）；
 - 表示阵列或对象元素的序号的列（叫做 ORD，总是以 0 开始，是一个非负整数）和；
 - 表示每个保存属性值的多元素应用属性的列。在 SVCATTR 表中这个列的名称由 DBCNAME 给出。在这个列中，值的数据类型和 SVCATTR 表中 ADTYPE 列的数据类型一致。

一个实例的多元素应用属性值存储在本表的几个连续行中。一个实例要求的行数 N 是由所有多元素属性的阵列/对象元素数的最大数量决定的。对于所有的行，IID 是不变的，而 ORD 是递增的，从 0 开始，以 N-1 结束。即使表中大量单元是空的情况下，这种情况也会发生。

示例：TEST_ARRAY 表。

假定 DBTNAME=“Test”的应用程元素“T1”和两个多元素应用程属性 ATT1 和 ATT2。ATT1 指定 ADTYPE=DS_DOUBLE（=21）和 DBCNAME=“DOUBLE_SEQ”。ATT2 指定 ADTYPE=DS_LONG（=20）和 DBCNAME=“LONG_SEQ”。

假定应用元素有两个实例，它们的序列在长度上不同。IID=1 的第一个实例有一个含 4 个双精度值的序列和一个含 2 个长值的序列。

IID=2 的第二个实例有一个含 2 个双精度值的序列和一个含 4 个长值的序列。

在这种情况下，有两个多元素应用属性的表被命名为“Test_ARRAY”并且有以下结构和内容，见表 57。

表57 多元素应用属性的表

IID	ORD	DOUBLE_SEQ	LONG_SEQ
1	0	1.0	2
1	1	2.0	4
1	2	3.0	
1	3	4.0	
2	0	1.0	2
2	1	2.0	4
22			6

23			8

对象的属性值（例如，具有数据类型枚举 DT_COMPLEX,DT_DCOMPLEX,DT_EXTERNALREFERENCE）有一个预先确定的元素数目，该元素的意义也已预先确定；每个元素的序号在表 58 中指定。

表58 对象类型应用程序属性

类型	字段	序号	字段数据类型
T_EXTERNALREFERENCE	描述	0	T_STRING
T_EXTERNALREFERENCE	Mime类型	1	T_STRING
T_EXTERNALREFERENCE	位置	2	T_STRING
T_COMPLEX	R	0	T_FLOAT
T_COMPLEX	1	1	T_FLOAT
T_DCOMPLEX	R	0	T_DOUBLE
T_DCOMPLEX	1	1	T_DOUBLE

6.3.5.1 通过 RPC-API 访问

ASAM ODS RPC-API 不能处理阵列或对象类型的属性。在通过 RPC-API 访问 ASAM ODS 服务器的情况下，要求对应用模型（可能物理存储中已经有值存在）进行修改，确保不会有多元素应用性存在。

只要应用模型创建时，存在多元素应用属性的应用元素，以后就不能通过 RPC-API 访问了。

6.3.6 枚举类型属性的存储

如果应用属性的数据类型 ADTYPE 是 DT_ENUM (= 30) 或 DS_ENUM (= 31)，那么将有一个或多个数据类型 T_LONG 值写入物理存储。

对于要呈现人类可读的字符串而不是枚举值的应用，必须寻找枚举的名称（由 SVCATTR 中的应用属性 ENUMNAME 列给出），然后找到和应用的属性值相对应的字符串表示 ENUMNAME（从 SVCENUM 表的 ENUMNAME 列获取，在行中 ENUMNAME 等于解决到枚举名称，ITEM 等于给定的值）。

有合适的方法给 ASAM ODS APIs 使用。但是从数值型应用属性值向字符串表示的转换是应用而不是 ASAM ODS 服务器的责任。编写应用也是如此：它必须向服务器写入数值，而用户通常通过字符串表示选择枚举项。将字符串表示转化成数字是由客户端的应用负责。

出于性能方面的考虑，客户端缓存枚举定义是明智的，否则会产生大量的客户机/服务器网络流量。如果将该属性做为排序标准使用，也应该使用值（T_LONG 类型）而不是字符串表示进行排序。

ASAM ODS 服务器允许客户端使用 OO-API 来创建和修改应用模型；还提供修改名称或枚举的特定名称值对的方法。这些方法最终是修改应用模型，因此只能由超级用户使用（其他任何人不得修改应用模型）。

6.3.7 继承概念和物理存储

继承概念主要由 ASAMODS 提供，其目的是给出存储信息的详细结构并节省存储空间，尤其是物理存储空间。至于如何完成，本节将进行解释。

6.3.7.1 应用属性和关系的存储

超类应用元素和子类应用元素都存储在 **SVCENT** 表中，而它们的应用属性存储在 **SVCATTR** 表中。通用的应用属性和超类的 **AID** 一起保存在 **SVCATTR** 表中。当确定子类的特定应用元素时，子类的 **AID** 只能在 **SVCATTR** 中使用。然而这里有一个例外：每个子类要额外确定派生自基本属性'id'的应用属性。因此，该应用属性在 **SVCATTR** 中被确定很多次：为超类确定一次，为每个子类确定一次。在那种情况下，虽然物理存储允许给该应用属性起不同的应用属性名称，但这种情况应严格避免，只能在现有的遗留数据存在这种情况的前提下，才能被 ASAM ODS 接受。

基础的应用关系（它的关系类型可能是 **FATHER_CHILD** 或 **INFO**）和扩展的应用关系（它的关类型可能是 **INFO**）是有区别的，他们不同的地方在于基础的应用关系在 **SVCATTR** 表的 **BANAME** 列有一条记录。超类和子类（关系类型是 **INHERITANCE**）之间的关系可以根据以下事实简单区分：子类派生自基本属性'id'的应用属性的 **FAID** 列不为空，而是包含相关的超类应用元素的 **AID**。

这些应用元素创建的实例存储在单独的实体表中。

为超类应用元素创建的实体表包括对于这两类元素来说都通用的所有的应用属性和应用关系的列。

为子类应用元素创建的实体表包括两种列：其中一种是还未包括在超类应用元素中的应用属性和关系的列，另一种是派生自基本属性'id'的属性列。

注：请注意ASAM ODS要求将基础的应用关系放到超类的应用元素中。所以子类的应用元素只能确定扩展的应用关系。

超类应用元素实例的所有属性和关系值都存储在超类的实体表中。超类应用元素实例的属性和关系值都将会在超类和子类的实体表之间分配。通用属性和关的所有值都存储在超类的实体表中，而所有特定的子类属性和关系值（此外，'id'值连接实体表的主键）都存储在子类的实体表。

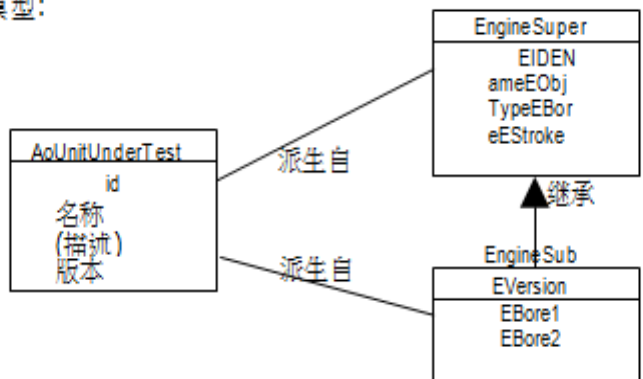
对于每个代表子类的继承应用元素，必须建立超类和子类表的数据库视图。这两个表通过派生自基本属性'id、实例类型'的应用属性值连接在一起，例如，实例的共同部分具有从超类继承来的特定应用元素相同的 **id** 值。

实现子类的表名在表 **SVCENT** 中存为 **DBTNAME**。视图名是表名导出的并增加了前缀'**SCV_**'（子类视图，请注意这个名字不同与物理存储中以'**SCV_**'开头的服务器表名）。

对于没有 **ORACLE** 序号的继承应用元素，为了便于自动计算在数据库中规定下一个自由标识符；代替超类中使用的 **id** 值。

示例：继承的物理存储见图 43。

给出以下应用模型：



实例EngineSuper(EName="SuperEng",EBores=80.4, EStroke=85.6)和实例EngineSub(EName="SubEng",EBores=80.5,EStroke=85.3,EVersion="2.4",EBores1=81.5,EBores2=79.5)被存储.标识符(EID)由服务器隐式确定;表名或列名也是如此

SVCENT包括以下记录(除其他之外):

AID	ANAME	BID	DBNAME	SECURITY
...	...	...	...	...
12	EngineSuper	21(AoUnitUnderTest)	TabEngineSuper	0
13	EngineSub	21(AoUnitUnderTest)	TabEngineSub	0

SVCATTR包括以下记录(除其他之外):

AID	ATTRNR	ANAME	ANAME	FAID	FUNIT	ADTYPE	AFLEN	DBNAME	ACLREFINVNAME	FLAG
...	...	...	...	...	...	...	...	...	...	...
12	0	EID	id			8		ColEID		5
12	1	EName	name			1	32	ColEName		
12	2	EObjType	objecttype			8		ColEObjType		
12	3	EBores				7		ColEBores		
12	4	EStroke				7		ColEStroke		
13	0	EID	id	12		8		ColEID		5
13	1	EVersion	version			1	32	ColEVersion		
13	2	EBores1				7		ColEBores1		
13	3	EBores2				7		ColEBores2		

注意，在子类应用元素中，派生自基本属性'id'的应用属性的名称和超类应用元素的相应名称设置相同。

'TabEngineSuper'表包括超类和子类应用元素所有实例的通用属性值:

ColEID	ColEName	ColEObjType	ColEBore	ColEStroke	
...	...	...	...	...	
27	SuperEng	12	80.4	85.6	
31	SubEng	13	80.5	85.3	

'TabEngineSub'表包括子类应用元素实例的额外属性值以及EID值:

ColEID	ColEVersion	ColEBore1	ColEBore2	程序
...	...	...	...	
31	2.4	81.5	79.5	

对于超类和子类应用元素的所有实例，EID值是唯一的。因为它的值在两个表中都存在，所以服务器可以建立子类应用元素的特定实例的完整属性集。

为了能够存取子类实例,视图SCV_TabEngineSub必须通过下面的SQL语句创建:

```
create or replace view SCV_TabEngineSub as select
    Sup.ColEID,
    Sup.ColEName, Sup
    .ColEObjType, Sup
    .ColEBore, Sup.Co
    lEStroke, Sub.Col
    EVersion, Sub.Col
    EBore1, Sub.ColEB
    ore2
from TabEngineSuper Sup, TabEngineSub Sub where
    Sup.ColEID=Sub.ColEID
and Sup.ColEObjType=13;
```

视图包括超类和子类的所有属性，但是只有一个ID属性。

图43 继承的物理存储

6.3.7.2 实例属性的存储

实例属性与相应元素的 AID 一起存储。因此，如果一个实例属性属于超类应用元素的一个实例，那么，SVCINST 表的 AID 列保存超类应用元素的 AID；如果一个实例属性属于子类应用元素的一个实例，那么 SVCINST 表的 AID 列保存子类应用元素的 AID。

6.4 混合模式的存储

在许多情况下，在汽车测试环境中，必须存储大量的的测量数据。为此 ASAM ODS 提供了基础元素 AoLocalColumn。在表 SVCVAL 中，基于 RDB 的物理存储要求以 blob 类型存储它们。这可能会快速增加数据库的大小，但由于数据库的限制，寻找具体的数据可能会非常耗时（如，在数据库中 Oracle

必须处理 blobs 时)。

为此 ASAM ODS 确定了基于纯 RDB 的存储替代规范, 允许在单独的文件中保存海量数据, 但要把这些外部数据的描述放在数据库中。在单独的二进制文件中存储海量数据的概念延伸自分割的概念。分割是将一个局部列的海量数据分割成许多二进制文件的过程; 这样做, 可以将大的二进制文件分成许多更小的文件, 这样会提高性能、并发能力、文件处理效率。

这种方法将在本节进行解释说明。这就是所谓的混合模式存储, 实施这种存储的服务器叫做混合模式服务器, 有关混合模式服务器的一些内容将在本节后面给出。

在单独的文件中存储数据仅限于存储海量数据。只有派生自 AoLocalColumn 的基本属性 'values' 和 'flags' 的属性才能存储在这些文件中。因此, 只有局部列必须知道值被存储在外部文件的方法。为此, 每一个局部列可能指向 AoExternalComponent 型应用元素的一个或多个实例。另请参阅第 4 章关于 ASAM ODS 基础模型的描述。

一个外部文件可能包含属于不同局部列的几个测量数量的数据。为了区分它们, 它们在文件中的确切位置必须确定, 并且样式必须有一定规律。这些文件的结构取决于将要存储在里面的数据类型。本节描述了混合模式存储目前支持的这些数据类型的结构。

应该指出的是, 派生自基本属性 'sequence_representation' 的属性值最终决定着 SVCVAL 表中的外部组成文件以及 VALBLOB (包括 VALUES 和 FLAGS 元素) 的内容。下表给出了可能存在的一个概述, 该概述确定 VALBLOB 的 VALUES 元素, VALBLOB 的 FLAGS 元素和外部组成文件 (如果有) 的内容。

部件组成文件和序列表示见表 59。

表59 部件组成文件和序列表示

sequence_representation	VALUES	FLAGS	external component files
explicit	the final values	the flags (if any)	none
implicit_constant implicit_linear implicit_saw	the generation parameters	no flags allowed	none
raw_linear raw_polynomial raw_linear_calibrated	the generation parameters followed by the raw values	the flags (if any)	none
external_component	empty	empty	the final values the flags (if any)
raw_linear_external raw_polynomial_external raw_linear_calibrated_external	the generation parameters	empty	the raw values the flags (if any)

在第 6.2.4 节描述了前三种情况, 没有涉及外部组成文件。

如果 'sequence_representation' 以 'external_component' 类型给出, 那么终值会被放在外部组成文件中。这些值的数据类型 (通过 API 呈现给用户) 是由派生自相应测量量的基本属性 'datatype' 的属性值决定的; 它可能和 DT_STRING、DT_SHORT、DT_FLOAT、DT_BOOLEAN、DT_BYTE、DT_LONG、DT_DOUBLE、DT_LONGLONG、DT_DATE、DT_BYTESTR、DT_BLOB、DT_COMPLEX、DT_DCOMPLEX 当中一个一致。然而外部组成文件的表示可能和这些数据类型不同; 它是由派生自相应

外部组成文件的基本属性“value_type”的属性值给出的。所以，服务器必须使用“value_type”读取来自外部组成文件的原始值，并且在呈现给用户之前能将它们转化成“datatype”类型。

如果“sequence_representation”是“raw_linear_external”、“raw_polynomial_external”或“raw_linear_calibrated_external”，那么原始值会被放在外部组成文件中，并且终值通过使用以个生成参数由原始值计算得到，这个生成参数由相应局部列的 VALBLOB 的 VALUES 元素给出。

局部列（通过 API 呈现给用户）的值的数据类型由派生自相应测量量的基本属性“datatype”的属性值确定；它可能和 DT_SHORT、DT_FLOAT、DT_BYTE、DT_LONG、DT_DOUBLE、DT_LONGLONG、DT_COMPLEX、DT_DCOMPLEX（注意，只允许数值型的数据类型）中的一个一致。原始数据的数据类型由派生自相应局部列的基本属性“raw_datatype”的属性值给出。然而，外部组成文件的表示可能和这些数据类型不同；它由派生自相应外部组成文件的基本属性“value_type”的属性值给出（也只允许数值型的数据类型）。服务器必须能使用“value_type”读取来自外部组成文件的原始值，并且将它们转换成相应的“raw_datatype”类型。

如果“sequence_representation”是“raw_linear_external”、“raw_polynomial_external”或“raw_linear_calibrated_external”，那么原始值会被放在外部组成文件中，并且终值通过使用以个生成参数由原始值计算得到，这个生成参数由相应局部列的 VALBLOB 的 VALUES 元素给出。

局部列（通过 API 呈现给用户）的值的数据类型由派生自相应测量量的基本属性“datatype”的属性值确定；它可能和 DT_SHORT、DT_FLOAT、DT_BYTE、DT_LONG、DT_DOUBLE、DT_LONGLONG、DT_COMPLEX、DT_DCOMPLEX（注意，只允许数值型的数据类型）中的一个一致。原始数据的数据类型由派生自相应局部列的基本属性“raw_datatype”的属性值给出。然而，外部组成文件的表示可能和这些数据类型不同；它由派生自相应外部组成文件的基本属性“value_type”的属性值给出（也只允许数值型的数据类型）。服务器必须能使用“value_type”读取来自外部组成文件的原始值，并且将它们转换成相应的“raw_datatype”类型。

外部组成文件可能包括值和标志。值和标志可能放在相同或不同的外部组成文件中。外部组成文件可能包括不同的数据类型值，从而将不同的局部列的数据组合起来。尽管不是所有的组合都允许，但是可能的组合还是很多的。如何将这外部组成文件结构化，本节将进行描述。

6.4.1 BLOB 类型值的存储

一个或多个二进制文件可能包含一个局部列的 BLOB 值。如果值是 blob 类型，那么一个二进制文件只包含一个 BLOB（不含 blob 头文件，它存储在外部组成文件的基本属性“description”派生出的属性中）。如果局部列包含多个 BLOB，那么每个 blob 都存储在一个单独的二进制文件中，并且通过使用它的一些属性，由 AoExternalComponent 类型的应用元素的单独实例描述：

——派生自基本属性“start_offset”、“block_size”、“valuesperblock”和“value_offset”的属性是不相关的并且可能被省略；

——派生自基本属性“value_type”的属性确定用来存储值的数据类型。对于 blob 类型值，它是枚举项“dt_blob”；

——派生自基本属性“filename_url”的属性确定保存 blob 的文件名称。

派生自基本属性“component_length”的属性确定 blob 的长度（字节数）。由于没有更多的信息保

存在该文件中，所以它和二进制文件的长度相同。如果局部列有多个外部组成文件，那么派生自基本属性 'ordinal_number' 的属性确定局部列中的 BLOB 序列。'ordinal_number' = 1 的外部组成文件保存第一个 blob 'ordinal_number' = 2 的外部组成文件包括第二个 blob，直到最后具有最高的 'ordinal_number' 的外部组成文件保存局部列的最后一个 blob。

有了这个信息，blob 类型的局部列的值可以很容易地检索：

——每个外部组成文件都可以读取 BLOB 值，因为它（从 'filename_url'）知道二进制文件的名称，（从 'value_type'）知道这个二进制文件只包括一个 blob，也（从 'component_length'）知道 blob 的大小；

——随后，局部列可以收集来自所有相关外部组成文件的 blobs，因为它（从 'ordinal_number'）知道外部组成文件的正确顺序。

6.4.2 字符串类型值的存储

一个或多个二进制文件可能包括一个或多个局部列的字符串值。

包括字符串值的二进制文件可能不包括其他数据类型的值。

每个二进制文件包括可选文件头部分，和紧随其后文件块。

如果值是字符串类型（而不是 BLOB 类型的数据），那么一个二进制文件可存储多个值，尽管属于一个局部列的文件中的所有值都必须放在同一块中，彼此相邻。

一个二进制文件可能包含多个局部列的值。另一方面，一个局部列的值可能分布在多个二进制文件中，如果适用的话。

因此，在混合模式环境中存储字符串类型值的外部二进制文件的整体结构可以表示成图 44。

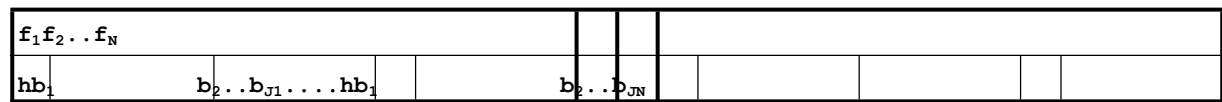


图44 混合模式存储：字符串类型值的文件结构

其中：

- $f_1, f_2, \dots, f_n, \dots, f_N$ 是外部二进制文件，该文件保存一个或多个局部列的字符串值；
- h 是一个二进制文件的文件头；
- $b_1, b_2, \dots, b_j, \dots, b_{Jn}$ 是二进制文件 f_n （ $n=1..N$ ）中的块；
- 二进制文件 f_n （ $n=1..N$ ）中的块数 J_n 可能和不同的二进制文件 f_m 中的块数 J_m 不同；
- 一个块包括一个局部列的一个或多个值一个文件中的两个不同的块不可能包括相同局部列的值；
- 一个块中的值是 NUL 结束的。0x00（NUL）值使一个块中的连续值彼此分开；
- 一个字符串的每个字符以 8 位数字形式存储。

示例：2 个字符串（ASCII 码内容为）“ABCDEFGH”和“XYZ”由以下 16 进制序列代表 414243444546470058595A0，总长是 12，所以 component_length=12。

与局部列相关的每个二进制文件的内容可以通过 AoExternalComponent 型应用元素一个实例的某

些属性描述：

- 派生自基本属性 'block_size'、'valuesperblock' 和 'value_offset' 的属性是不相关的并且可能被省略；
- 派生自基本属性 'value_type' 的属性确定存储值的数据类型。对于字符串类型值，它是枚举项 'dt_string' 或 'dt_string_flags_beo'（引入这两种类型可以区分相应标记的字节存储次序（见下面）；字符串中的字节序列不依赖于这里确定的枚举项）；
- 派生自基本属性 'filename_url' 的属性确定存储字符串值的文件名称；
- 派生自基本属性 'component_length' 的属性确定存储在文件中的所有字符串值的总长度（字节数），这个文件属于相应的局部列。NUL 终止符也要增加这个值，所以 **component_length** 等于所有存储字符串的字符数加上保存的字符串数；
- 派生自基本属性 'start_offset'（字节数）的属性确定了文件中相应局部列的第一个字符串值开始的字节位置。这是文件中的文件头长度加上所有块累计长度；
- 派生自基本属性 'ordinal_number' 的属性确定局部列中的外部组成文件的序列，如果局部列有多个外部组成文件。'ordinal_number' = 1 的外部组成文件保存第一组字符串值，'ordinal_number'=2 的外部组成文件保存第二组字符串值，等等，直到最后具有最大的 'ordinal_number' 的外部组成文件包括局部列的最后一组字符串值。

有了这个信息，字符串类型的局部列的值可以很容易检索：

- 每个外部组成文件都可以收集和它的局部列相关的字符串值，因为它（从 'filename_url'）知道二进制文件的名称，（从 'value_type'）知道这个二进制文件保存字符串值，（从 'start_offset'）知道属于相应局部列的字符串的开头的字节位置，（从 'component_length'）知道必须分析的字节总数，也知道被 NUL 终止符分开的字符串。**component_length** 总是包括每个字符串末尾的 NUL 终止符。一个外部组成文件中的字符串数可以通过计算外部组成文件内的 NUL 字符而被修改；
- 随后，局部列可以收集来自所有相关外部组成文件的字符串值的列表，因为它可以（从 'ordinal_number'）知道外部组成文件的正确序列。

6.4.3 字节流类型值的存储

一个或多个二进制文件可能包括一个或多个局部列。包含字节流值的二进制文件可能不包含其他数据类型的值。每个二进制文件包括可选文件头部分，和紧随其后文件块。如果值是字节流类型（而不是 BLOB 类型的数据），那么一个二进制文件可存储多个值，尽管属于一个局部列的文件中的所有值都必须放在同一块中，彼此相邻。

一个二进制文件可能包含多个局部列的值。另一方面，一个局部列的值可能会遍及多个二进制文件，如果适用的话。

因此，在混合模式环境中存储字节流类型值的外部二进制文件的整体结构可以表示成图 45。

$f_1 f_2 \dots f_N$							
hb ₁	$b_2 \dots b_{j1} \dots$	hb ₁		b _i		$\dots b_{jN}$	

图45 混合模式存储：字节流类型值的文件结构

其中：

—— $f_1, f_2, \dots, f_n, \dots, f_N$ 是外部二进制文件，该文件包括一个或多个局部列的字节流值。 h 是一个二进制文件的文件头；

—— $b_1, b_2, \dots, b_j, \dots, b_{J_n}$ 是一个二进制文件 f_n ($n=1..N$) 中的块；

——二进制文件 f_n ($n=1..N$) 中的块数 J_n 可能和不同的二进制文件 f_m 中的块数 J_m 不同；

——一个块包括一个局部列的一个或多个值；

——一个文件中的两个不同的块不可能包括相同的局部列；

——一个块内的值前面是四个字节的长度信息，可以解释为无符号的整数值；它的字节存储顺序取于派生自基本属性‘value_type’的属性给出的值。

示例：对于 DT_BYTESTR_BE0, 2 个字节流（ASCII 码）“ABCDEFGH”和“XYZ”由以下 16 进制序列代表 00000007414243444546470000000358595A 总长度是 18，所以 component_length=18。

与局部列相关的每个二进制文件的内容可以通过 AoExternalComponent 型应用元素一个实例的某些属性描述：

——派生自基本属性‘block_size’、‘valuesperblock’和‘value_offset’的属性是不相关的并且可能被省略派生自基本属性‘value_type’的属性确定存储值的数据类型。对于字节流类型值，它是枚举项“dt_bytestr”或“dt_bytestr_beo”（引入这两种类型可以区分相应标记的字节存储次序（见下面）和字节流的长度信息；字节流中的字节序列不依赖于这里确定的枚举项）；

——派生自基本属性‘filename_url’的属性确定存储字节流值的文件名称；

——派生自基本属性‘component_length’的属性确定保存在文件中的所有字节流值的总长度（字节数），其中这个文件属于相应的局部列。每个字节流的 4 位长度信息也有助于这个值，所以 component_length 等于所有包括的字节流的字节数和包括的字节流数的和。派生自基本属性‘start_offset’（字节数）的属性确定了文件中相应局部列的第一个字节流值开始的字节位置（确切地说是：它的长度信息的开始部分）。这是文件中的文件头长度加上所有块累计长度；

——派生自基本属性‘ordinal_number’的属性确定局部列中的外部组成文件序列，如果局部列有多个外部组成文件。‘ordinal_number’=1 的外部组成文件保存第一组字节流值，‘ordinal_number’=2 的外部组成文件保存第二组字节流值，等等，直到最后具有最高的‘ordinal_number’的外部组成文件保存局部列的最后一组字节流值。

有了这个信息，字节流类型的局部列的值可以很容易检索：

——每个外部组成文件都可以收集和它的局部列相关的字节流值，因为它（从‘filename_url’）知道二进制文件的名称，（从‘value_type’）知道这个二进制文件包括字节流值，（从‘start_offset’）知道属于相应局部列的字节流的开头的字节位置，（从‘component_length’）知道必须分析的字节总数，也知道字节流在每个字节流开头前面有 4 字节的长度信息。在一个外部组成文件中的字节流数能通过读取整个块以及计算字节流检索出来；

——随后，局部列可以收集来自所有相关的外部组成文件的字节流值的列表，因为它（从

“ordinal_number”) 知道外部组成文件的正确序列。

6.4.4 数值型值的存储

一组二进制文件可能包括一组局部列的整型或浮点值。

每个二进制文件包括可选文件头部分，和紧随其后文件块。每个块可能包括不同局部列的值。尽管属于一个局部列的文件中的所有值都必须放在同一块中，彼此相邻。但是在同一个块中局部列可能多个值，因此要创建一个子块。一个二进制文件中的每个块的结构都必须是相同的；这意味着如果它和相同的局部列相关，它必须包括相同数量的子块，而且每个块中子块的位置必须相同。因此，在混合模式环境中存储字节流类型值的外部二进制文件的整体结构可以表示成图 46。

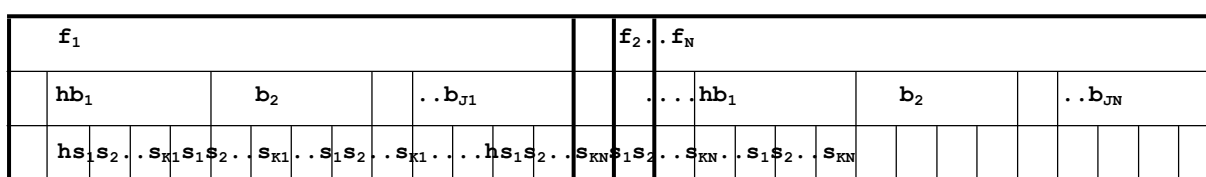


图46 混合模式存储：数值的文件结构

其中：

- $f_1, f_2, \dots, f_n, \dots, f_N$ 是外部二进制文件，该文件包括不同局部列的数值；
- h 是一个二进制文件的文件头；
- $b_1, b_2, \dots, b_j, \dots, b_{jN}$ 是一个二进制文件 f_n ($n=1..N$) 中的块；二进制文件 f_n ($n=1..N$) 中的块数 J_n 可能和不同的二进制文件 f_m 中的块数 J_m 不同；
- $s_1, s_2, \dots, s_k, \dots, s_{Kn}$ 是二进制文件 f_n ($n=1..N$) 中一个块的子块。相同二进制文件中的每个块的子块数必须是相同的。二进制文件 f_n 中的块中的子块数 K_n 可能和不同二进制文件 f_m 中的块中的子块数 K_m 不同；
- 一个子块包括一个局部列的一个或多个值。

在相同二进制文件 f_n 中，任何一个块 b_j 中的子块 s_k 必须包括相同局部列的相同数值。因为子块在另一个块 b_l (其中 $j, l \in [1, J_n]$) 中有相同的位置，一个块中两个不同的子块不可能包括相同的局部列值。

一个文件的子块 s_k 中的所有值属于相同的局部列；与它们出现在文件中的次序一样，它们出现在局部列中的次序也相同。

如果一个局部列的值遍布多个二进制文件，那么派生自 `AoExternalComponent` 的基本属性 'ordinal_number' 的属性会确定局部列值的最终次序是如何由每个文件中的单个值次序创建的：它以 'ordinal_number'=1 的外部组成文件引用的文件值开始，随着 'ordinal_number' 的增长而增长。

`AoExternalComponent` 提供了一些可能在相应的应用元素中用到的属性，目的是确切地确定局部列值将存储在二进制文件中的什么位置：

- 每个局部列知道保存它的值的所有外部组成文件。外部组成文件用来读取值的序列是由派生自基本属性 'ordinal_number' 的属性确定的（如果局部列有多个外部组成文件）。'ordinal_number'=1 的外部组成文件保存第一组值，'ordinal_number'=2 的外部组成文件保存第

二组值等等,直到最后具有最高的'ordinal_number'的外部组成文件保存局部列的最后一组值。每个外部组成文件能确切地知道由派生自基本属性'filename_url'的属性给出的二进制文件。每个外部组成文件值的序列包含在这个文件中。

- 每个外部组成文件能知道由派生自基本属性'start_offset'的属性给出文件中的文件头长度（字节数）。它必须跳过文件开始部分的字节数。
- 每个外部组成文件知道每个块内的子块的位置（含相关值），这是由派生自基本属性“value_offset”的属性给出。它又被确定为字节数；外部组成文件必须跳过每个块开始的字节数。
- 每个外部元件知道将要存取的值的数据类型，这由派生自基本属性 value_type 的属性给出。它也知道这些值的物理存储方法，其中包括在二进制文件（见下文）中存储数据类型的位/字节数的信息。

注意，'value_type' 确定二进制文件中值的物理表示，但不一定要和 ASAM ODS 服务器内通过 API 接口获得的值。后者由派生自相应测量量的基本属性 'datatype' 的属性确定。

每个外部元件知道每个子块内值的数量，这由派生自基本属性“valuesperblock”的属性给出。因此，它可以确切地提取来自当前块的值的数量（即分块）。

每个外部元件都知道块的大小，这由派生自基本属性“BLOCK_SIZE”（字节数）的属性给出。因此，通过添加“BLOCK_SIZE”字节到当前子块开始部分，可以进入到下一个块内的相同子块的开始部分，然后它可以提取其他组（其他的子块）值。

每个外部组成文件知道那个文件中相应局部列分割段的值的总数，这是由派生自基本属性 'component_length' 的属性给出的。因此，它知道何时移到下一个子块以及何时停止。局部列可能会收集来自所有外部组件的值列表并建立终值序列，以派生自基本属性 'ordinal_number' 的属性信息为基础。

下面的段落描述了外部二进制文件的数值型数据类型值的物理表示。它们之所以不同是因为值的类型（由派生自外部组成文件的基本属性“value_type”的属性给出）。

注意外部组成文件中使用的数据类型（值的类型，可以在 typespec_enum 中找到）和 ASAM ODS 的内部数据类型（在 datatype_enum 中找到）之间存在差异。ODS 服务器将在文件数据类型和内部数据类型之间进行转换。

6.4.4.1 值类型 dt_boolean 或 dt_boolean_flags_beo 的值表达

在二进制文件中每个布尔对象都应该用一个位表示（0=false, 1=true）。这些被填充到字节序列中。如果位的数目不足以完全填补一个字节，那么该字节的其余部分应保持闲置。字节是从左到右填充的，也就是以最高有效位开始。在位序列中，在 dt_boolean 和 dt_boolean_flags_beo 之间没有差异；介绍的这两种类型仅仅是用来区别相应标记的字节存储次序的（见下面）。在字节序列（BY1, BY2, by3,...）中，第 n 个布尔对象（N=1...NBI）可以通过使用以下的整数公式进行定位。假设一个字节中的数位位置是编号（也是从左到右）7,6,5,4,3,2,1,0，那么第 n 个布尔对象是数位位置 lbi 的第 lby 个字节：

$$lby = (n+7) / 8;$$

$$lbi = lby * 8 - n.$$

使用的字节数目（Nby）可以通过整数算法由布尔对象数计算得到：

$N_{by}=1+(N_{bi}-1)/8$;

派生自相应外部组成文件的基本属性'component_length'的属性值表达使用的位（即给定的布尔值数）的数量，因此等于 NBI。

6.4.4.2 值类型 dt_byte 或 dt_byte_flags_beo 的值表达

这种无符号的数据类型只是简单地表现为字节。因此派生自相应外部组成文件的基本属性'component_lengt'的属性值表达存储在外部组成文件并属于外部组成文件的字节值数量。在字节序列中，在 dt_byte 和 dt_byte_flags_beo 之间不存在差异；介绍的这两种类型仅仅是用来区别相应标记的字节存储次序的（见下面）。

6.4.4.3 值类型 dt_sbyte 或 dt_sbyte_flags_beo 的值表达

这种有符号的数据类型只是简单地表现为字节。因此派生自相应外部组成文件的基本属性'component_length'的属性值表达存储在外部组成文件并属于外部组成文件的字节值数量。在字节序列中，在 dt_sbyte 和 dt_sbyte_flags_beo 之间不存在差异；介绍的这两种类型仅仅是用来区别相应标记的字节存储次序的（见下面）。

6.4.4.4 值类型 dt_short 或 dt_short_beo 的值表达

dt_short 和 dt_short_beo 代表 16 位的有符号的整型数据，使用 2 个字节存储在外部二进制文件中。

——值类型=dt_short: 当读/写来自/到文件 i/o 流的字节时，最低有效字节位于开头（低地址），最后出现的是最高有效字节（高地址）；

——值类型=dt_short_beo: 当读/写来自/到文件 i/o 流的字节时，最高有效字节位于开头，最后出现的是最低有效字节；这个顺序也被称为“大字节存储次序”。

因此派生自相应外部组成文件的基本属性'component_length'的属性值表达存储在外部组成文件并属于外部组成文件的整数值数量。

6.4.4.5 值类型 dt_ushort 或 dt_ushort_beo 的值表达

dt_ushort 和 dt_ushort_beo 代表 16 位无符号整型数据，使用 2 个字节存储在外部二进制文件中。

——值类型=dt_ushort: 当读/写来自/到文件 i/o 流的字节时，最低有效字节位于开头（低地址），最后出现的是最高有效字节（高地址）；

——值类型=dt_ushort_beo: 当读/写来自/到文件 i/o 流的字节时，最高有效字节位于开头，最后出现的是最低有效字节；这个顺序也被称为“大字节存储次序”。

因此派生自相应外部组成文件的基本属性'component_length'的属性值表达存储在外部组成文件并属于外部组成文件的整数值数量。

6.4.4.6 值类型 dt_long 或 dt_long_beo 的值表达

dt_long 和 dt_long_beo 代表 32 位有符号整型数据，使用 4 个字节存储在外部二进制文件中。

——值类型=dt_long: 当读/写来自/到文件 i/o 流的字节时，最低有效字节位于开头（低地址），最后出现的是最高有效字节（高地址）；

——值类型=dt_long_beo: 当读/写来自/到文件 i/o 流的字节时, 最高有效字节位于开头, 最后出现的是最低有效字节; 这个顺序也被称为“大字节存储次序”。

因此派生自相应外部组成文件的基本属性 'component_length' 的属性值表达存储在外部组成文件并属于外部组成文件的整数值数量。

6.4.4.7 值类型 dt_ulong 或 dt_ulong_beo 的值表达

dt_ulong 和 dt_ulong_beo 代表 32 位无符号整型数据, 使用 4 个字节存储在外部二进制文件中。

——值类型=dt_ulong: 当读/写来自/到文件 i/o 流的字节时, 最低有效字节位于开头(低地址), 最后出现的是最高有效字节(高地址);

——值类型=dt_ulong_beo: 当读/写来自/到文件 i/o 流的字节时, 最高有效字节位于开头, 最后出现的是最低有效字节; 这个顺序也被称为“大字节存储次序”。

因此派生自相应外部组成文件的基本属性'component_length'的属性值表达存储在外部组成文件并属于外部组成文件的整数值数量。

6.4.4.8 值类型 dt_longlong 或 dt_longlong_beo 的值表达

dt_longlong 和 dt_longlong_beo 代表 64 位有符号整型数据, 使用 8 个字节存储在外部二进制文件中。

——值类型=dt_longlong: 当读/写来自/到文件 i/o 流的字节时, 最低有效字节位于开头(低地址), 最后出现的是最高有效字节(高地址);

——值类型=dt_longlong_beo: 当读/写来自/到文件 i/o 流的字节时, 最高有效字节位于开头, 最后出现的是最低有效字节; 这个顺序也被称为“大字节存储次序”。

因此派生自相应外部组成文件的基本属性'component_length'的属性值表达存储在外部组成文件并属于外部组成文件的整数值数量。

6.4.4.9 值类型 ieeefloat4 或 ieeefloat4_beo 的值表达

ieeefloat4 和 ieeefloat4_beo 代表 4 个字节浮点数据, 使用 4 个字节存储在外部二进制文件中。

——值类型=ieeefloat4: 当读/写来自/到文件 i/o 流的字节时, 最低有效字节位于开头(低地址), 最后出现的是最高有效字节(高地址);

——值类型=ieeefloat4_beo: 当读/写来自/到文件 i/o 流的字节时, 最高有效字节位于开头, 最后出现的是最低有效字节; 这个顺序也被称为“大字节存储次序”。

符号位、指数和尾数的位置和数量是根据二进制浮点架构的 IEEE 标准(ANSI/IEEE 754)中确定的 IEEE 单精度式定义。

因此派生自相应外部组成文件的基本属性 'component_length' 的属性值表达存储在外部组成文件并属于外部组成文件的浮点值数量。

(1) 复杂的单精度值

如果派生自相应测量量的基本属性'datatype'的属性是 DT_COMPLEX(意味着测量量的值是复数值),那么派生自相应外部组成文件的基本属性'value_type'的属性必须是 ieeefloat4 或 ieeefloat4_beo.

外部二进制文件中的值的物理表示, 必须始终显示两个连续的 4 个字节的浮点值对儿, 第一个解

释为复数值的实数部分，第二个是虚数部分。一个文件中包含的浮点数的总数是（复数值）数值数量的两倍。即使是在这种情况下，派生自基本属性'component_length'的属性包括 4 个字节的浮点值的数量。因为每个数字是由两个 4 个字节的浮点值表达的，所以数值的数量是派生自基本属性'component_length'的属性值的一半，并且这个属性可能只包括非负甚至整数值。

6.4.4.10 值类型 ieeefloat8 或 ieeefloat8_beo 的值表达

ieeefloat8 和 ieeefloat8_beo 代表 8 个字节的浮点数据，使用 8 个字节存储在外部二进制文件中。

——值类型=ieeefloat8：当读/写来自/到文件 i/o 流的字节时，最低有效字节位于开头（低地址），最后出现的是最高有效字节（高地址）；

——值类型=ieeefloat8_beo：当读/写来自/到文件 i/o 流的字节时，最高有效字节位于开头，最后出现的是最低有效字节；这个顺序也被称为“大字节存储次序”。

符号位的位置和数量，指数和尾数是指根据二进制浮点架构的 IEEE 标准（ANSI/IEEE 754）中确定的 IEEE 双精度型（double）数据格式定义。

因此派生自相应外部组成文件的基本属性'component_length'的属性值表达存储在外部组成文件并属于外部组成文件的浮点值数量。

（1）复杂的双精度值

如果派生自相应测量量的基本属性'datatype'的属性是 DT_COMPLEX（意味着测量量的值是复数值，那么派生自相应外部组成文件的基本属性'value_type'的属性必须是 ieeefloat8 或者 orieeefloat8_beo。外部二进制文件中的值的物理表示，必须始终显示两个连续的 8 个字节的浮点值对，第一个解释为复数值的实数部分，第二个是虚数部分。一个文件中包含的浮点数的总数是（复数值）数值数量的两倍。即使是在这种情况下，派生自基本属性'component_length'的属性包括 8 个字节的浮点值的数量。为每个数字都是由两个 4 个字节的浮点值表达的，所以数值的数量是派生自基本属性'component_length'的属性值的一半，并且这个属性可能只包括非负甚至整数值。

6.4.5 标记的存储

二进制文件可能包括各个局部列中一个或多个测量值的值标记。如果派生自基本属性 sequence_representation 的属性值是 external_component、raw_linear_external、raw_polynomial_external、raw_linear_calibrated_external，那么标记会放在二进制文件中而不是 VALBLOB 的 FLAGS 元素中。在这些情况下，派生自基本属性 flags_filename_url 的属性值决定存储它们的文件。如果那个属性是空的，那么将不会有任何标记。

保存标志的文件可能和（但不必）保存值的文件相同。

每个二进制文件包括可选文件头部分，和紧随其后文件块。局部列分割段的所有标记必须放在彼此紧邻的文件中的相同块中。一个二进制文件包括一个或多个局部列的标记。

因此，混合模式环境下，存储标记的外部二进制文件的整体结构可以由图 47 表示。

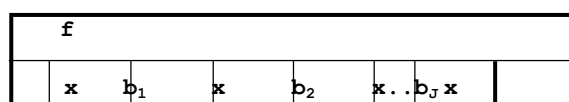


图47 混合模式存储：标记的文件结构

- 其中：
- f** 是包括一个或多个局部列标记的外部二进制文件；
 - 块'**x**'是可选的；如果它们存在，那么它们包括与标记无关的信息（不关心）；
 - b₁,b₂,...,b_j** 是包括标记信息的二进制文件中的块；
 - 块 **b_j** 包括局部列分割块中的所有标记；
 - 标记的数量必须和相应局部列分割段值的数量相同；
 - 每个标记都以 **16 位**（也就是 **DT_SHORT**）的数存储。因此，一个块中的字节数是相应局部列值的数量的两倍。如果派生自那个外部组成文件的基本属性'**value_type**'的属性为“大字节存储次序”（通过 **extension_beo** 字段），那么，标记存储在大字节存储次序中，以便当读/写来自/到文件 **i/o** 流的字节时最高有效字节位于开头。否则，最低有效字节出现在开头（小字节存储次序）；
 - 每个标记都是数值信息状态的一个组合；每个状态组合都被编码成 **16 位** 的标记位。每个位的意思在表 **60** 中给出。通常，所有的位都被设定了（因此，值为 **0x000F**）。这些标记的含义会在基础模型的 **AoLocalColumn** 中做进一步的描述。

表60 值标记的位编码

缩写	值	描述 [↵]
AO_VF_VALID	0x0001	值是有效的
AO_VF_VISIBLE	0x0002	值必须可视化 ·
AO_VF_UNMODIFIED	0x0004	值不能被修改 · [↵]
AO_VF_DEFINED	0x0008	值可以定义.如果在一个值矩阵中其值是不可用的，那么这个位不能设定 · [↵]

6.4.6 示例

本节提供混合模式数字值存储的一些例子。

示例1：单通道加上时间信息见图 48。

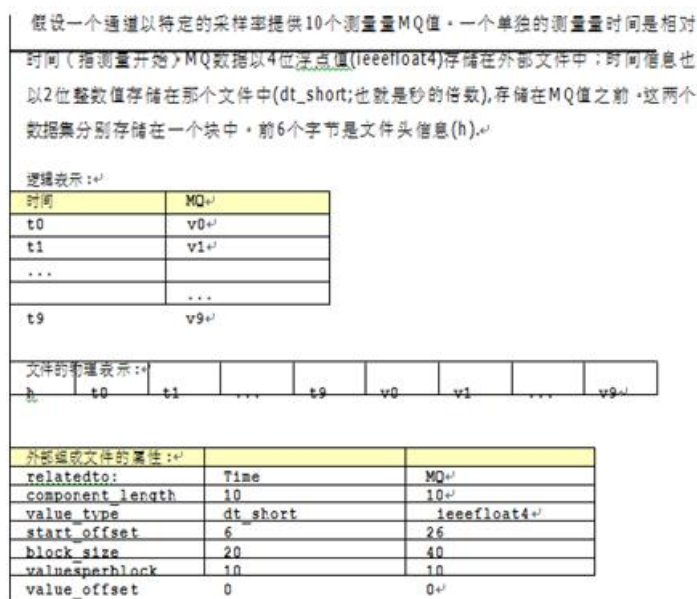


图48 单通道加上时间信息

示例2: 3个通道交错见图49。

假设下表说明的3个通道以不同并且非相等的时间间隔分别为3个不同的测量提供3个值(时间值t1,t2,...代表时间上的等距点)。⁴⁾

这样的一组数据不能依据合模式存储概念保存在外部二进制文件中。⁴⁾

逻辑表示：⁴⁾

时间	MQ1	MQ2	MQ3 ⁴⁾
t1	m1v1	m2v1	m3v1 ⁴⁾
t2		m2v2 ⁴⁾	
t3	m1v2	m2v3	m3v2 ⁴⁾
t4	m1v3 ⁴⁾		
t5	m1v4	m2v4	m3v3 ⁴⁾
t6			m3v4 ⁴⁾
... ⁴⁾			

图49 三个通道交错

示例3: 3个通道部分交错见图50。

假设3个通道以相同的采样率分别为3个不同的测量量提供6个值。值以8位浮点值 (ieeefloat8)存储在外部文件中，一个通道中的两个连续值彼此紧邻的放在一子块中，并且紧随其后的是下一个通道的两个连续值。在下面的例子中时间存储在外部文件中。

逻辑表示:

Time	MQ1	MQ2	MQ3
t1	m1v1	m2v1	m3v1
t2	m1v2	m2v2	m3v2
t3	m1v3	m2v3	m3v3
t4	m1v4	m2v4	m3v4
t5	m1v5	m2v5	m3v5
t6	m1v6	m2v6	m3v6

文件的物理存储:

m1v1m1v2	m2v1	m2v2	m3v1	m3v2	m1v3	m1v4	m2v3	>>
>>m2v4	m3v3	m3v4	m1v5	m1v6	m2v5	m2v6	m3v5	m3v6

外部组成文件的属性:

relatedto:	MQ1	MQ2	MQ3
component length	6	6	6
value type	ieeefloat8	ieeefloat8	ieeefloat8
start offset	0	0	0
block size	48	48	48
valuesperblock	2	2	2
value offset	0	16	32

图50 三个通道部分交错

注意，尽管混合模式存储定义非常灵活，但有些情况不适合混合模式存储。下面的例子就是这种情况

示例4：非均匀测量；不支持，见图 51。

假设3个通道以不同的采样率进行测量:

m1:10Hz采样率的双精度浮点数
m2:20Hz采样率的短整数
m3:30Hz采样率的长整数

逻辑表示:(时间值t1,t2,...代表在时间上以1/60s为一个时间间隔的等距点)

时间	MQ1	MQ2	MQ3
t1	m1v1	m2v1	m3v1
t2			
t3			m3v2
t4		m2v2	
t5			m3v3
t6			
t7	m1v2	m2v3	m3v4

图51 非均匀测量

这种情况下，转换器应该将这个数据映射到与 ASAM ODS 兼容的任何物理存储中。下一个例子在混合模式中也不可能实施，尽管每个通道的样本是以相等的时间间隔采集：

示例5：均匀测量；不支持，见图 52。

假设3个通道以不同的采样率进行测量:

m1:10Hz采样率的双精度浮点数

m2:20Hz采样率的短整数

m3:30Hz采样率的长整数

逻辑表示:(时间值t1,t2,...代表在时间上以1/60s为一个时间间隔的等距点)

时间	MQ1	MQ2	MQ3
t1	m1v1	m2v1	m3v1
t2			
t3			m3v2
t4		m2v2	
t5			m3v3
t6			
t7	m1v2	m2v3	m3v4
t8			
t9			m3v5
t10		m2v4	
t11			m3v6
t12			
t13	m1v3	m2v5	m3v7
...			

因为这些值来自测量仪器,所以如果这些值被写入到文件中,那么就会发现以下的文件结构:

m1v1	m1v2	m3v1	m3v2	m2v2	m3v3	m1v2	m2v3	m3v4	>>
>>	m3v5	m2v4	m3v6	m1v3	m2v5	m3v7	...	...	...

这样的一组数据不能依据混合模式存储概念保存在外部二进制文件中。但是注意,ATF的组成文件可以采用这样的文件结构,像5.7.22节给出的例5一样的文件结构。

图52 均匀测量

6.4.7 与 ATF 组件的联系

混合模式存储概念和 ATF/CLA 组件元素(见第 10 章)关系紧密。在 ATF 中,二进制文件保存海量数据,然而,描述性数据存储于 RDB 中(单独保存在 ASCIIATF 文件中)。

然而,还是有些不同:

- 当 ATF/CLA 规范和一个块中相同局部列值的位置有关时,混合模式存储规范比 ATF/CLA 规范灵活性小。6.4.6 节的例 5 可能不会使用混合模式存储,尽管它描述的可能是一个有效的 ATF/CLA 组件文件。这是因为 AoExternalComponent 可能只确定派生自基本属性 'value_offset' 的一个属性值,而 ATF/CLA 可能有一组 VALOFFSETS;
- 混合模式存储规范比 ATF/CLA 规范灵活性大,因为在 ATF/CLA 规范中,它允许将值分割并将他们分配给物理文件。它们中的每一个再被确定为一个外部组成文件,局部列也可能涉及由派生自基本属性 'ordinal_number' 的属性值辨别的多个外部组成文件。ATF/CLA 规范要求所有的局部列的值放在相同的组成文件中;
- 根据 ATF/CLA,标记不会存储在二进制组成文件中;混合模式存储要求,当值存储在二进制文件时,标记才能存储在二进制文件中。

6.4.7.1 AoExternalComponent 属性到 ATF/CLA 元素的映射

表 61 描述了 ATF/CLA 元素和混合模式存储规范之间的对比。

表61 AoExternalComponent 属性到 ATF/CLA 元素的映射

组件结构中的ATF/CLA元素	AoExternalComponent属性
标识符	filename_url
类型规范	value_type
长度规范	component_length
DESCRIPTION	description
INIOFFSET	start_offset
BLOCKSIZE	block_size
VALOFFSETS	value_offset(note:onlyoneallowed)
VALPERBLOCK	valuesperblock
N/A(只允许一个二进制文件)	ordinal_number

6.4.8 混合模式服务器

图 53 描述了混合模式服务器的模型。

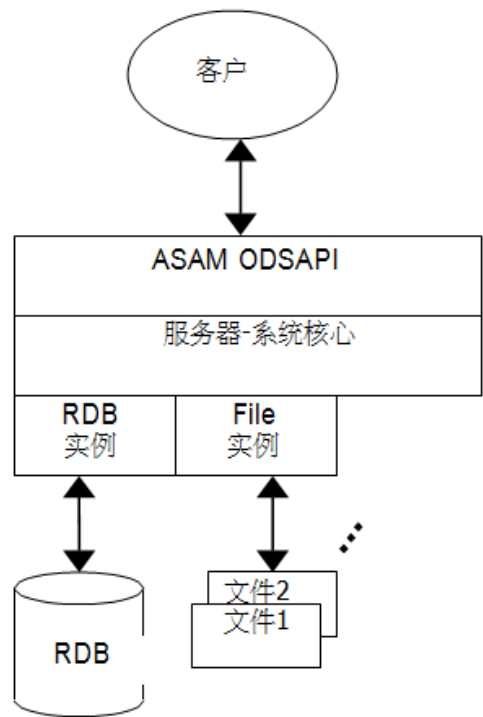


图53 混合模式服务器架构

- 混合模式服务器必须满足以下要求：
- 访问数据库可以存取所有 ASAMODS 元素；
 - 在数据库内或在数据库以外的单独文件中存储海量数据，这个行为可以由服务器配置。为了这

个目的，一个特定的会话 **context** 变量可能会由客户端通过 **APIs** 设定，它确定新写入的海量数据应该保留在数据库中还是放在外部文件中；

——**API** 的透明性，因此，客户端不需要关心海量数据应该怎样存储，以及何时检索它们；

——异步访问，服务器必须保证对文件异步访问的同步性。使用 **API** 的不同客户端的访问必须由服务器协调。而不同客户端对相同信息的访问通常由数据库管理系统同步，而在混合模式操作中，是由 **ASAMODS** 服务器负责的。**ASAM ODS** 服务器对不使用 **API** 对文件进行的访问是不能同步的。

6.4.8.1 混合模式服务器操作的用例

(1) 用例 1：创建一个具有初始值的新局部列

- a) 创建带有值的新外部组件文件；
- b) 创建额外的二进制文件（片段），取决于要写入值的数量和外部组成文件的大小。

(2) 用例 2：向局部列追加值

检查现存（最后的）外部组件文件中值的数量和文件大小，并决定是将值追加到现存文件中还是创建一个新的二进制文件（片段）。

(3) 使用案例 3：向局部列插入值

- a) 检查现存的（最后的）外部组件文件中值的数量和文件大小，并决定文件是否能容纳现存文件插入新值后值得总数量；
- b) 如果值可以插入到现存的文件中并且文件不被分开，那么这个文件必须完整读入，包括新值然后写回现有文件；
- c) 如果这个文件不得被分成一个或多个片段，那么受影响的二进制文件（片段）必须完整读入，并且新的文件必须写回磁盘。

(4) 用例 4：修改局部列的值

- a) 对于数值型的数据类型（固定长度），可以直接更新文件而无需移动数据；
- b) 对于字符串，字节流，**blobs**，每次改变字节数，都要根据上面描述的使用案例“向局部列插入值”，先读入，然后写回。

(5) 用例 5：删除局部列的值

- a) 受影响文件中的值被删除；二进制文件必须完整读取，要删除的值必须被移动，必须写回文件；
- b) 如果一个片段中的所有值都被删除，那么外部组件实例必须被删除，局部列中的其他外部组件的序列号必须得更新，局部列实例也必须得相应地得修改。外部组件文件可能被删除；然而，除了删除文件，**ODS** 服务器可能决定保持文件现在的样子，尽管它不再被 **ODS** 数据库引用。

(6) 用例 6：删除整个局部列

- a) 检查外部组件文件是否也保持其他局部列的值。如果是这种情况，要删除值在物理上移动后，整个文件必须完整地读取然后写回。否则，整个外部组件文件可能被删除。相反地，**ODS** 服务器可能决定保持文件现在的样子，尽管它不会再被 **ODS** 数据库引用。

b) 局部列实例和外部组件实例必须删除，它们之间的所有的关系也必须删除。

6.4.8.2 混合模式服务器实施注意事项

混合模式服务器实施注意事项如下：

- 在对外部组件文件进行任何写入操作后，服务器必须保证数据的一致性。就是说，如果客户端要求删除局部列的一些值，而没有向局部矩阵的剩余局部列发出删除命令，那么服务器必须探测检查这个不一致性，必能做“回滚”（保存文件副本或在写入以前检查文件，等）；
- 在修改文件时服务器必须锁住文件（共享锁定—不能对有关文件进行写入访问，只能读取访问）。在“提交”阶段，服务器必须持有对文件的专属副本，必须对有关文件使用排他锁；
- 在发生系统故障后，服务器必须支持部分起动阶段的错误恢复。

6.4.8.3 定义每个片段文件大小的参数

必须引入一个客户端可以修改的服务器参数，这个参数叫做 `EXT_COMP_SEGSIZ (DT_LONG)` 它以字节方式定义了新片段的最大文件的大小。当客户端改这个参数值时，接下来的所有写入操作必须使用新的参数。应该通过服务器配置机制（比如启动文件，启动参数）为参数的默认值设定一个合适值，在服务器起动阶段启用。

6.4.8.4 混合模式服务器的模式转换

为了决定将数据写入外部文件，还是服务器数据库内部存储空间，需要一个额外的参数。这个参数必须作为上下文变量在 `AoSession` 中设定（见 13.2.2 节，`AoSession_SetContext`）。参数是 `write_mode`，其可能值为：`database (default) /file`。

6.4.8.5 对 ODS 安全性的影响

对 ODS 安全性的影响如下：

- 测量权限将被继承；
- 二进制文件的存储必须通过 `ASAMODS`（拷贝）的 API 完成。

7 基础模型

7.1 基础模型介绍

7.1.1 概况

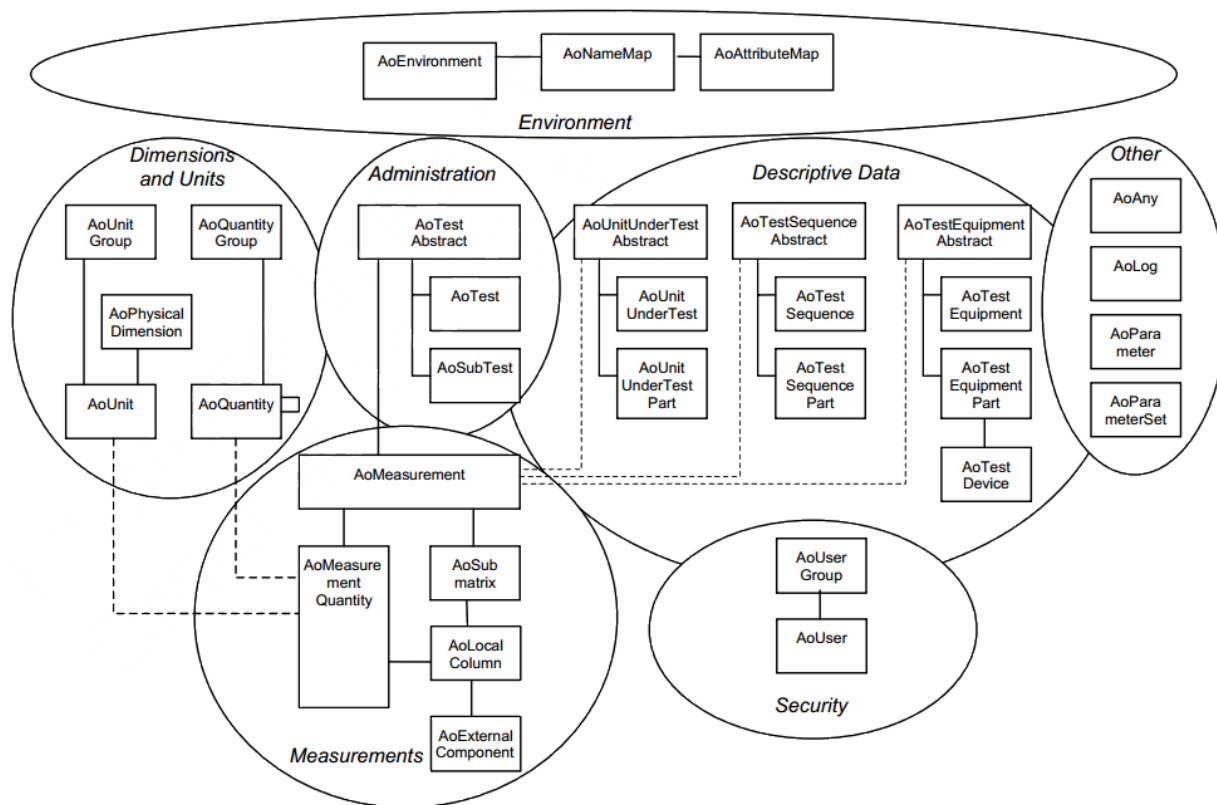


图54 ASAM ODS 基础模型

ASAM ODS 基础模型见图 54，模型包含了下面的基础元素，从这些元素里可以派生出应用元素。这些元素根据它们在自动化及测量系统中的不同用法被分为不同的组。

7.1.2 派生原则

任何一台 ASAM ODS 模型都应该能够根据测试的具体情境在测量执行后纳入测量数据。因为这些数据出现在许多公司或部门，所以无法将它们映射到同一个模型上。再者，事实证明在不同的公司、部门或不同的应用之间交换数据是很有必要的。

为了方便在不同的数据库之间进行数据交换，这些数据库中的数据模型差异不能太大，即使是为了满足某些实际应用或公司的特殊要求而导致数据模型之间存在差异，这种差异也不能太大。

在 ASAM ODS 中，这种差别得到了解决。我们创建了一个基础模型，然后再从这个基础模型派生出其它特殊用途的模型。这些派生类型包括：

- 属性的传递（用作特殊用途的数据从基础模型中获得属性和联系）；
- 规则的传递（基础模型给特殊用途的数据提供规则，后者必须服从）。

这种派生出来的模型通常是在某个方面的典型应用，因此被称作应用模型。它的各个部分被称为应用元素，而基础模型的各个部分被称为基础元素。

请注意，ASAM ODS 中派生和传承是两个不同的概念。想了解更多继承方面的知识，请参照 5.7。

节。

7.1.3 元素属性关系

7.1.3.1 基础元素、属性和关系的名称

值得注意的是，此章中基础模型表明的这些实体的名称要分大小写。它们可能是大写，小写或大小写同时出现。但是我们建议最好是使用此模型说明书中给出的名称形式。

7.1.3.2 应用原素的名称，属性及关系

在一个应用模型中，其应用元素的名称，它们的特性和相互关系是由一串 ISO/IEC 8859-1 (ISO-Latin) 组成的字符。以下几个是特例：

- 名称不能采用空格键开头的字符 (0x09...0x0D, 0x20)；
- 名称不能包含 0x00，因为它代表一串字符的结尾，因此剩下的字符会被漏掉；
- 名称不应包含 0x01...0x08， 0x0E...0x0F，因为这些字符可能会在验证导出的 ATF/XML 文件时造成问题；
- 名称中不能出现与预留的 ATF/CLA 名单中出现的名称相同的关键字，因为如果用同样的名称，那这些它们可能无法被 ATF/CLA 传输（见 9.5 节和 9.13.2 节）；
- 名称不能以数字开头，因为这些字符可能会在分析 ATF/CLA 文件夹的过程中造成问题；
- 应用元素的名称不能以 Ao 开头，因为这些名字为基础元素预留；
- 下面的名词不应被用作应用元素的名称：‘ACLI’、‘ACLA’、‘ACL Template’、‘Initial RightsAt’、‘SecurityLevel’，因为称作这些名称的元素可能会在 ATF 传输的过程中造成问题；
- 应用元素的名称，应用特性的名称和应用关系的名称最多可以由 30 个字符组成（物理存储规定这一限制）。

7.1.3.3 可选属性和必选属性

把一个基本属性或基本关系指定为必选属性意味着一个应用模型必须提供一个应用属性，该应用属性由这个基本属性派生出来。

把一个属性或关系指定为可选属性或可选关系意味着相应的应用元素的实例可以省略这个属性或关系。（不给该属性提供值，没有相关实例存在此关系）

在说明一个基础元素的基本属性或关系时可能会出现表 62 中下列组合：

表62 必选属性和可选属性

必选属性	可选	说明
是	否	每个包含这个基础元素的应用元素的应用模型必须定义一种从这个基本属性或关系中派生出来的应用属性或关系
是	是	每个包含这个基础元素的应用元素的应用模型必须定义一种从基本属性或关系中派生出来的应用属性或关系。该应用元素的实例可以但不必要为那个应用属性或关系提供值。
否	否	该基础元素可以但不必要定义一种从基本属性或关系中派生出来的应用属性或关系。如果它定义了此种应用属性或关系，该应用元素的每一次实例都必须为

必选属性	可选	说明
		应用属性或关系提供值。
否	是	一个包含这个基础元素的应用元素的应用模型可以但不必要定义一种从基本属性或关系中派生出来的应用属性或关系。如果它定义了此种应用属性或关系，该应用元素的实例可以但不必要为那个应用属性或关系提供值。

7.1.3.4 SET 或 LIST 型的属性和关系

在属性或关系是 SET [..] 或 LIST [..]的情况下它没有可选关键字。在某个实例中是否必须给予这个属性一个值，根据括号中的具体情况而定)：

- 一个定义为 LIST [0]的应用属性被视为一个可选属性，实例不必要为它提供值(尽管它们可以)。
- 如果一个客户端应用程序试图存取这些属性的值，它会发现这样的清单并不存在，或者找到的是
一张没有内容的空单子；
- 一个被定义为 LIST [1...]的应用属性要求每一个实例至少有一个该属性的值。

7.1.4 ASAM ODS 基础模型和基于 ACL 的安全概念

基于 ACL 的安全概念在第二章中已做详细讲解。该章陈述了建立在存取控制表基础上的复杂的安全机制可以添加到一个 ASAM ODS 数据库(尽管这完全没必要)。假如这个安全机制添加到一个 ASAM ODS 数据库上，除了有应用元素的实例中存储的信息，同时还会有些要求用来控制存取应用属性的实例或值的附加信息。

这些附加信息融合了应用模型和实例信息，尚未被认为可以变成一组基础元素，而是在一组实体中被分为几组，每组包含几项。从这些实体中无法派生出应用元素，因此也无法成为应用模型的一部分。然而这些实体在物理存储中和 ATF 文件夹中能找到，而且我们可以通过 ASAM ODS 的应用程序接口读取和写入它们的项目值。

这些实体被包含在基础模型的正式 EXPRESS 说明中，而且在这个部分接下来的段落中会有描述。

与安全性相关的附加信息包括五个实体，每个实体各由一组项目组成。这些实体的数据集会在有必要的时候出现。

SecurityLevel（安全等级）实体用来说明一个应用因素的安全等级。安全等级构成见表 63。

表63 SecurityLevel（安全等级）

appl_element_name	(DT_STRING) 安全等级已标明的应用元素的名称
level	(DT_STRING) 安全等级的位模式，它由属性安全、实例安全和元素安全中的一项或多项组成，各项的位状态为{位 2 位 1 位 0}。

- “element security 元素安全” 允许定义对一个应用元素的所有实例都有效的存取权限。
 - “instance security 实例安全” 允许分别对一个应用元素的各个实例定义存取权限。
 - “attribute security 属性安全” 允许在一个应用元素的实例范围内分别对每个应用程序属性的值定义存取权限。
- 安全等级实体的一个数据集最多可以提供给一个应用元素。

ACLI 实体用来为一次实例存储一次 ACL-存取记录（在实例安全性中）。它明确界定用户组存取一次实例的权限，ACLI 实体由以下几部分组成，见表 64：

表64 ACLI 实体组成

instance_id	(DT_LONGLONG) 为实例的标识符（一个唯一的号码），其存取权限已界定
appl_element_name	(DT_STRING) 为实例所属的应用元素的名称
users	(AoUserGroup) 为享有存取权的用户组
rights	(DT_SHORT) 为存取权位模式，它由{赋予 删除 插入 升级 读取}中的一项或多项组成，每项的位模式为{位 4 位 3 位 2 位 1 位 0}

ODS 数据存储中，提供给实例的 ACLI 实体数据集在数量上是有求必应。

ACLA 实体用来为应用元素（在元素安全性中）或应用程序属性（在属性安全性中）存储 ACL 存取记录。它为用户组界定存取一个应用元素（元素安全性）的所有实例及分别存取其中各个属性的值的权限（属性安全性），它由以下几部分组成，见表 65。

表65 ACLA 实体组成

attribute_name	(DT_STRING) 为应用程序属性的名称，其存取权限已界定，若丢失或上面没有内容，“element security” 则被当成 “attribute security”
appl_element_name	(DT_STRING) 为应用元素的名称，应用元素从属于它，其存取权限已分别界定（元素安全性）
users	(AoUserGroup) 为享有存取权的用户组
rights	(DT_SHORT) 为存取权限位模式，它由{赋予 删除 插入 升级 读取}中的一项或多项组成，各项位模式分别为{位 4 位 3 位 2 位 1 位 0}

在 ODS 数据存储中，给应用元素或应用程序属性的 ACLA 实体提供其要求的数量的数据集。

ACL Template 实体被用来存储一个 ACL 模板，该模板用来为将要产生的实例自动生成 ACL-入口；它只适用于已声明 “instance security” 的应用程序属性的实例。它界定用户组存取某个具体应用元素的实例的权限，ACL Template 实体组成见表 66。

表66 ACL Template 实体组成

instance_id	(DT_LONGLONG) 为储存这个 ACL 模板的实例的标识符（为一个唯一的号码）。该标识符可被忽略，ACL 模板则会直接被存储在应用元素名称的应用元素上
appl_element_name	(DT_STRING) 为实例所属的应用元素的名称，它存储这个 ACL 模板
ref_appl_elem_name	(DT_STRING) 为应用元素的名称，其（新生成的）实例通过从这个 ACL 模板获取用户组及存取权限获得 ACL 入口
rights	(DT_SHORT) 为存取权限位模式，它由{赋予 删除 插入 升级 读取}中的一项或多项组成，各项的位状态分别是{位 4 位 3 位 2 位 1 位 0}

在 ODS 数据存储中，给 ACL 模板提供其要求数量的数据集（而且存储到任何实例或应用元素上）。

InitialRightsAt 实体用来说明应用程序关系应该遵循什么原则来找到 ACL 模板，有了这个模板，

ODS 服务器就能在生成实例的过程中自动生成一个 ACL 入口。它只适用于已说明“instance security”的应用元素，组成见表 67。

表67 InitialRightsAt 实体

appl_element_name	(DT_STRING) 为应用元素的名称
attribute_name	(DT_STRING) 为以上应用元素的应用关系的名称；对于应用软件元素 appl_element_name 将生成的一个实例，这种关系引用一个（已存在的）实例并且暗中参照它的应用元素；在此可以找到一个有用的 ACL 模型，在此模型上可以生成一个新实例的 ACL 入口

InitialRightsAt 实体的一个数据集最多可以提供给一个应用程序关系。

值得注意的是，ASAM ODS 基础模型中的这些可以被派生出的实体（（因此可以从中产生应用元素））与以上提及的五种不能派生出应用元素的实体之间是有区别的。这一点在它们的名称中得到了体现。

如果基础模型的实体能派生出应用元素，它就只能被命名为“基础元素”。因此上面提到的五种实体不是基础元素。

只有基础元素可以具有基本属性和基本关系。因此，上面提到的五中实体不具有基本属性或基本关系。它们由“条目”构成。

只有应用元素可以有实例。因此上面提到的五中实体没有实例，它们的值在“数据集”中提供。

有关 ASAM ODS 安全概念及存取控制机制和模型的使用详见第 6 章。

7.2 环境基础元素的描述

7.2.1 AoEnvironment (BID=1)

AoEnvironment 的基本属性和关系见表 68。

表68 AoEnvironment 的基本属性和关系

基本属性	意义
id	DT_LONGLONG 必选属性（应用模型中需要），应用元素实例唯一的标识，必须是正数（如 1,2,...）
name	DT_STRING 必选属性（应用模型中需要）ASAM ODS 服务器的名称
description	DT_STRING 可选 对 ASAM ODS 服务器的描述
version	DT_STRING 可选 ASAM ODS 服务器的版本
version_date	DT_DATE 可选 版本更新日期

mime_type	DT_STRING 可选 实例的文件类型
external_references	DT_EXTERNALREFERENCE LIST[0: ?] 外部信息的引用
objecttype	DT_LONGLONG 可选 包括实例所属的应用元素的 id（子类和超类）。这里不能用，因为继承机制对种类 AoEnvironment 的应用元素无用。
meaning_of_aliases	DT_STRING LIST[0: ?]
max_test_level	DT_LONG 可选
base_model_version	DT_STRING 可选 ODS 基础模型的名称，是由“asam”后接基础模型的编号组成的字符串（目前版本为“asam30”）。
pplication_model_type	DT_STRING 可选 可以包含多个名称，中间用逗号隔开。
application_model_version	DT_STRING 可选 任何运营商提供的名称
timezone	DT_STRING 可选 环境时区根据全球时间设置时间日期，时差用 GMT 表示（如，“GMT-07: 00”，“GMT+05: 45”）
关系	
entity_mapping	INFO_FROM AoNameMap 的集合 SET[0: ?]
tests	CHILD AoTest 的集合 SET[0: ?]
uuts	CHILD AoUnitUnderTest 的集合 SET[0: ?]
equipments	CHILD AoTestEquipment 的集合 SET[0: ?]
sequences	CHILD AoTestSequence 的集合 SET[0: ?]

在应用模型中，从基础模型中至多可以派生出一个应用元素（单派生）。

对于这种实体，任何一次数据存储中最多可能产生一个实例。

从基本属性“**meaning_of_aliases**”中派生出的应用元素的值可能有一系列的描述(一个有序列表),逐一解释每一个别名所代表的意义。

这些描述可以用来说明一个别名组的语言,该语言在描述的结尾以部分字符串标识,以逗号与前面的描述隔开,后面可以接一个或两个空格(作为空白字符)。

这个部分字符必须遵循“**language=<langcode>**”的语法,根据 ISO 639-1, **<langcode>**是一个编码。

使用计算机语言编码描述的例子有:

“**language=de**”, “**testbench,language=en**”, “**test, analysis,language=en**”

不使用计算机语言编码描述的例子有:

“**testbench**”, “**: test, analysis**”, “**Test,language**”, “**language=en**”

除了基本关系“**entity_mapping**”被用作别名外,只有下面这四个层次列表的最高级的实体被纳入以下集合:

- AoTest**;
- AoUnitUnderTest**;
- AoTestEquipment**;
- AoTestSequence**。

在“**entity_mapping**”中,所有的别名对应都被已被纳入。该集包括每个实体(应用元素)一个**AoNameMap**。客户端应用软件中的别名索引可以用来从给定列表中选择合适的别名,这样能很方便切换语言版本。

在别名列表中的每一个位置都应该提供给“**meaning_of_aliases**”一串字符,这样可以将名称映射到不同语言上。见 5.10.4 节多语种支持。

“**Max_test_level**”属性显示试验层次结构中的级别数。

(可选)属性“**application_model_type**”在环境内部标明应用模型的类型。一种类型用一个或多个关键字标识。若有多个关键字,每个关键字之间则用逗号隔开。关键字通常是以该公司或该机构来定义并且以它们的名称开头,后接一个下划线,再接字母和数字,反映该应用模型的基本特征。也可以在关键字里加入版本信息,用分号隔开。版本信息由字母和数字及小圆点组成(**a..z, A..Z,0..9,'.**)。

以“**ASAM**”开头的关键字预留给**ASAM ODS**的定义。目前**ASAM ODS**预先定义的关键字有:

其它关键字的例子有:“**ISO_VSIM**”、“**AVL_ENGINE**”、“**NI_TDM**”, ...。

带版本的关键字有:“**ASAM_GEOMERTY; 2**”、“**ASAM_NVH; 4**”、“**AVL_ENGINE; PUMA**”。

7.2.2 AoNameMap (BID=46)

AoNameMap基本属性及关系见表69。

表69 AoNameMap 基本属性及关系

基本属性	意义
id	DT_LONGLONG 必选属性(应用模型中要求) 应用元素实例唯一的id,必须是正数(1,2,...)

name	DT_STRING 必选属性 （应用模型中要求） 实例的名称
description	DT_STRING 可选 实例描述文本
version	DT_STRING 可选 实例版本
version_date	DT_STRING 可选 版本变更日期
mime_type	DT_STRING 可选 实例的文件类型
external_references	DT_EXTERNALREFERENCE LIST[0: ?] 引用外部信息
objecttype	DT_LONGLONG 可选 包含实例所属的应用元素（子类或超类）的 ID。
entity_name	DT_STRING 必选属性 （应用模型中要求） 应用元素的名称
alias_names	DT_STRING LIST[0: ?] 必选属性（应用模型中要求） 应用元素的替代名称
关系	
environment	AoEnvironment 的信息 entity_mapping 的相反关系
attribute_mapping	AoAttributeMap 的子集 SET[0: ?] 必选属性 （应用模型中要求）

在应用模型中，从此基础元素中只能派生出一个应用元素（单派生）

从基础元素派生出的应用元素的每个实例中，（基本属性'alias_names'给出）一系列的替代应用元素名称用来命名一个应用元素，因此可以为应用元素编出一系列的别名列表。这张列表允许应用软件在不同的语言版本之间进行切换。

这张列表的长度应该和 AoEnvironment 中 “meaning_of_aliases” 的列表长度相同。

每个属性 resp。来自 AoAttributeMap 的应用元素的一个实例的应用元素的关系可（由基本关系“attribute_mapping”）提供。这包含这个应用元素的属性 resp.关系。

7.2.3 AoAttributeMap (BID=47)

AoAttributeMap 的基本属性和关系见表 70。

表70 AoAttributeMap 的基本属性和关系

基本属性	意义
id	DT_LONGLONG 必选属性应用模型中要求) 应用元素实例唯一的 id，必须是正数（1,2, ..）
name	DT_STRING 必选属性 （应用模型中要求） 实例名称，它在层次列表中其前身范围内是唯一的。
description	DT_STRING 可选 对实例描述性的文字
version	DT_STRING 可选 实例的版本
version_date	DT_DATE 可选 版本变更日期
mime_type	DT_STRING 可选 实例的文件类型
external_references	DT_EXTERNALREFERENCE LIST[0: ?] 引用外部信息
objecttype	DT_LONGLONG 可选 包含实例所属的应用元素（子类或超类）的 id
attribute_name	DT_STRING 必选属性（应用模型中要求） 应用属性名称
alias_names	DT_STRING 必选属性 （应用模型中要求） 应用属性的别名
关系	
name_mapping	父类 AoNameMap 属性映射的相反关系 必选属性 （应用模型中要求）

在一个应用模型中，从这个基础元素中只能派生出一个应用元素（单派生）。

从这个基础元素中引派生出来的应用元素的每一个实例包含一个应用属性或关系的一系列的别名

（通过基本属性 “alias_names”），因此可以编出一个别名的列表。

这个别名列表的长度应该和 AoNameMap 中列表的长度相同。

7.3 基础元素尺寸和单位的描述

下面的元素用来储存单位信息：

- AoQuantity;
- AoUni;
- AoPhysicalDimension;
- AoQuantityGroup;
- AoUnitGroup。

7.3.1 AoQuantity (BID=11)

AoQuantity的基本属性和关系见表71。

表71 AoQuantity 的基本属性和关系

基本属性	意义
id	DT_LONGLONG 必选属性（应用模型中要求） 应用元素实例唯一的 id 必须是正数（1,2, ..）
name	DT_STRING 必选属性 （应用模型中要求） 实例名称
description	DT_STRING 可选 对实例描述行的文本
version	DT_STRING 可选 实例版本
version_date	DT_DATE 可选 版本变更日期
mime_type	DT_DATE 可选 实例的文件类型
external_references	DT_EXTERNALREFERENCE LIST[0: ?] 引用外部信息
objecttype	DT_LONGLONG 可选 包含实例所属的应用元素（子类或超类）的 id。

default_mq_name	DT_STRING 可选 为测量数量缺省名称，当该测量数量没有自己明确的名称时，就可以使用缺省名称。
default_datatype	数据类型_枚举型别 可选 缺省数据存储模式 注意：数据类型不能是 DT_ID 或 DT_UNKNOWN
default_rank	DT_LONG 可选 缺省张量阶和值维度
default_dimension	DT_LONG LIST[0: ?] 每个阶的缺省值
default_type_size	DT_LONG 可选 缺省值长度界限，例如最大值，一个字符串或字节流的长度。
关系	
default_unit	INFO_TO AoUnit 可选
successors	子类 SET[0: ?] OF AoQuantity
groups	INFO_REL SET[0: ?] OF AoQuantityGroup INVERSE FOR quantities
measurement_quantities	INFO_FROM SET[0: ?] OF AoMeasurementQuantity INVERSE FOR quantity
predecessor	父类 SET[0: ?] OF AoQuantity INVERSE FOR successors

在应用模型中，从基础元素中只能引派生出一个应用元素（单派生）。

“数量”是一个命名的测试变量，具有不同的特征，如：

——单位，物理意义；

——值的数据类型（实型，整型，布尔型，字符串，矩阵以及其它类型）；

一个数量不是一个测量通道，而是描述一个物理现象。

示例：数量。

- 扭矩（实型）；
 - 时间（实型）
 - 估计温度范围 { “冷”，“暖”，“热” }；
 - 点火装置开（布尔）；
 - 点火特性曲线（适用于工作点）（实际值的矩阵）；
- 一个数量可以被许多测量数量使用，即使这些数量属于同一个测量。

数量之间可以以一种层次的关系排序（使用继任者/前任属性），有些数量可能从来没分配给任何“测量数量”。

示例：数量层次结构见表 72。

表72 数量层次结构

数量名称	继任者	前任	测量数量
m	CO	-	否
CO	CO 之前， CO 之后	m	否
0 之前	-	CO	是
CO 之后	-	CO	是

所有四种情况都是同一个应用元素的实例（例如，“数量”），只有“CO 之前”和“CO 之后”被测量数量使用，这样可以从这些数量中得到名称、单位等的缺省值。

请注意有些数量没有单位，如字符串，布尔量，百分比等等，通常会用一个“空”单位（如用“—”）来达到此目的。

7.3.2 AoUnit（BID=13）

AoUnit 的基本属性和关系见表 73。

表73 AoUnit 的基本属性和关系

基本属性	意义
id	DT_LONGLONG MANDATOR ATTRIBUTE（应用模型中要求） 应用元素实例唯一的 id，必须是正数（1,2, ..）
name	DT_STRING 必选属性（应用模型中要求） 实例名称
description	DT_STRING 可选 对实例进行描述性的文本
version	DT_STRING 可选 实例版本
version_date	DT_DATE 可选

	版本变更日期
mime_type	DT_STRING 实例的文件类型
external_references	DT_EXTERNALREFERENCE LIST[0: ?] 引用外部信息
objecttype	DT_LONGLONG 可选 博阿含实例所属的应用元素（子类或超类）的 id
factor	DT_DOUBLE 必选属性应用模型中要求） 获得国际标准单位的因素
offset	DT_DOUBLE 必选属性（应用模型中要求） 获得国际标准单位的偏移量
关系	
phys_dimension	INFO_TO AoPhysicalDimension 必选属性（应用模型中要求）
groups	NFO_REL SET[0: ?] OF AoUnitGroup INVERSE FOR units
measurement_quantities	DET[0: ?] OF AoMeasurementQuantity 必选属性应用模型中要求） INVERSE FOR unit
parameters	INFO_FROM SET[0: ?] OF AoParameter INVERSE FOR unit
quantities	INFO_FROM SET[0: ?] OF AoQuantity INVERSE FOR default_unit

在应用模型中，从基础元素中只能派生出一个应用元素（单派生）。

因子是（国际标准单位值）当（此单位值）= 1。

偏移量是（国际标准单位值）当（此单位值）= 0。

（此单位值）乘以（因子）= （国际标准单位值）。

（此单位值）加上（偏移量）= （国际标准单位值）。

（此单位值）乘以（因子）加上（偏移量）= （国际标准单位值）。

示例：从非国际单位值换算到国际单位值见表 74。

表74 从非国际单位值换算到国际单位值

(4200) 1/分 = (4200 *因子) 1/秒 = (70) 1/秒	[因子=0.016667]
(20) 摄氏度 = (20+偏移量) 开氏度 = (293.15) 开氏度	[偏移量 = 273.15]
(68) 华氏度 = (68 *因子 + 偏移量) 开氏度 = (293,15) 开氏度	[因子 = 0.55555]
	[偏移量 = 255.372]
(200) 毫米 = (200 *因子) = 0.2 米	[因子 = 0.001]

这个实体说明了测量实体被测量时使用的单位。当物理维度恒定不变时（如体积），单位则可能会变（如“克”，“毫克”，“千克”，“磅”）“单位”指同样的维度，可以通过“单位偏移量”和“单位因子”的方式相互转换。两个单位可以表示同一个物理维度，但前提是两个单位之间可以相互转换。

7.3.3 AoPhysicalDimension (BID=15)

AoPhysicalDimension 的基本属性和关系见表 75。

表75 AoPhysicalDimension 的基本属性和关系

基本属性	意义
id	DT_LONGLONG 必选属性（应用模型中要求） 应用元素实例唯一的 id 必须是正数（1,2,...）
name	DT_STRING 必选属性（应用模型中要求） 实例名称
description	DT_STRING 可选 对实例进行描述性的文本
version	DT_STRING 可选 实例版本
version_date	DT_STRING 可选 版本变更日期
mime_type	DT_STRIN 可选 实例的文件类型
external_referen ces	DT_EXTERNALREFERENCE LIST[0: ?] 引用外部信息
objecttype	DT_LONGLONG 可选 包含实例所属的应用元素（子类或超类）的 id

基本属性	意义
length_exp	DT_LONG 必选属性（应用模型中要求） 长度指数分母
mass_exp	DT_LONG 必选属性应用模型中要求） 体积指数分母
time_exp	DT_LONG 必选属性（应用模型中要求） 时间指数分母
current_exp	DT_LONG 必选属性（应用模型中要求） 电流指数分母
temperature_exp	DT_LONG 必选属性应用模型中要求） 温度指数分母
molar_amount_exp	DT_LONG 必选属性（应用模型中要求） 摩尔量指数分母
luminous_intensity_exp	DT_LONG MANDATORY ATTRIUTE（应用模型中要求） 光指数分母
length_exp_den	DT_LONG 可选 分母（长度） DT_LONG
mass_exp_den	可选 分母（体积） DT_LONG
time_exp_den	可选 分母（时间） DT_LONG
current_exp_den	可选 分母（温度） DT_LONG
temperature_exp_den	OPTIOANL 分母（温度） DT_LONG
molar_amount_exp_den	可选 分母（摩尔量） DT_LONG
luminous_intensity_exp_den	可选 分母（光）

基本属性	意义
关系	
units	INFO_FROM SET[0: ?] OF AoUnit INVERSE FOR 物理维度

在应用模型中，从这个基础元素中只能派生出一个应用元素（单派生）

一个物理维度由国际标准基本维度的七个维度指数来代表，分别是长度，体积，时间，温度，电流，摩尔数，发光强度（以国际标准基本单位米，千克，秒，开，安，摩尔，坎德拉来测量）。通常指数中的很大一部分都为 0，特别是无量纲单位（如“%”）指数都为零。

请注意可能存在几个物理维度具有同一组指数的情况。

示例：示例：相似的物理单位。

下面的物理维度有指数集：

- “长度₂ * 体积₁ * 时间⁻²”；
- “能量”，使用单位为“牛”，“千瓦时”；
- “扭矩”，使用单位为“牛”，“Nmm”……。

下面的物理维度有个普通的零指数集：

- “比例”，使用单位为“百分比”，“百万分比”；
- “角度”，使用单位为“角度”，“弧度”。

大多数情况下，维度指数为整数，即它们的分母等于 1。当分母省略时，则自动假设为 1。

7.3.4 AoQuantityGROUP (BID=12)

AoQuantity 组的基本属性和关系见表 76。

表76 AoQuantity 组的基本属性和关系

基本属性	意义
id	DT_LONGLONG 必选属性（应用模型中要求） 应用模型实例唯一的 id，必须是正数（1, 2, ...）
name	DT_STRING 必选属性（应用模型中要求） 实例名称
description	DT_STRING 可选 对实例进行描述性的文本
version	DT_STRING 可选 实例日期
version_date	DT_DATE

	可选 版本变更日期
mime_type	DT_STRING 可选 实例文件类型
external_references	DT_EXTERNALREFERENCE LIST[0: ?] 引用外部信息
objecttype	DT_LONGLONG 可选 包含实例所属的应用元素（子类或超类）的 id。
关系	
quantities	INFO_REL SET[0: ?] OF AoQuantity 必选属性（应用模型中要求）

在应用模型中，从这个基础元素中只能派生出一个应用元素（单派生）

一个“数量组”描述一个“数量”集的特征。一个“数量”可以属于好多“数量组”，由此可能产生一个关系网。

示例：分组的数量。

如其层次名称结构显示，数量“F.FL”，属于力（“F”）及位置“左前”（“FL”）。

此外，它还可能属于数量组：“长力（length force）”，“计算量”，“过滤量”及“左前方现象”。

7.3.5 AoUnitGROUP（BID=14）

AoUnitGroup 的基本属性和关系见表 77。

表77 AoUnitGroup 的基本属性和关系

基本属性	意义
id	DT_LONGLONG 必选属性应用程序中要求) 应用元素实例唯一的 id，必须是正数（1,2, ..）
name	DT_STRING 必选属性 （应用程序中要求） 实例的名称
description	DT_STRING 可选 对实例描述性的文字
version	DT_STRING 可选 实例的版本
version_date	DT_DATE

	可选 版本变更日期
mime_type	DT_STRING 可选 实例的文件类型
external_references	DT_EXTERNALREFERENCE LIST[0: ?] 引用外部信息
objecttype	DT_LONGLONG 可选 包含实例所属的应用程序（自己或超类）的 id
关系	
units	INFO_REL SET[0: ?] OF AoUnit 必选属性 （应用程序中要求）

在应用模型中，从基础元素中只能派生出一个应用元素（单派生）。

AoUnitGROUP 允许根据自由标准对“单位”进行分组。

示例：“MKS”、“USA”、“outdated”。

7.4 测量基础元素的描述

下面的元素被用来存储从测量中获得的海量数据：

- AoMeasurement;
- AoMeasurementQuantity;
- AoSubMatrix;
- AoLocalColumn;
- AoExternalComponent。

7.4.1 AoMeasurement (BID=3)

AoMeasurement的基本属性和关系见表78。

表78 AoMeasurement 的基本属性和关系

基本属性	意义
id	DT_LONGLONG 必选属性（应用模型中要求） 应用模型实例唯一的 id，必须是正数（1, 2, ..）
name	DT_STRING 必选属性（应用模型中要求） 实例名称，在分层中其前身范围内唯一。
description	DT_STRING 可选

	对实例进行描述性的文本
version	DT_DATE 可选 实例日期
version_date	DT_DATE 可选 版本变更日期
mime_type	DT_STRING 可选 实例文件类型
external_references	DT_EXTERNALREFERENCE LIST[0: ?] 引用外部信息
objecttype	DT_LONGLONG 可选 包含实例所属的应用元素（子类或超类）的 id。
measurement_begin	DT_DATE 可选 测量开始时间标记
measurement_end	DT_DATE 可选 测量结束时间标记
关系	
test	父类 AoTest 或 AoSubTest 必选属性（应用模型中要求）
units_under_test	INFO_REL SET[0: ?] OF AoUnitUnderTest 和/或 AoUnitUnderTest 测试可能包含这两个基本类型的实例
sequences	INFO_REL SET[0: ?] OF 测试序列和/或测试序列部分 该集可能包含这两个基本类型的实例
equipments	INFO_REL SET[0: ?] OF AoTestEquipment 和/或仪器 该集可能包含这所有三种类型的实例
measurement_quantities	子类 SET[0: ?] OF AoTest 数量 必选属性（应用模型中要求）
submatrices	子类 SET[0: ?] OF Ao AoSubMatric 必选属性（应用模型中要求）

在应用模型中，从这个基础元素中只能派生出一个应用元素（单派生）。

AoMeasurement 是与测试相关所的数据的连接点。

AoMeasurement 引用：

——该测试描述性的数据。它包括使用的设备及其设置，被测试的对象及配置和测试期间执行的步骤；

——测试的结果数据。它包含所有的测试结果，计算结果，状态结果，最后还有预先定义的设定值。

另外，它们还包括对所有那些被观察到的，计算过的或作为设定值变量的数量的描述。

7.4.2 AoMeasurementQuantity (BID=4)

AoMeasurementQuantity 的基本属性和关系见表 79。

表79 AoMeasurementQuantity 的基本属性和关系

基本属性	意义
id	DT_LONGLONG 必选属性应用程序中要求) 应用元素实例唯一的 id，必须是正数 (1, 2, ...)
name	DT_STRING 必选属性 (应用模型中要求) 实例名称，在分层中其前身范围内唯一。
description	DT_STRING 可选 对实例进行描述性的文本
version	DT_STRING 可选 实例的版本
version_date	DT_DATE 可选 版本变更日期
mime_type	DT_STRING 可选 实例的文件类型
external_references	DT_EXTERNALREFERENCE LIST[0: ?] 引用外部信息
objecttype	DT_LONGLONG 可选 包含实例所属的应用程序 (自己或超类) 的 id
datatype	日期类型枚举 必选属性 (应用程序中要求)

rank	DT_LONG 可选 张量的阶，值维度排名
dimension	DT_LONG LIST[0: ?] 每个等级的值的数目
type_size	DT_LONG 可选 值的长度界限，例如字符串的最大长度
minimum	interpolation_enum 可选 在内插过程中，需要时使用的类型
minimum	DT_DOUBLE 可选 最小值
maximum	DT_DOUBLE 可选 最大值
average	DT_DOUBLE 可选 平均值
standard_deviation	DT_DOUBLE 可选 标准偏差
关系	
quantity	INFO_TO AoQuantity 必选属性（应用模型中要求）
unit	INFO_TO AoUnit 必选属性（应用模型中要求）
local_columns	INFO_FROM SET[0: ?] OF AoLocalColumn 必选属性（应用模型中要求）
channel	INFO_TO AoTestEquipment 或 AoTestEquipmentPart 或 AoTestDevice 可选
is_scaled_by	INFO_FROM LIST[0: ?] OF AoMeasurementQuantity 用作信元缩放，对于多维度缩放（等级>1），最右边的指数在最内层的循环中递增
measurement	父类

	AoMeasurement INVERSE FOR measurement_quantities 必选属性（应用模型中要求）
scales	INFO_TO SET[0: ?] OF AoMeasurementQuantity INVERSE FOR 缩放尺度

在应用模型中，从这个基础元素中只能派生出一个应用元素（单派生）。

一个“测量数量”进一步描述了一个值矩阵的列。它表示了在“测量”过程中使用的“数量”。指定的“测量量”在一次“测量”中仅出现一次。“测量量”可能是相互关联的，即一个数量可能取决于另外一个数量。尤其是对于时间流的测量，这种关系用来测量数量和相应的时间量。

示例：测量数量之间的相互依赖关系。一个测量数量可能依赖于：

——“满负荷”测量中的扭矩；

——在一个测试周期中的时间序列测量时间。

测量数据可能与测试设备（或其组成部分之一）有关系。这个基本关系被命名为“通道”，并应说明测试设备的什么部分（如多通道系统的数据采集通道，使用燃料的规模）最终产生了测量值。

“minimum”、“maximum”、“average”和“standard_deviation”这些基本属性已经提供，因此在不需要浏览所有相关地方列的情况下就可以得到这个测量量值的快捷信息。

以下规则适用于：

——由客户端计算最大值，最小值，算术平均，其每次输入数据或删除数据的所有相关本地列或修改这个测量量的本地列的数据范围之内的所有值的标准偏差。这种计算得来的结果将被放到相应的属性里。服务器不会执行计算，它只提供存储这些信息的可能性；

——这些属性只有当本地列值的标志“AO_AF_VALID”被设置的情况下才能被计算；

——请注意，应用模型不需要定义相应的应用属性，如果定义了，实例则不需要为这些属性提供值。

然而，如果这些属性有值的话，客户端在添加/删除/修改相应的本地列的值时，必须消除这些属性值或重新计算它们。如果属性有值的话，它们必须显示该本地列正确的信息。

7.4.3 AoSubMatric (BID=38)

AoSubMatric 的基本属性和关系见表 80。

表80 AoSubMatric 的基本属性和关系

基本属性	意义
id	DT_LONGLONG 必选属性（应用模型中要求） 应用元素实例唯一的 id 必须是正数（1,2, ..）
name	DT_STRING 必选属性（应用模型中要求）

	实例名称，在分层中其前身范围内唯一。
description	DT_STRING 可选 对实例进行描述性的文本
version	DT_STRING 可选 实例版本
version_date	DT_DATE 可选 版本变更日期
mime_type	DT_STRING 可选 实例的文件类型
external_references	DT_EXTERNALREFERENCE LIST[0: ?] 引用外部信息
objecttype	DT_LONGLONG 可选 包含实例所属的应用程序（自己或超类）的 id
number_of_rows	DT_LONG 必选属性应用程序中要求） 此子矩阵中“测量点”的数目
关系	
local_columns	子类 LIST[0: ?] OF AoLocalColumn 必选属性应用程序中要求）
measurement	父类 AoMeasurement INVERSE FOR 子矩阵 必选属性应用程序中要求）

在应用模型中，从基础元素中只能派生出一个应用元素（单派生）。

7.4.4 AoLocalColumn (BID=39)

AoLocalColumn 的基本属性和关系见表 81。

表81 AoLocalColumn 的基本属性和关系

基本属性	意义
id	DT_LONGLONG 必选属性（应用模型中要求） 应用模型实例唯一的 id，必须是正数（1,2, ..）

name	DT_STRING 必选属性（应用模型中要求） 实例名称，在分层中其前身范围内唯一。
description	DT_STRING 可选 对实例进行描述性的文本
version	DT_STRING 可选 实例版本
version_date	DT_DATE 可选 版本变更日期
mime_type	DT_STRING 可选 实例的文件类型
external_references	DT_EXTERNALREFERENCE LIST[0: ?] 引用外部信息
objecttype	DT_LONGLONG 可选 包含实例所属的应用程序（自己或超类）的 id
global_flag	DT_SHORT 可选 其位具有特殊含义的 2 字节整数，“global_flag”或“flags”必须由实例给出。
flags	DT_SHORT LIST[0: ?] 具有按位含义的 2 字节整数名单，长度必须和“values”的长度相同，而且必要与子矩阵中的值“number_of_rows”一致。“global_flag”或“flags”必须由实例给出。
independent	DT_SHORT 必选属性应用程序中要求： 独立 从属 每个子矩阵中最多只能有个一本本地列可以是独立的
minimum	DT_DOUBLE OPTIONAL 最小值
maximum	DT_DOUBLE 可选 最大值
sequence_representation	序列代表枚举

	必选属性应用程序中要求) 枚举; 下面列出了可能值
generation_parameters	DT_DOUBLE LIST[0: ?] 计算原始的本地列需要的参数列表, 省略了明确的, 隐式的和外部组件的本地列。描述如下。
raw_datatype	数据类型枚举 可选 描述原始数据的类型的可选数据类型的枚举值。必须是数据类型枚举的枚举项之一。
关系	
values	序列值 可选 一个包含每个“测量点”的值的值序列, 值的数目在子矩阵的值“行数”中被给出, 省略了隐式的本地列。
external_component	子类 LIST[0: ?] OF AoExternalComponent
submatrix	父类 AoSubMatric INVERSE FOR 本地列 必选属性应用程序中要求)
measurement_quantity	INFO_TO AoMeasurementQuantity INVERSE FOR 本地列 必选属性应用程序中要求)

在应用模型中, 从基础元素中只能派生出一个应用元素 (单派生)。

一个本地列是一个子矩阵的局部的队列。

关于基本属性 “name (名称)” 有一个限制。本地列的每一个实例指代一个测量数量的实例 (由基本关系 “测量数据” 提供)。一个本地列的实例中的属性 “名称” 的值必须和其相对应的测量数据的基本属性 “名称” 的值相同。(远程过程调用应用程序接口的限制要求这样)。

关于基本属性 “independent (独立)” 有一个限制。只有当相应的测量数量的数据类型可以很容易地用作排序目的时本地列才可以被设为 “independent”。T_SHORT、T_FLOAT、T_BYTE、T_LONG, T_DOUBLE、T_LONGLONG 和 T_DATE 都属于这种情况。这种限制是必要的, 因为值矩阵的行将根据独立列的值排序。

基本属性 “maximum” 和 “minimum” 已被提供, 因此在不需要浏览所有值的情况下就可以快速获取关于这个测量数量的值的信息。下面的规则适用于:

——客户端每次在本地列中储存信息, 删除信息或修改信息时都由它们自己 (重新) 计算该本地列中所有值的最大值和最小值。该计算值将被放到其相应属性内。服务器不执行计算, 它只会提供存储信息的可能性;

——这些属性的计算只有在值的标志“AO_VF_VALID”被设定时才会被执行。（见第 7.4.4 节中对标志的介绍）；

——请注意，应用模型不需要定义相应的应用属性，如果定义了，则实例不需要为这些属性提供值。

然而，如果这些属性有值的话，客户端在添加/删除/修改相应的本地列的值时，必须消除这些属性值或重新计算它们。如果属性有值的话，它们必须显示该本地列正确的信息。

属性“sequence_representation”可以取下列枚举值之一（见表 82），它们被定义为“seq_rep_enum”。

表82 sequence_representation 可以取下列枚举值

explicit	(=0)
implicit_constant	(=1)
implicit_linear	(=2)
implicit_saw	(=3)
raw_linear	(=4)
raw_polynomial	(=5)
formula	(=6) ASAM ODS 5.2.0 版中不用
external_component	(=7)
raw_linear_external	(=8)
raw_polynomial_external	(=9)
raw_linear_calibrated	(=10)
raw_linear_calibrated_external	(=11)

将来 ASAM ODS 会对此枚举进行扩展。

属性“sequence_representation”（属于“seq_rep_enum”类型）的值决定了怎样预见属性 AoLocalColumn 的属性。下表给出了一个概况。

不同值表达的本地列的属性见表 83。

表83 不同值表达的本地列的属性

序列表达	生成参数	值	外部组件
explicit	——	最终值	——
implicit_constant implicit_linear implicit_saw	（生成参数）*1	生成参数*2	——
raw_linear	生成	原值	——
raw_polynomia	参数		
formula	——	——	——
external_component	——	从文件夹中读取的值	外部组件的描述
raw_linear_external raw_polynomial_external raw_linear_calibrated raw_linear_calibrated_externa	生成参数	从文件夹中读取的值（原值）	外部组件的描述

序列表达	生成参数	值	外部组件
注3: *1 由于之前的版本所以被留用。这种情况中, 应该通过“value”属性存取生成参数。			
注4: *2 见第 6 章中允许的有隐式本地列的数据类型。			

产生的(或提供的)值 $x_1, x_2, \dots, x_N$ 中的数字 N 由从相应得子矩阵的基本属性“number_of_rows”中派生出来的属性提供。

如果“sequence_representation”是显式则可以立即从 ASAM ODS 服务器中获得, 如果“sequence_representation”是外部组件则可以立即从外部组件文件夹中获得, 这两种情况都不需要任何服务器的处理。在这两种情况下, 从相应测量数量的属性“datatype”中派生出来的属性可以指定字符串以及数字数据类型 (DT_STRING、DT_SHORT、DT_FLOAT、DT_BOOLEAN、DT_BYTE、DT_LONG、DT_DCOMPLE、DT_EXTERNALREFERENCE)。

由于现在 ASAM ODS 不指定公式语言, 所以值“formula”不应该被指定给“sequence_representation”。

除了这些情况以外, 值可能会在它们的原型代表中被表示为 N 值 $r_1, r_2, \dots, r_N$ (例如像没有缩放的整数) 或者会动态生成。

在这些情况下, 从相应的测量数量的基本属性“datatype”中派生出来的属性可能只能指定数字数据类型, 对应 DT_SHORT, DT_FLOAT, DT_BOOLEAN, DT_BYTE, DT_LONG, DT_DOUBLE, DT_LONGLONG, DT_COMPLEX, DT_DCOMPLEX。

在后面的情况中有生成参数, 它们的数量和意义取决于序列代表的值。假设 M 生成参数 $p_1, p_2, \dots, p_M$ 已给出, 它们表示的含义 (x_n 表示值序列中返回的第 n 个最终值)。

explicit	—— (x_n 可立即获得)
implicit_constant	$x_n = p_1$ (不变值, 偏移)
implicit_linear	$x_n = p_1 + (n-1) * p_2$ (开始值, 增量)
implicit_saw	$x_n = p_1 + ((n-1) \bmod (p_3 - p_1) / p_2) * p_2$
raw_linear	$x_n = p_1 + p_2 * n$ (偏移, 因子)
raw_polynomial	$x_n = p_2 + p_3 * n + p_4 * n^2 + p_5 * n^3 + \dots + p_{n+1} * n^n$
external_component	—— (x_n 可以立即从外部文件夹中获取)
raw_linear_external	$x_n = p_1 + p_2 * n$ (偏移, 因子, 原值从外部文件中读取)
raw_polynomial_external	$x_n = p_2 + p_3 * n + p_4 * n^2 + p_5 * n^3 + \dots + p_{n+1} * n^n$ (原值从外部文件中读取)
formula	——
raw_linear_calibrated	$x_n = (p_1 + p_2 * n) * p_3$ (偏移, 因子, 校准)
raw_linear_calibrated_external	$x_n = (p_1 + p_2 * n) * p_3$ (偏移, 因子, 校准, 原值从外部文件中读取)

“外部组件”属性要求下列几项:

- external_component;
- raw_linear_external;
- raw_polynomial_external;

——raw_linear_calibrated_external。
并包含一个或多个对 AoExternalComponent 的引用。

在所有其它情况下它是空值（可以省略）。

属性“flag”不允许出现在“sequence_representation”是“implicit_constant”、“implicit_linear”或“implicit_saw”的情况中。在所有其它情况下，它可以为每个值持有标志，它们的定义如表 84。

表84 值标志的位编码

缩写	值	描述
AO_AF_VALID	0×01	值有效（“1”），数据生成时通常总是这样；检查之后最终把它们设置为无效（“0”）
AO_AF_VISIBLE	0×02	该值必须用作可视化“1”，数据产生时也是如此；检查之后最终把它们设置为无效（“0”）。
AO_AF_UNMODIFIED	0×04	值未改，因此是原始数据。一旦值被应用程序更改该标志则必须设为“0”。
AO_AF_DEFINED	0×08	值被定义（“1”），因此可获取。如果值矩阵中的值无法获取，则位被设为“0”。

位的相同含义对属性“global_flag”有效。

“flags”和“global_flag”这两个属性必须遵循一下规则。它们包括处理现有数据的具体规则，如 ASAM ODS 以前的版本允许同时指定“flag”和“global_flag”。

应遵守以下规则：

——在隐式本地列的实例中（这个本地列的从基本属性“sequence_representation”中派生出来的应用属性是“implicit_constant”，“implicit_linear”或“implicit_saw”）“flag”可能不存在，因此必须提供一个“global_flag”的值；

——在非隐式本地列的实例中，“global_flag”和“flag”二者其一必须存在，但两者不能同时存在（如果实例提供“flag”则必须去掉“global_flag”这个属性，因此就这个意义而言，“global_flag”是可选的。）；

——如果现有的数据（无论什么原因）两个属性都包含值，值“flag”决定值的处理方式；

——如果在现有数据或执行中所有的属性都不可获取，这种情况可以当作“global_flag”被设为 15 一样来理解。

本地列的值和单位有以下隐式关系：

- 1) 如果相关的测量量有单位（即它有一个从基本关系“unit”中派生出来的关系，这个基本关系“unit”引用基本类型 AoUnit 的一个实例），这个单位被用来解释值，否则；
- 2) 如果相关的测量数量和数量有关系（即它的从基本关系“数量”中派生出来的关系引用基本类型 AoQuantity 的一个实例），这个数量有一个默认的关系（即它有个一从引用基本类型 AoUnit 的基本关系“default_unit”中派生出来的关系），该单位被用来解释值，否则；
- 3) 该值必须为不带任何相关单位信息的纯值。

请注意值通常是通过 APIs 的具体方法存取，APIs 提供指定与服务器中储存的单位不一样的单位的方法。在这种情况下，API 方法将以用户要求的单位传送值。在相应的 API 方法描述中可以找到更多的信息。

7.4.5 AoExternalComponent (BID=40)

AoExternalComponent 的基本属性和关系见表 85。

表85 AoExternalComponent 的基本属性和关系

基本属性	意义
id	DT_LONGLONG 必选属性(应用模型中需要),应用元素实例唯一的 id,必须是正数(如 1,2....)
name	DT_STRING 必选属性 (应用模型中要求) 实例名称,它在层次列表中其前身范围内是唯一的。
description	DT_STRING 可选 对实例描述性的文字
version	DT_STRING 可选 实例的版本
version_date	DT_DATE 可选 版本变更日期
mime_type	DT_STRING 可选 实例的文件类型
external_references	DT_EXTERNALREFERENCE LIST[0: ?] 引用外部信息
objecttype	DT_LONGLONG 可选 包含实例所属的应用元素(子类或超类)的 id
ordinal_number	DT_LONG 可选 序数,指定这个外部组件的段数,在其父件内部唯一。第一部分将有值 1。一个本地列的外部组件的序数中不存在空隙。如果这个基本属性没被包含在应用模型中,每个将其值储存在外部组件中的本地列每列可能只使用一个外部组件。如果外部组件的实例没有指定序数的值,那么将被假定为缺省值 1。
component_length	DT_LONG 必选属性应用模型中要求) 该组件的长度;对于所有数字数据类型,它显示为值的数目,对于其它数据类型(DT_STRING, DT_BYTESTR, DT_BLOB),它显示为字节数。

filename_url	DT_STRING 必选属性（应用模型中要求） 包含相应本地列的值的文件名的地址；绝对和相对文件名的详细信息，请参阅第 5.10.6 节
value_type	类型规格枚举 必选属性（应用模型中要求） 枚举细节见下面的描述。
start_offset	DT_LONGLONG 可选 必选属性（应用模型中要求） 起始位置的偏移；在值类型=“DT_BLOB”的情况下未指定，在所有其它情况下必须指定。
block_size	DT_LONG 可选 必选属性（应用模型中要求） 块大小；在值类型=“DT_BLOB”的情况下未指定，在所有其它情况下必须指定。
valuesperblock	DT_LONG 可选 必选属性（应用模型中要求） 每块的值；在值类型=“DT_BLOB”的情况下未指定，在所有其它情况下必须指定。
value_offset	DT_LONG 可选 必选属性（应用模型中要求） 值偏移；在值类型=“DT_BLOB”的情况下未指定，在所有其它情况下必须指定。
flags_filename_url	DT_STRING 可选 包含相应本地列的标志的文件名的地址；见 5.10.6 节中关于绝对和相对文件名的详细资料。
flags_start_offset	DT_LONG 可选 标志开始的地方偏移
关系	
local_column	父类 AoLocalColumn INVERSE FOR external_component 必选属性（应用模型中要求）

在应用模型中，从此基础元素中只能派生出一个应用元素（单派生）

“类型说明枚举”的含义是下列几项的枚举：

- dt_boolean (0) ,
- dt_byte (1) ,
- dt_short (2) ,
- dt_long (3) ,
- dt_longlong (4) ,
- ieeefloat4 (5) ,
- ieeefloat8 (6) ,
- dt_short_beo (7) ,
- dt_long_beo (8) ,
- dt_longlong_beo (9) ,
- ieeefloat4_beo (10) ,
- ieeefloat8_beo (11) ,
- dt_string (12) ,
- dt_bytestr (13) ,
- dt_blob (14) ,
- dt_boolean_flags_beo (15) ,
- dt_byte_flags_beo (16) ,
- dt_string_flags_beo (17) ,
- dt_bytestr_beo (18) ,
- dt_sbyte (19) ,
- dt_sbyte_flags_beo (20) ,
- dt_ushort (21) ,
- dt_ushort_beo (22) ,
- dt_ulong (23) ,
- dt_ulong_beo (24)。

括号内的数字标识枚举项的项目编号。每项的文字说明是在用户界面中使用，而项目编号则是在编程的接口上使用。

此枚举中的项目描述外部组件文件夹中值表达的数据类型。关于值在组件文件夹中的存储方式的详细描述可以在第 6.4 节里描述混合模式服务器的内容中找到。

请注意，这些枚举项用来描述外部二进制文件中包含的数据的数据类型。需要把它们和 ODS APIs 使用的数据类型区别开。表 86 提供了 API 数据类型和文件数据类型之间的对应。

表86 内部数据类型和文件数据类型的对应关系

内部数据类型的枚举项（来自数据类型枚举）	文件数据类型的枚举项（来自数据说明枚举）
DT_STRING	dt_string, dt_string_flags_beo
DT_STRING	dt_string, dt_string_flags_beo
DT_SHORT	dt_short, dt_short_beo, (dt_sbyte, dt_sbyte_flags_beo)
DT_FLOAT	ieeefloat4, ieeefloat4_beo
DT_BOOLEAN	dt_boolean, dt_boolean_flags_beo
DT_BYTE	dt_byte, dt_byte_flags_beo

DT_LONG	dt_long, dt_long_beo, (dt_ushort, dt_ushort_beo)
DT_DOUBLE	ieeefloat8, ieeefloat8_beo
DT_LONGLONG	dt_longlong, dt_longlong_beo, (dt_ulong, dt_ulong_beo)
DT_ID	--notusedanymore--
DT_DATE	dt_string, dt_string_flags_beo
DT_BYTESTR	dt_bytestr, dt_bytestr_beo
DT_BLOB	dt_blob
DT_COMPLEX	ieeefloat4, ieeefloat4_beo
DT_DCOMPLEX	ieeefloat8, ieeefloat8_beo
DT_EXTERNALREFERENCE	dt_string, dt_string_flags_beo
DT_ENUM	--noDT_ENUMinLocalColumnvalues--
注：表注如下： 1）依据创建组件文件的电脑的构架使用 Big Enian Order（BEO）或者 Little Endian Order。如果客户端读取组件文件时，自己电脑的构架和文件的端模式顺序不匹配，必须交换数据后再做进一步处理； 2）如果客户端（使用 API 方式）往 ODS 服务器中输入值，必须把这些值转换成适当的内部数据类型，因为 API 方式只识别来自数据类型枚举中的数据类型； 3）如果客户端以 API 方式读取值，则这些值以内部数据类型出现； 4）如果混合模式运行下的服务器从外部组件文件中读取值，它将使用文件数据类型并在将它们转换成内部数据类型之后再做进一步处理。对有些文件数据类型来说，转换要求使用下一个更大的内部符号的数据类型，如表中所示； 5）混合模型 ODS 服务器将使用文件数据类型把值写入外部组件文件中；然而它并不在上表中括号内出现的数据类型。	

7.5 执行基础元素描述

为了使测量能进行几个层次的分测量，以下元素用于构建一个树形结构管理：

- AoTest（AoTestAbstract 子类型）；
- AoSubtest（AoTestAbstract 子类型）。

7.5.1 AoTest（BID=36）

AoTest 的基本属性和关系见表 87。

表87 AoTest 的基本属性和关系

基本属性	意义
id	DT_LONGLONG 必选属性（应用模型中要求） 应用模型实例唯一的 id，必须是正数（1, 2, ...）
name	DT_STRING 必选属性（应用模型中要求） 实例名称

基本属性	意义
description	DT_STRING 可选 对实例进行描述性的文本
version	DT_STRING 可选 实例版本
version_date	DT_DATE 可选 版本变更日期
mime_type	DT_STRING 可选 实例的文件类型
external_references	DT_EXTERNALREFERENCE LIST[0: ?] 引用外部信息
objecttype	DT_LONGLONG 可选 包含实例所属的应用程序（自己或超类）的 id
关系	
children	子类 SET[0: ?] OF AoSubtest 或者 SET[0: ?] OF AoMeasurement 必选属性（应用模型中要求） 这些关系引用从基础元素 AoSubtest 中派生出来的应用元素和从基础元素 AoMeasurement 中派生出来的基础元素中之一，但不能两者结合。
environment	父类 SET[0: ?] OF AoEnvironment INVERSE FOR TESTS

在应用模型中，从此基础元素中只能派生出一个应用元素（单派生）。

7.5.2 AoSubTest (BID=2)

AoSubTest的基本属性和关系见表88。

表88 AoSubTest 的基本属性和关系

基本属性	意义
id	DT_LONGLONG 必选属性（应用模型中要求） 应用元素实例唯一的 id，必须是正数（1, 2, ...）
name	DT_STRING 必选属性（应用模型中要求） 实例的名称，它在层次列表中其前身范围内是唯一的。

description	DT_STRING 可选 对实例描述性的文字
version	DT_STRING 可选 实例的版本
version_date	DT_DATE 可选 版本变更日期
mime_type	DT_STRING 可选 实例的文件类型
external_references	DT_EXTERNALREFERENCE LIST[0: ?] 引用外部信息
objecttype	DT_LONGLONG 可选 包含实例所属的应用元素（子类或超类）的 id
关系	
parent_test	父类 AoTest 或者 AoSubTest 必选属性应用模型中要求
children	子类 SET[0: ?] OF AoSubTest 或者 SET[0: ?] OF AoMeasurement 必选属性（应用模型中要求） 这些关系引用从基础元素 AoSubTest 中派生出来的应用元素和从基础元素 AoMeasurement 中派生出来的基础元素中之一，但不能两者结合。

7.6 描述性数据基础元素的描述

以下元素用来构建测试的补充描述结构：

- a) 被测对象
 - 1) AoUnitUnderTest (AoUnitUnderTest 的子类型)；
 - 2) AoUnitUnderTest (AoUnitUnderTest 子类型)。
- b) 测试序列
 - 1) AoTestSequence (AoTestSequenceAbstract 的子类型)；
 - 2) AoTestSequencePart (AoTestSequenceAbstract 的子类型)。
- c) 测试设备
 - 1) AoTestEquipment (AoTestEquipmentAbstract 的子类型)；
 - 2) AoTestEquipmentPart (AoTesEquipmentAbstract 的子类型)。
- d) AoTestDevice (AoTestEquipmentPart 的子类型)

7.6.1 AoUnitUnderTest (BID=21)

AoUnitUnderTest 的基本属性和关系见表 89。

表89 AoUnitUnderTest 的基本属性和关系

基本属性	意义
id	DT_LONGLONG 必选属性应用模型中要求) 应用元素实例唯一的 id, 必须是正数 (1,2, ..)
name	DT_STRING 必选属性 (应用模型中要求) 实例名称
description	DT_STRING 可选 对实例描述性的文字
version	DT_STRING 可选 实例的版本
version_date	DT_DATE 可选 版本变更日期
mime_type	DT_STRING 可选 实例的文件类型
external_references	DT_EXTERNALREFERENCE LIST[0: ?] 引用外部信息
objecttype	DT_LONGLONG 可选 包含实例所属的应用元素 (子类或超类) 的 id
关系	
children	子类 SET[0: ?] OF AoUnitUnderTest
environment	父类 SET[0: ?] OF AoEnvironment INVERSE FOR 待测物
measurement	INFO_REL SET[0: ?] OF AoMeasurement INVERSE FOR 被测部件

7.6.2 AoUnitUnderTest (BID=22)

AoUnitUnderTest 的基本属性和关系见表 90。

表90 AoUnitUnderTest 的基本属性和关系

基本属性	意义
id	DT_LONGLONG 必选属性（应用模型中要求） 应用模型实例唯一的 id，必须是正数（1,2, ..）
name	DT_STRING 必选属性（应用模型中要求） 实例名称，它在层次列表中其前身范围内是唯一的。
description	DT_STRING 可选 对实例描述性的文字
version	DT_STRING 可选 实例的版本
version_date	DT_DATE 可选 版本变更日期
mime_type	DT_STRING 可选 实例的文件类型
external_references	DT_EXTERNALREFERENCE LIST[0: ?] 引用外部信息
objecttype	DT_LONGLONG 可选 包含实例所属的应用元素（子类或超类）的 id
关系	
children	子类 SET[0: ?] OF AoUnitUnderTest
measurement	INFO_REL SET[0: ?] OF AoMeasurement INVERSE FOR 被测部件
parent_unit_under_test	父类 AoUnitUnderTest INVERSE FOR 子类 若被测父类单位部分存在，它则可能不存在
parent_unit_under_test_part	父类 AoUnitUnderTest INVERSE FOR 子类 若被测父类单位存在，它则可能不存在

7.6.3 AoTestSequence (BID=25)

AoTestSequence的基本属性和关系见表91。

表91 AoTestSequence 的基本属性和关系

基本属性	意义
id	DT_LONGLONG 必选属性（应用模型中要求） 应用模型实例唯一的 id，必须是正数（1,2, ..）
name	DT_STRING 必选属性（应用模型中要求） 实例名称，它在层次列表中其前身范围内是唯一的。
description	DT_STRING 可选 对实例描述性的文字
version	DT_STRING 可选 实例的版本
version_date	DT_DATE 可选 版本变更日期
mime_type	DT_STRING 可选 实例的文件类型
external_references	DT_EXTERNALREFERENCE LIST[0: ?] 引用外部信息
objecttype	DT_LONGLONG 可选 包含实例所属的应用元素（子类或超类）的 id
关系	
children	子类 SET[0: ?] OF AoTestSequencePart
environment	INFO_REL SET[0: ?] OF AoTest INVERSE FOR 顺序
measurement	父类 AoTestSequence INVERSE FOR 子类 若父类顺序部分存在，它则可能不存在

7.6.4 AoTestSequencePart (BID=26)

AoTestSequencePart的基本属性和关系见表92。

表92 AoTestSequencePart 的基本属性和关系

基本属性	意义
id	DT_LONGLONG 必选属性（应用模型中要求） 应用模型实例唯一的 id，必须是正数（1, 2, ..）
name	DT_STRING 必选属性（应用模型中要求） 实例的名称，它在层次列表中其前身范围内唯一。
description	DT_STRING 可选 对实例描述性的文本
version	DT_STRING 可选 实例的版本
version_date	DT_DATE 可选 版本变更日期
mime_type	DT_STRING 可选 实例的文件类型
external_references	DT_EXTERNALREFERENCE LIST[0: ?] 引用外部信息
objecttype	DT_LONGLONG 可选 包含实例所属的应用元素（子类或超类）的 id
关系	
children	子类 SET[0: ?] OF AoTestSequencePart
measurement	INFO_REL SET[0: ?] OF AoMeasurementt 与 sequences 相反
parent_sequence	父类 AoTestSequence 与 children 相反 若 parent_sequence_part 存在，它则可能不存在。
parent_sequence_part	父类 AoTestSequencePart 与 children 相反 若 parent_sequence 存在，它则可能不存在

7.6.5 AoTestEquipment (BID=23)

AoTestEquipment的基本属性和关系见表93。

表93 AoTestEquipment 的基本属性和关系

基本属性	意义
id	DT_LONGLONG 必选属性（应用模型中要求） 应用模型实例唯一的 id，必须是正数（1,2, ..）
name	DT_STRING 必选属性（应用模型中要求） 实例名称
description	DT_STRING 可选 对实例进行描述性的文本
version	DT_STRING 可选 实例版本
version_date	DT_DATE 可选 版本变更日期
mime_type	DT_STRING 可选 实例的文件类型
external_references	DT_EXTERNALREFERENCE LIST[0: ?] 引用外部信息
objecttype	DT_LONGLONG 可选 包含实例所属的应用程序（自己或超类）的 id
关系	
children	子类 SET[0: ?] OF AoTestEquipmentPart 和/或 AoTestDevice 该集可能包含两个基本类型之一的实例或两个类型的实例都包含。
measurement_quantities	INFO_FROM SET[0: ?] OF AoMeasurementQuantity INVERSE FOR 通道
environmen	父类 SET[0: ?] OF AoEnvironment INVERSE FOR 设备
measurement	INFO_REL SET[0: ?] OF AoMeasurement INVERSE FOR 设备

7.6.6 AoTestEquipmentPart (BID=24)

AoTestEquipmentPart的基本属性和关系见表94。

表94 AoTestEquipmentPart 的基本属性和关系

基本属性	意义
id	DT_LONGLONG 必选属性（应用模型中要求） 应用模型实例唯一的 id，必须是正数（1,2, ..）
name	DT_STRING 必选属性（应用模型中要求） 实例名称
description	DT_STRING 可选 对实例进行描述性的文本
version	DT_STRING 可选 实例版本
version_date	DT_DATE 可选 版本变更日期
mime_type	DT_STRING 可选 实例的文件类型
external_references	DT_EXTERNALREFERENCE LIST[0: ?] 引用外部信息
objecttype	DT_LONGLONG 可选 包含实例所属的应用程序（自己或超类）的 id
关系	
children	子类 SET[0: ?] OF AoTestEquipmentPart 和/或 AoTestDevice 该集可能包含两个基本类型之一的实例或两个类型的实例都包含。
measurement_quantities	INFO_FROM SET[0: ?] OF AoMeasurementQuantity INVERSE for channel
measurement	INFO_REL SET[0: ?] OF AoMeasurement INVERSE FOR equipments
parent_equipment	父类 AoTestEquipment INVESE FOR 子类 若父类设备部分存在，它则可能不存在

parent_equipment_part	父类 AoTestEquipmentPart INVERSE FOR 子类（若父类设备存在，它则可能不存在）

7.6.7 AoTestDevice (BID=37)

AoTestDevice 的基本属性和关系将表 95。

表95 AoTestDevice 的基本属性和关系

基本属性	意义
id	DT_LONGLONG 必选属性（应用模型中要求） 应用模型实例唯一的 id，必须是正数（1,2, ..）
name	DT_STRING 必选属性（应用模型中要求） 实例名称，它在层次列表中其前身范围内是唯一的。
description	DT_STRING 可选 对实例描述性的文字
version	DT_STRING 可选 实例的版本
version_date	DT_DATE 可选 版本变更日期
mime_type	DT_STRING 可选 实例的文件类型
external_references	DT_EXTERNALREFERENCE LIST[0: ?] 引用外部信息
objecttype	DT_LONGLONG 可选 包含实例所属的应用元素（子类或超类）的 id
关系	
children	子类 SET[0: ?] OF AoTestDevice
measurement	INFO_REL SET[0: ?] OF AoMeasurement INVERSE FOR equipments
measurement_quantities	INFO_FROM SET[0: ?] OF AoMeasurementQuantity

	INVERSE FOR channel
parent_equipment	父类 AoTestEquipment INVERSE FOR children 若父类设备部分存在，它则可能不存在
parent_equipment_part	父类 AoTestEquipmentPart 或 AoMeasurementDevice INVERSE FOR children 若父类设备存在，它则可能不存在

7.7 安全性的基础元素描述

为安全数据存储提供以下元素：

- AoUser;
- AoUserGroup。

7.7.1 AoUser (BID=34)

AoUser 的基本属性和关系见表 96。

表96 AoUser 的基本属性和关系

基本属性	意义
id	DT_LONGLONG 必选属性（应用模型中要求） 应用模型实例唯一的 id，必须是正数（1,2, ..）
name	DT_STRING 必选属性（应用模型中要求） 实例名称
description	DT_STRING 可选 对实例进行描述性的文本
version	DT_STRING 可选 实例版本
version_date	DT_DATE 可选 版本变更日期
mime_type	DT_STRING 可选 实例的文件类型
external_references	DT_EXTERNALREFERENCE LIST[0: ?] 引用外部信息

objecttype	DT_LONGLONG 可选 包含实例所属的应用程序（自己或超类）的 id
password	DT_STRING 必选属性（应用模型中要求） 包含指令
关系	
groups	INFO_REL SET[0: ?] OFAoUserGroup INVERSE FOR 用户 必选属性（应用模型中要求）

7.7.2 AoUserGroup (BID=35)

AoUserGroup基本属性和关系见表97。

表97 AoUserGroup 基本属性和关系

基本属性	意义
id	DT_LONGLONG 必选属性（应用模型中要求） 应用模型实例唯一的 id，必须是正数（1,2, ..）
name	DT_STRING 必选属性（应用模型中要求） 实例名称
description	DT_STRING 可选 对实例进行描述性的文本
version	DT_STRING 可选 实例版本
version_date	DT_DATE 可选 版本变更日期
mime_type	DT_STRING 可选 实例的文件类型
external_references	DT_EXTERNALREFERENCE LIST[0: ?] 引用外部信息
objecttype	DT_LONGLONG 可选 包含实例所属的应用程序（自己或超类）的 id
superuser_flag	DT_SHORT

	必选属性（应用模型中要求） Flag，指示是否是 Superuser
关系	
users	INFO_REL SET[0: ?] OF AoUser 必选属性（应用模型中要求）

7.8 其它数据基础元素的描述

为存储其它数据提供以下元素：

- AoAny;
- AoLog;
- AoParameter;
- AoParameterSe。t

7.8.1 AoAny

AoAny的基本属性和关系见表98。

表98 AoAny 的基本属性和关系

基本属性	意义
id	DT_LONGLONG 必选属性（应用模型中要求） 应用模型实例唯一的 id，必须是正数（1,2, ..）
name	DT_STRING 必选属性（应用模型中要求） 实例名称
description	DT_STRING 可选 对实例描述性的文字
version	DT_STRING 可选 实例的版本
version_date	DT_DATE 可选 版本变更日期
mime_type	DT_STRING 可选 实例的文件类型
external_references	DT_EXTERNALREFERENCE LIST[0: ?] 引用外部信息

objecttype	DT_LONGLONG 可选 包含实例所属的应用元素（子类或超类）的 id
关系	
parent	父类 SET[0: ?] OF AoAny INVERSE FOR 子类
children	子类 SET[0: ?] OF AoAny

若父子关系不存在，那么 ASAM ODS 环境中名称和版本的结合必须唯一。

7.8.2 AoLog

AoLog的基本属性和关系见表99。

表99 AoLog 的基本属性和关系

基本属性	意义
id	DT_LONGLONG 必选属性（应用模型中要求） 应用模型实例唯一的 id，必须是正数（1,2, ..）
name	DT_STRING 必选属性（应用模型中要求） 实例名称
description	DT_STRING 可选 对实例描述性的文字
version	DT_STRING 可选 实例的版本
version_date	DT_DATE 可选 版本变更日期
mime_type	DT_STRING 可选 实例的文件类型
external_references	DT_EXTERNALREFERENCE LIST[0: ?] 引用外部信息
objecttype	DT_LONGLONG 可选 包含实例所属的应用元素（子类或超类）的 id
date	DT_DATE

	输入日志的日期
关系	
parent	父类 SET[0: ?]OF AoLog INVERSE FOR children
children	子类 SET[0: ?] OF AoLog

7.8.3 AoParameter (BID-44)

AoParameter 的基本属性和关系见表 100。

表100 AoParameter 的基本属性和关系

基本属性	意义
id	DT_LONGLONG 必选属性（应用模型中要求） 应用元素实例唯一的 id，必须是正数（1,2, ..）
name	DT_STRING 必选属性 （应用模型中要求） 实例名称，它在层次列表中其前身范围内是唯一的。
description	ST_STRING 可选 对实例描述性的文字
version	DT_STRING 可选 实例的版本
version_date	DT_DATE 可选 版本变更日期
mime_type	DT_STRING 可选 实例的文件类型
external_references	DT_EXTERNALREFERENCE LIST[0: ?] 引用外部信息
objecttype	DT_LONGLONG 可选 包含实例所属的应用元素（子类或超类）的 id
parameter_datatype	datatype_enum 必选属性（应用模型中要求）
Pvalue	DT_STRING 必选属性（应用模型中要求） 参数值总是以字符串表示，非字符做相应的转换。

关系	
unit	INFO_TO AoUnit 可选 必选属性（应用模型中要求）
parameter_set	父类 AoParameterSet INVERSE FOR 参数 必选属性（应用模型中要求）

参数数据类型可以包含在 5.5.2 中描述的引用除了 DT_UNKNOWN 和 DT_ID 序列类型之外的任何基础模型数据类型为枚举的枚举元素。序列类型（DS_……）则不允许。

7.8.4 AoParameterSet (BID=45)

AoParameterSet 的基本属性和关系见表 101。

表101 AoParameterSet 的基本属性和关系

基本属性	意义
id	DT_LONGLONG 必选属性（应用模型中要求） 应用元素实例唯一的 id，必须是正数（1,2, ..）
name	DT_STRING 必选属性（应用模型中要求） 实例名称
description	ST_STRING 可选 对实例描述性的文字
version	DT_STRING 可选 实例的版本
version_date	DT_DATE 可选 版本变更日期
mime_type	DT_STRING 可选 实例的文件类型
external_references	DT_EXTERNALREFERENCE LIST[0: ?] 引用外部信息
objecttype	DT_LONGLONG 可选

	包含实例所属的应用元素（子类或超类）的 id
关系	
parameters	子类 SET[0: ?] OF AoParameter 必选属性应用模型中要求)

7.9 ASAM ODS 基础模型形式描述

使用 ISO 10303 的方法给 ASAM ODS 基础模型下定义。各个定义之间的区别作了解释，并且添加了图形帮助加强理解。

7.9.1 为什么采用标准化的表述

为大量不同但是协调的应用模型定义一个基础模型要求一个对精度和处理要求严格的描述。为了避免理解上的错误，描述的语言必须严格定义。使用全球标准化的语言可以保证标准获得较长的生命周期和模型描述的高可靠性，而且可以促进全世界可用的工具来处理语言，推动派生和执行的自动化。

这种国际化的描述在 ISO 10303 中可用。（“交换产品模型数据的标准”，缩写为：STEP）。这个标准包括一种数据描述语言，称作“EXPRESS”，以及：

- 一个图形表示（EXPRESS-G）；
- 物理数据存储和存取的方案；
- 种类繁多的数据模型（基础模型称为“资源”，“派生模型称为”应用协议）。

有大量的工具可用，包括 EXPRESS 编译器，元数据和数据的解析器以及许多的应用，例如在计算机辅助设计、工程和流程工业中。

由于要做到描述精确及遵从国际标准，所以使用 ISO 10303（EXPRESS）来描述 ASAM ODS 基础模型。

7.9.2 用 ISO 10303 描述元模型

本部分将介绍 EXPRESS 语言及定义和表示实体的方式。

EXPRESS 是一种 ISO 10303 系列标准的第 11 部分标准化的语言，用于数据建模。在 EXPRESS 中，用模型架构的形式描述数据的外观，表现形式和处理。在 EXPRESS 中，一个模型包括实体（即数据类型）和它们的名称，属性，规则，功能及定义。下面有 EXPRESS 的两种表现方式：

- 在 ASC II 中是一个完整版本（请见下一节的第一部分）；
- 在图示格式下的一个简化的版本，因此被称为 EXPRESS-G。

在 EXPRESS 中可以结合一个实例来理解一个实体的定义：

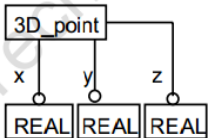
示例：这个例子显示了一个简单的 3D 点的定义。可以看出，3D 点有三个属性（分别是“x”，“y”，“z”），每个属性都是 REAL 类型。

ENTITY 3D_point;

x:REAL;

y:REAL;

z:REAL;



END_ENTITY

用 EXPRESS 写数据模型需要给每个属性定义一个名称和类型。这样的属性也有可能是一个个的关系，在这种情况下，属性的类型可能是关系指向的实体的名称。不同于关系数据库管理系统，在数据建模中，在指向原始数据类型（如整型，布尔型，字符串型，...）和指向其它实体的属性之间没有原则上的差异。

示例：一个点可能是一行的一部分。这样的话，它和一个实体有关，表现为：

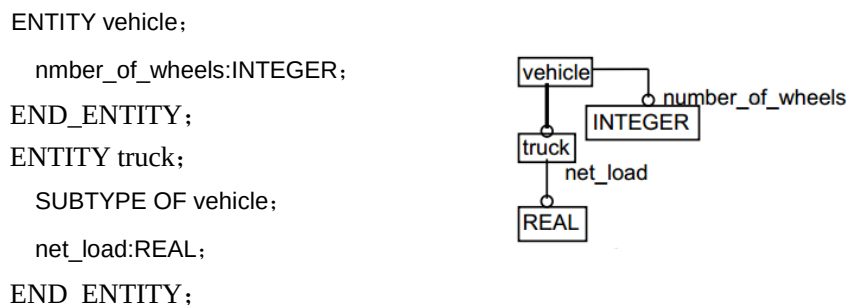


如果在两个方向上都需要同个关系，可以显式的制定属性的逆关系。相反属性的定义前有关键字“INVERSE”。

EXPRESS 还允许描述实体的推导，即在现有实体的基础上创建其子类型，这些子类型继承了父类的所有属性而且可能还有它们自己的属性。

在 EXPRESS-G 中，实体属性关系表现为细线，而父类-子类关系表现为粗线。

示例：这个例子显示了属性和继承关系之间的区别。卡车有两个属性：number_of_wheels（卡车的轮子个数）（继承其父类汽车的属性）和 net_load（净负荷）。



ASAM 用户都知道从基础元素中派生出应用元素的过程。这是一个典型的父类-子类之间的关系，其和 EXPRESS 语言之间的映射如下：

- 在 EXPRESS 语言中，每个 ASAM ODS 用一个实体代表；
- 在 EXPRESS 语言中，每个 ASAM ODS 应用元素也由一个实体代表；
- 每个代表应用元素的实体在 EXPRESS 语言中被指定为代表相应基础元素的实体的子类型。

属性的使用经常受到相关容许值（数值范围）、类型或项数的限制。在 EXPRESS 中，这些限制被制定成了规则。规则之前有关键字“WHERE”。

有时候，在规则内进行计算、检查或评估是有必要的。这通常是用不同的功能完成，经常位于 EXPRESS 文件的一个独立的章节中

用 EXPRESS 语言编写的模型称为“模式”。有几个工具可以测试和检查该模式，并且可以相应的配置数据存储库，这些工具被称为“EXPRESS 编译器”。

用 EXPRESS 语言编写数据模型和处理这些模式是 ISO 10303 即“STEP 方法学”的一部分。

7.9.3 用 EXPRESS 表达的基础模型

SCHEMA asam30;

(*

ASAM-ODS Base Model for ODS 5.2.0

23 Jun 2008

30.06.2009:- AoExternalComponent start_offset & flags_start_offset
attribute changed type from T_LONG to T_LONGLONG to address
binary files greater 2GB

- Removed SUPERTYPE definition of deletetd (d)
- complex_inst_attribute
- Children on AoQuantity.successors should be unique in
name/version
- Children on AoParametersSets.Parameters should be unique
in name/version
- Children on AoLog.children should be unique in name/version

18.11.2008: Added dt_sbyte, dt_sbyte_flags_bco, dt_ushort,
dt_ushort_bco, dt_ulong, dt_ulong_bco to the typespec_enum
to allow addressing of component files that were binary
written by loggers and want to address their raw data. Make
sure that T_BYTE is an unsigned type that already existed.

23.06.2008: Removed flag attribute of attribute_object.
Because the flag storage of the server should be removed
(never used) the flag attribute of the attribute_object is
removed too. This should not cause any problem because neither
file nor server have a storage for those flags and they do
not affect existing implementations.

28.9.2006: clean up all SET/LIST[0:?]

5.12.2005: Model 29 for ODS 5.1

- remove ModelMapper & Query
- minor update on xxx_BEO & MANDATORY

12.10.2005: preliminary model 29 (ODS 5.1)

- xxx_BEO in typespec_enum
- no formula representation in local columns
- update on MANDATORY attributes especially for inverses
- improved rules (on flags, IDs, ext.comp., MeaQuant,
Quantity)

06.5.2005: updates WG27apr05:

- no quantity_classification
- number of independent columns
- MANDATORY in model / OPTIONAL in instantiation
- timzone in AoEnvironment, component_length in
AoExternalComponent
- rules for measurement_quantities and ordinal_number
- inverse relation from AoUnit to AoMeasurementQuantity
- loosened uniqueness rule for meaqs in measurements

20.4.2005: inverse relation "measurement_quantity"
in AoTestEquipmentAbstract, MANDATORY ATTRIBUTES marked
preliminarily

22.6.2004: AoModelMapper+AoQuery added

25.3.2004: Model 28 used in ASAM ODS 5.0

- new attribute in AoLocalColumn: raw_datatype

- 24.11.2003: Little adjustments in comments
- writing of DT_EXTERNALREFERENCE and dt_bytestr
 - upper case for T_XXX
 - STRING as attribute type replaced by T_STRING
- Visualization: ASAM27.DOC (model pictures in Express-G)
-
- 20.3.2003: Model 27 for ASAM ODS 4.2 (due in June 2003)
- sequence_representation extended by "calibrated"
- 13.12.2002: Model 27
- adjust where rules and uniqueness rules
 - remove blob sequence, add boolean and longlong sequence
- 17.10.2002: Model 27 for ASAM ODS 4.2 (preliminary)
- extend seq_rep_enum
 - new elements AoNameMap/AoAttributeMap andAoParameter/AoParameterSet
 - AoEnvironment extended by Application_model_type and rules
 - some attributes in AoMeasurement and AoMeasurementQuantitybecome optional
- (9.5.2002) Model 26 for ASAM ODS 4.2 (study)
- AoLog added
 - Modifications in AoExternal component
 - * new attributes ordinal_number and length_in_bytes
 - * attribute "blocksize" changed to "block_size"
 - * value_type added / value_offset changed in AoExternalComponentplus addition of typespec_enum
 - AoAny extended with father/child relation
 - Data type DT_ENUM added for enumerations
 - still to do: rephrase unique rules
- (19.9.2001) Extended AoEnvironment + modified Instance Attrs
- (03.9.2001) Model 25 with improved relationship types and moreinverse attributes
- (12.7.2001) Model 25 for Discussion
- (10.7.2001) For ODS-Version 4.1
- ModelMapper commented out
 - independent (flag in AoLocalColumn) is t_short
 - superuser_flag (in AoUserGroup) is t_short
 - AoEnvironment corrected and old Environment integrated
 - instance attributes of all types with units
 - integration of attributes of all types with units and flags
 - relation types given for reference attributes
- (30.5.2001) Corrections for inheritance support:
- Attribute "objecttype" of asam_base_entity with "optional"declarator.
 - Rules for Test and Measurement were adjusted to avoidcontradictions.
-
- (20.02.2001) To be used for ODS-API 4.1
- Derived from models 22 (3.2) and 23 (4.0)
 - Proposed entities added: AoEnvironment, AoModelMapper,AoExternalComponent
- (14.09.00) Base Model version 23 valid for ODS-API 4.0 (derived from version 21) =====
- (7.9.2000) Derived version 22 (from 21) only valid for ODS-API 3.2
- (13.07.00) Changes in version 21 (from version 20) :
- To be used for Version 3.2 and prepared for Version 4.0
 - Several attributes marked as "optional", some enumerations

- cut,
- Several entities marked as "NOT FOR VERSION 3.2" are not to be used by API before Version 4.0
- no 64-bit integer, no complex, no boolean numbers
- AoEnvironment inserted provisionally for compatibility reasons and for eventual future use.
- (15.06.00) Changes in version 20 (from version 19) :
 - Security added again
 - Simple data types more precise
- (20.01.00) Changes in version 19 (from version 17) :
 - Adding MimeTypes and ExternalReference
 - No security entities
- (17.05.99) Changes in version 18 (from version 17) :
 - Adding security entities
- (03.02.99) Changes in version 17 (from version 16) :
 - Implicit, raw data and formula data added in AoLocalColumn
 - subtypes "implicit_sequence" and "formula_sequence" removed
 - division in numeric and textual_sequences
- (30.10.98) Model 16.

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

*)

(*

The following explanation of OPTIONAL and MANDATORY ATTRIBUTE has been set up and should be understood:

Concerning existence rules there are 2 classifications for attributes:

The 1st classification distinguishes the 2 cases for the model:

- 1.1 a MANDATORY ATTRIBUTE must appear in the application element
- 1.2 not marked as MANDATORY ATTRIBUTE: the attribute can be omitted in the application element

The 2nd classification distinguishes the 2 cases for the data:

- 2.1 an OPTIONAL attribute need not have a value in the instance element
- 2.2 not marked as OPTIONAL: the attribute must have a value in the instance element (NOT NULL in database systems)

*)

```

ENTITY asam_base_entity
ABSTRACT SUPERTYPE OF (ONE OF (
  AoEnvironment,
  (* AoModelMapper, *)
  (* AoQuery, *)
  AoTestAbstract,
  AoMeasurement,
  AoMeasurementQuantity,
  AoSubmatrix,
  AoLocalColumn,
  AoExternalComponent,
  AoUnitUnderTestAbstract,

```

```

AoTestEquipmentAbstract,
AoTestSequenceAbstract,
AoQuantity,
AoUnit,
AoPhysicalDimension,
AoQuantityGroup,
AoUnitGroup,
AoUser,
AoUserGroup,
AoLog,
AoParameter,
AoParameterSet,
AoNameMap,
AoAttributeMap,
AoAny
) );
name          : T_STRING;  (* MANDATORY ATTRIBUTE *)
id            : T_LONGLONG; (* MANDATORY ATTRIBUTE *)
version       : OPTIONAL T_STRING;
description   : OPTIONAL T_STRING;
version_date  : OPTIONAL T_DATE;
mime_type     : OPTIONAL T_STRING;
external_references : LIST [0:?] OF T_EXTERNALREFERENCE;
instance_attributes : SET [0:?] OF instance_attribute;
objecttype    : OPTIONAL T_LONGLONG;
               (*appl_id for inheritance*)

END_ENTITY;

(*
This supertype lets all base entities inherit the attributes that each
of them needs. Being an "abstract" supertype it is not to be instantiated.
name,id: used for identifying any instance of an Asam entity
The mime types are special strings with the following pattern:
    application/x.asam.ods.AOAOAO.XXXXXXX
with AOAOAO = name of an AsamOds base element
and XXXXXX = user defined name of a specialisation
external_reference usually points to files, but may also point to remote
documents (e.g. available via WWW)
The objecttype supports implementations that handle IDs of application
elements for convenience reasons.
*)

(* ----- *)
(* types and auxiliary entities *)
(* ----- *)

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

ENTITY AoNameMap (* BID=46 *)
SUBTYPE OF (asam_base_entity);

```

```

        entity_name : T_STRING;    (* MANDATORY ATTRIBUTE *)
        alias_names : LIST[0:?] OF T_STRING;    (* MANDATORY ATTRIBUTE *)
        attribute_mapping : SET [0:?] OF AoAttributeMap;    (*CHILD*)
                                (* MANDATORY ATTRIBUTE *)

INVERSE
    environment : AoEnvironment FOR entity_mapping;    (*INFO_TO*)
UNIQUE
    UR1: SELF\asam_base_entity.id;
WHERE
    WR1: SIZEOF (SELF.alias_names) =SIZEOF
        (SELF.environment.meaning_of_aliases);
END_ENTITY;

(*
    For the entity name any number of alias names (e.g. for language versions)
    may be specified.
    The list allows different language version switched by the application
    software.
    The length of the list shall be equal to the list "meaning_of_aliases"
    in AoEnvironment (Where Rule 1) .
    For each entity a name mapping may be given;
    For each attribute of the entity an attribute mapping may be given.
*)

ENTITY AoAttributeMap (* BID=47 *)
SUBTYPE OF (asam_base_entity);
    attribute_name : T_STRING;    (* MANDATORY ATTRIBUTE *)
    alias_names : LIST[0:?] OF T_STRING;    (* MANDATORY ATTRIBUTE *)
INVERSE
    name_mapping : AoNameMap FOR attribute_mapping;    (*FATHER*) (*
        MANDATORY ATTRIBUTE *)
UNIQUE
    UR1: SELF\asam_base_entity.id;
WHERE
    WR1: SIZEOF (SELF.alias_names) =SIZEOF
        (SELF.name_mapping.alias_names);
END_ENTITY;

(*
    Specifies any number of alias names for the attribute "attribute_name".
    The length of the list of alias names shall be equal to the length of
    that list in AoNameMap (Where Rule 1) .
*)

ENTITY AoParameter (* BID=44 *)
SUBTYPE OF (asam_base_entity);
    unit : OPTIONAL AoUnit;    (*INFO_TO*)    (* MANDATORY ATTRIBUTE *)
    parameter_datatype : datatype_enum;    (* MANDATORY ATTRIBUTE *)
    pvalue : T_STRING;    (*always string, other types are converted*)
        (* MANDATORY ATTRIBUTE *)
INVERSE
    parameter_set : AoParameterSet for parameters;    (*FATHER*)
        (* MANDATORY ATTRIBUTE *)

```


UNIQUE

UR1: SELF\asam_base_entity.id;
 UR2: SELF\asam_base_entity.name,
 SELF\asam_base_entity.version,
 parameter_set;

WHERE

WR1: SELF.parameter_datatype <> dt_unknown
 ANDSELF.parameter_datatype <> dt_id;

END_ENTITY;

ENTITY AoParameterSet (* BID=45 *)

SUBTYPE OF (asam_base_entity);

parameters : SET [0:?] OF AoParameter; (*CHILD*)
 (* MANDATORY ATTRIBUTE *)

UNIQUE

UR1: SELF\asam_base_entity.id;

WHERE

WR1: unique_name_and_version(parameters);

END_ENTITY;

TYPE instance_attribute = SELECT (

byte_inst_attribute,
 short_inst_attribute,
 long_inst_attribute,
 float_inst_attribute,
 double_inst_attribute,
 (*complex_inst_attribute, complex_instance_attributes taken
 outdcomplex_inst_attribute,*)
 string_inst_attribute,
 time_inst_attribute

);

END_TYPE;

(*

Instance attributes allow giving one or more values with aspecific
 meaning to any instance.
 The meaning shall be expressed in their attribute "name".
 The value (s) shall be put into the "values" attribute ofone of the
 subtypes according to the data type.
 The entities for the instance attributes are defined in thefamily of
 attribute_object.

*)

ENTITY attribute_object

ABSTRACT SUPERTYPE OF (ONEOF (

byte_abs_attribute,
 short_abs_attribute,
 long_abs_attribute,
 float_abs_attribute,
 double_abs_attribute,

```

        complex_abs_attribute,
        dcomplex_abs_attribute,
        string_abs_attribute,
        time_abs_attribute
    ) );
    unit : OPTIONAL AoUnit;
END_ENTITY;

(*
For each datatype we provide three instanciable subtypes of
attribute_object:
xxxx_attribute      with 1.unit,2.svalue
xxxx_seq_attribute  with 1.unit,2.values
xxxx_inst_attribute with 1.unit,2.values,3.name
(xxxx = float/long/string/...)
These attribute objects can be used for
a) instance attributes
b) application attributes -> then their data type is not primitive (e.g.
    plain float or list of float) but it is an object with
    - in case of instance attributes: a name
    - a value (or values)
    - an optional unit
An attribute "quantity" was not provided.
An application attribute may always be
- a simple (primitive) attribute: a value or list of values
- an attribute object like described above.
This must be valid also for those application attributes which are
inherited from base attributes.
Therefore a select type containing all options is created for each
datatype, its name is:
xxxx_attr_select (xxxx = float/long/string/...)
*)

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

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

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

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

```

```

END_ENTITY;

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

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

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

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

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

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

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

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

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

ENTITY float_attribute

```

```

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

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

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

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

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

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

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

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

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

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

```

```

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

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

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

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

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

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

ENTITY complex_attribute
(* SUPERTYPE OF (complex_inst_attribute) *)
SUBTYPE OF (complex_abs_attribute);
    svalue : LIST[2:2] OF T_FLOAT;
END_ENTITY;

ENTITY complex_seq_attribute
SUBTYPE OF (complex_abs_attribute);
    values : LIST[2:?] OF T_FLOAT;
WHERE
    WR1: SIZEOF (values) MOD 2 = 0;
END_ENTITY;

(* taken out
    ENTITY complex_inst_attribute
    SUBTYPE OF (complex_attribute);
        name : T_STRING;
    END_ENTITY;
*)

ENTITY dcomplex_abs_attribute

```

```

ABSTRACT SUPERTYPE OF (ONEOF (
    dcomplex_seq_attribute,
    dcomplex_attribute) )
SUBTYPE OF (attribute_object);
END_ENTITY;

ENTITY dcomplex_attribute
    (* SUPERTYPE OF (dcomplex_inst_attribute) *)
SUBTYPE OF (dcomplex_abs_attribute);
    svalue : LIST[2:2] OF T_DOUBLE;
END_ENTITY;

ENTITY dcomplex_seq_attribute
SUBTYPE OF (dcomplex_abs_attribute);
    values : LIST[2:?] OF T_DOUBLE;
WHERE
    WR1: SIZEOF(values) MOD 2 = 0;
END_ENTITY;

(* taken out
    ENTITY dcomplex_inst_attribute
    SUBTYPE OF (dcomplex_attribute);
    name : T_STRING;
    END_ENTITY;
*)

(* THE types for most attributes: *)
TYPE byte_attr_select = SELECT (
    T_BYTE,
    byte_sequence,
    byte_attribute,
    byte_seq_attribute );
END_TYPE;

TYPE short_attr_select = SELECT (
    T_SHORT,
    short_sequence,
    short_attribute,
    short_seq_attribute );
END_TYPE;

TYPE long_attr_select = SELECT (
    T_LONG,
    long_sequence,
    long_attribute,
    long_seq_attribute );
END_TYPE;

TYPE float_attr_select = SELECT (
    T_FLOAT,
    float_sequence,
    float_attribute,
    float_seq_attribute );
END_TYPE;

TYPE double_attr_select = SELECT (

```

```

        T_DOUBLE,
        double_sequence,
        double_attribute,
        double_seq_attribute );
END_TYPE;

TYPE complex_attr_select = SELECT (
    T_COMPLEX,
    complex_sequence,
    complex_attribute,
    complex_seq_attribute );
END_TYPE;

TYPE dcomplex_attr_select = SELECT (
    T_DCOMPLEX,
    dcomplex_sequence,
    dcomplex_attribute,
    dcomplex_seq_attribute );
END_TYPE;

TYPE time_attr_select = SELECT (
    T_DATE,
    time_sequence,
    time_attribute,
    time_seq_attribute );
END_TYPE;

TYPE string_attr_select = SELECT (
    T_STRING,
    string_sequence,
    string_attribute,
    string_seq_attribute );
END_TYPE;

TYPE T_BOOLEAN = BOOLEAN;
(* 1 bit *)
END_TYPE;

TYPE T_BYTE = INTEGER;
(* 8 bit unsigned numeric value *)
END_TYPE;

TYPE T_SHORT = INTEGER;
(* 16 bit *)
END_TYPE;

TYPE T_LONG = INTEGER;
(* 32 bit *)
END_TYPE;

TYPE T_LONGLONG = INTEGER;
(* 64 bit *)
END_TYPE;

(*
    TYPE T_ID = T_LONGLONG;

```

```

        * 64 bit *
    END_TYPE;
*)

TYPE T_FLOAT = REAL;
(* 32 bit *)
END_TYPE;

TYPE T_DOUBLE = REAL;
(* 64 bit *)
END_TYPE;

TYPE T_COMPLEX = LIST [2:2] OF REAL;
(* 32 bit *)
END_TYPE;

TYPE T_DCOMPLEX = LIST [2:2] OF REAL;
(* 64 bit *)
END_TYPE;

TYPE T_STRING = STRING;
(* UTF8 string terminated by 0 *)
END_TYPE;

TYPE T_DATE = STRING;
(* meaning: YYYYMMDDhhmmsslllccnnn....
year month day hour minute second millisec microsec nanosec ...*)
END_TYPE;

TYPE T_BYTESTR = STRING;
(* first 4 byte = number of following bytes *)
END_TYPE;

TYPE T_BLOB = STRING;
(* 1.text, 2.size in bytes, 3.binary part *)
END_TYPE;

TYPE datatype_enum = ENUMERATION OF (
    DT_UNKNOWN,      (* 0 , also for "reserved" *)
    DT_STRING,       (* 1 *)
    DT_SHORT,        (* 2 *)
    DT_FLOAT,        (* 3 *)
    DT_BOOLEAN,      (* 4 *)
    DT_BYTE,         (* 5 *)
    DT_LONG,         (* 6 *)
    DT_DOUBLE,       (* 7 *)
    DT_LONGLONG,     (* 8 *)
    DT_ID,           (* 9 , not used in attributes *)
    DT_DATE,         (* 10 *)
    DT_BYTESTR,      (* 11 *)
    DT_BLOB,         (* 12 *)
    DT_COMPLEX,      (* 13 *)
    DT_DCOMPLEX,     (* 14 *)
    DT_EXTERNALREFERENCE, (* 28 *)
    DT_ENUM);        (* 30 *)
END_TYPE;

```



```

(* ----- *)
(* Begin of ASAM-ODS base elements *)
(* ----- *)

ENTITY AoEnvironment (* BID=1 *) (* previously: BID=42 *)
SUBTYPE OF (asam_base_entity); (*relation_types: *)
    tests      : SET [0:?] OF AoTest; (*CHILD*)
    uuts       : SET [0:?] OF AoUnitUnderTest; (*CHILD*)
    equipments  : SET [0:?] OF AoTestEquipment; (*CHILD*)
    sequences   : SET [0:?] OF AoTestSequence; (*CHILD*)
    entity_mapping : SET [0:?] OF AoNameMap; (*INFO_FROM*)
    meaning_of_aliases : LIST [0:?] OF T_STRING;
    max_test_level : OPTIONAL T_LONG;
    base_model_version : OPTIONAL T_STRING;
                        (* schema name, in this case 'asam29' *)
    application_model_type : OPTIONAL T_STRING;
                        (* may contain many names, comma-separated *)
    application_model_version : OPTIONAL T_STRING;
                        (* any operator-supplied name *)
    timezone : OPTIONAL T_STRING;
                        (* Examples: "GMT-07:00", "GMT+05:45"
                           Timezones are additional information, they do not
                           change time values. *)

UNIQUE
    UR1: SELF\asam_base_entity.id;
WHERE
    WR1: unique_name_and_version (tests);
    WR2: unique_name_and_version (uuts);
    WR3: unique_name_and_version (equipments);
    WR4: unique_name_and_version (sequences);
    (* There shall not be 2 members of the sets with
       identical name and version. *)
END_ENTITY;

(*
    There should be only one instance of environment.
    The server name should be stored in the attribute "name" which is
    inherited from asam_base_entity.
    Only instances of the top entities of the following 4 hierarchical trees
    are collected in the sets:
        - AoTest
        - AoUnitUnderTest
        - AoTestEquipment
        - AoTestSequence.
    In entity_mapping all the maps are to be collected.
    The set contains one AoNameMap per entity.
    An alias_index in the application software allows to use the appropriate
    alias_name from the given lists.
    In this way language versions can be switched easily.
    For each position in the lists of alias names a string should be provided
    in "meaning_of_aliases"; this allows for name mapping to different
    languages.
    The attribute "max_test_level" shows the number of levels in the test
    hierarchy.
*)

```

(*

```

taken out for ODS 5.1:
ENTITY AoModelMapper      * BID=41 *
SUBTYPE OF (asam_base_entity);
    ruletext : T_STRING;
UNIQUE
    UR1: SELF\asam_base_entity.id;
END_ENTITY;
ENTITY AoQuery             * BID=48 *
SUBTYPE OF (asam_base_entity);
    xml_text : T_STRING;
UNIQUE
    UR1: SELF\asam_base_entity.id;
END_ENTITY;

```

*)

(*

Application elements are built by deriving subtypes from the base entities given in this schema. For the management of measurements and tests the following rules have to be followed:

1. In the whole model only one entity of type AoTest (or a derived subtype) may be instantiated. This one is the maintest (e.g. an application element called "Testseries").
2. Assuming no entities of type AoSubTest (or its subtype) are used, then the attribute "children" of the AoTestAbstractelement "Testseries" is always of the type "Measurement", i.e. children of an instance of "Testseries" are all instances of "Measurement". It is not allowed to have children of type "Testseries".
The Test-Measurement-Hierarchy then has the form
Testseries --o Measurement
3. Assuming there is one application element derived from AoSubTest (e.g. called "Maintest"), the attribute "children" of "Testseries" is of type "Maintest", and the attribute "parent_test" of "Maintest" is of type "Testseries".
The attribute "children" of "Maintest" is of type "Measurement".
The Test-Measurement-Hierarchy then has the form
Testseries --o Maintest --o Measurement
4. Assuming more than one application element have been derived from AoSubTest (e.g. "Maintest", "Subtest" and "Subsubtest"), then they have to be built strictly as a chain:
Testseries --o Maintest --o Subtest --o Subsubtest

|
o

Measurement

In this chain the uppermost application element of AoSubTest (here "Maintest") has a "parent_test" attribute of type "Testseries" (the AoTest element), and the lowermost application element of AoSubTest (here "Subsubtest") has a "children" attribute of type "Measurement". The other "parent_test" and "children" attributes of the inner AoSubTest application elements always have exactly the type of the corresponding application elements in the chain (e.g. the "parent_test" attribute of "Subtest" is of type "Maintest", and the "children" attribute of "Subtest" is of type "Subsubtest").

So the rule says: It is not allowed to have a "parent_test" relation to

an instance of another application element than the one defined in the chain as the parent test.

An analogous rule is valid for the "children" attribute.

*)

```
ENTITY AoTestAbstract
ABSTRACT SUPERTYPE OF (ONEOF (
    AoTest,
    AoSubTest ) )
SUBTYPE OF (asam_base_entity) ;
    children: SET [0:?] OF test_measurement_select; (*CHILD*)
        (* MANDATORY ATTRIBUTE *)
UNIQUE
    UR1: SELF\asam_base_entity.id;
WHERE
    WR1: unique_name_and_version(children) ;
    (* There shall not be 2 members of this set with identical name and
    version. *)
END_ENTITY;
```

```
ENTITY AoTest (* BID=36 *)
SUBTYPE OF (AoTestAbstract);
INVERSE
    environment : SET [0:1] OF AoEnvironment for tests; (*FATHER*)
UNIQUE
    UR1: SELF\asam_base_entity.name,
        SELF\asam_base_entity.version;
END_ENTITY;
```

```
ENTITY AoSubTest (* BID=2 *)
SUBTYPE OF (AoTestAbstract);
    parent_test : AoTestAbstract; (*FATHER*) (* MANDATORY ATTRIBUTE
    *)
END_ENTITY;
```

```
TYPE test_measurement_select = SELECT (
    AoSubTest,
    AoMeasurement );
END_TYPE;
```

(*

only the AoTestAbstract in the bottommost level shall have an AoMeasurement, in higher levels it shall be an AoSubTest.

*)

```
ENTITY AoMeasurement (* BID=3 *)
SUBTYPE OF (asam_base_entity);
    test : AoTestAbstract; (*FATHER*)
        (* MANDATORY ATTRIBUTE *)
    measurement_quantities : SET [0:?] OF AoMeasurementQuantity;
        (* MANDATORY ATTRIBUTE *) (*CHILD*)
    submatrices : SET [0:?] OF AoSubmatrix;
        (* MANDATORY ATTRIBUTE *) (*CHILD*)
    units_under_test : SET [0:?] OF AoUnitUnderTestAbstract;
        (*INFO_REL*)
    sequences : SET [0:?] OF AoTestSequenceAbstract;
```

```

                                (*INFO_REL*)
equipments                     : SET [0:?] OF AoTestEquipmentAbstract;
                                (*INFO_REL*)
measurement_begin              : OPTIONAL T_DATE;
measurement_end                : OPTIONAL T_DATE;
UNIQUE
    UR1: SELF\asam_base_entity.id;
WHERE
    WR1 : unique_name (measurement_quantities);
    WR2 : unique_name_and_version (submatrices);
END_ENTITY;

ENTITY AoMeasurementQuantity  (* BID=4 *)
SUBTYPE OF (asam_base_entity);
    local_columns : SET [0:?] OF AoLocalColumn;  (*INFO_FROM*)
                                (* MANDATORY ATTRIBUTE *)
    quantity      : OPTIONAL AoQuantity;  (*INFO_TO*)
                                (* MANDATORY ATTRIBUTE *)
    unit          : OPTIONAL AoUnit;  (*INFO_TO*)
                                (* MANDATORY ATTRIBUTE *)
    channel       : OPTIONAL AoTestEquipmentAbstract;  (*INFO_TO*)
    is_scaled_by  : LIST [0:?] OF AoMeasurementQuantity;
                                (* used for scaling the cells !
                                For multidimensional scaling (rank>1) the
                                rightmost index is incremented in the innermost
                                loop *)  (*INFO_FROM*)
    datatype      : datatype_enum;  (* MANDATORY ATTRIBUTE *)
    rank          : OPTIONAL T_LONG;  (*of one cell*)
    dimension     : LIST [0:?] OF T_LONG;  (*of one cell*)
    type_size     : OPTIONAL T_LONG;  (*of a scalar in byte*)
    interpolation : OPTIONAL interpolation_enum;
    minimum       : OPTIONAL T_DOUBLE;
    maximum       : OPTIONAL T_DOUBLE;
    average       : OPTIONAL T_DOUBLE;
    standard_deviation : OPTIONAL T_DOUBLE;
INVERSE
    measurement   : AoMeasurement for measurement_quantities;
                                (*FATHER*)  (* MANDATORY ATTRIBUTE *)
    scales        : SET [0:1] OF AoMeasurementQuantity for is_scaled_by;
                                (*INFO_TO*)
UNIQUE
    UR1: SELF\asam_base_entity.id;
WHERE
    WR1: SIZEOF (SELF.dimension) = SELF.rank;
        (* The attribute dimension contains so many integers as rank (the
        number of dimensions of a cell) specifies. *)
    WR2: (NOT EXISTS (SELF.is_scaled_by) ) XOR
        (SIZEOF (SELF.is_scaled_by) =SELF.rank) ;
        (* If a scaling exists, then the number of scaling vectors is given
        by rank *)
    (*
    WR3: unique_name_and_version (local_columns) ;
    This rule is commented out, because the attribute
    "local_columns" describes an INFO relation (not FATHER/CHILD) .
    *)
    WR4: SELF.datatype <> dt_unknown AND SELF.datatype <> dt_id;

```

END_ENTITY;

```

TYPE interpolation_enum = ENUMERATION OF (
    no_interpolation,
    linear_interpolation,
    application_specific );
END_TYPE;

```

```

(*)
    state is no longer used.
    It has been replaced by measurement_quantity, its namedescribes the
    desired state and the values in itslocal columns may be of type boolean.
    For logging events the type may also be string.
*)

```

```

ENTITY AoSubmatrix    (* BID = 38 *)
SUBTYPE OF (asam_base_entity);
    number_of_rows : T_LONG;    (* MANDATORY ATTRIBUTE *)
    local_columns : LIST [0:?] OF AoLocalColumn;    (*CHILD*)
                        (* MANDATORY ATTRIBUTE *)
INVERSE
    measurement : AoMeasurement for submatrices;    (*FATHER*)
                        (* MANDATORY ATTRIBUTE *)
UNIQUE
    UR1: SELF\asam_base_entity.id;
WHERE
    (*WR1: NOT EXISTS (SELF\asam_base_entity.version); *)
    WR2: unique_name_and_version(local_columns);
    (* Remark: The function compares also instances that have no version.
    *)
END_ENTITY;

```

```

(*)
    Value representation:
    local columns and sequences, possibly including generation
*)

```

```

ENTITY AoExternalComponent    (* BID=40 *)
SUBTYPE OF (asam_base_entity);
    ordinal_number : OPTIONAL T_LONG;
    component_length : T_LONG;    (* MANDATORY ATTRIBUTE *)
    filename_url : T_STRING;    (* MANDATORY ATTRIBUTE *)
    value_type : typespec_enum;    (* MANDATORY ATTRIBUTE *)
    start_offset : OPTIONAL T_LONGLONG;    (* MANDATORY ATTRIBUTE *)
    block_size : OPTIONAL T_LONG;    (* MANDATORY ATTRIBUTE *)
    valuesperblock : OPTIONAL T_LONG;    (* MANDATORY ATTRIBUTE *)
    (*value_offsets : LIST [1:?] OF T_LONG;    (old) *)
    value_offset : OPTIONAL T_LONG;    (* MANDATORY ATTRIBUTE *)
    flags_filename_url : OPTIONAL T_STRING;
    flags_start_offset : OPTIONAL T_LONGLONG;
INVERSE
    local_column : AoLocalColumn for external_component;
                        (* MANDATORY ATTRIBUTE *) (*FATHER*)
UNIQUE
    UR1: SELF\asam_base_entity.id;
WHERE

```

- (* Several external components can belong to a local column.
 They are collected in a list in attribute "external_component".
 Within this list the external components must be distinguishable by
 their ordinal_number.
 This ordinal_number shall begin with 1 and is incremented by 1 for
 each consecutive external component.
 Therefore the attribute "ordinal_number" requires the following
 rules:
- minimum ordinal_number = 1
 - maximum ordinal_number = number of external components of this local
 column
 - all ordinal_numbers of external components in a local column are
 different
 - in case of only 1 component ordinal_number can be omitted,
 - the default value is 1

For blobs we accept only external components that have exactly one
 blob per file. Therefore the 4 attributes
 'start_offset', 'block_size', 'values_per_block', 'value_offset' are
 left UNDEFINED (i.e. without value) on instantiation
 if value_type=DT_BLOB. In all other cases these attributes must be set.
 This is expressed in WR5.
 *)

WR1: SIZEOF (QUERY (ec <* SELF\local_column.external_component
 | ec.ordinal_number < 1)) = 0;
 (* For all external components in the local column of THIS external
 component shall be:
 ordinal_number < 1 does not occur
 *)

WR2: SIZEOF (QUERY (ec <* SELF\local_column.external_component
 | ec.ordinal_number > SIZEOF
 (SELF\local_column.external_component)
)) = 0;
 (* For all external components in the local column of THIS external
 component shall be:
 ordinal_number > number of components does not occur
 *)

WR3: SIZEOF (QUERY (ec <* SELF\local_column.external_component
 | ec.ordinal_number = SELF\ordinal_number)) = 1;
 (* For all external components in the local column of THIS external
 component shall be:
 ordinal_number of THIS external component occurs only once
 *)

WR4: EXISTS (ordinal_number) OR (ordinal_number=1);
 (* If ordinal_number does not exist, it is assumed = 1
 The rule for different ordinal_number (WR3) limits omission to
 only one external component.
 *)

WR5: ((EXISTS(start_offset)) AND (EXISTS(block_size)) AND (EXISTS
 (values_per_block)) AND (EXISTS(value_offset)) AND
 (value_type <> DT_BLOB))

XOR

```
((NOT EXISTS(start_offset)) AND (NOT EXISTS(block_size)) AND (NOT EXISTS
(valuesperblock)) AND (NOT EXISTS(value_offset)) AND (value_type=DT_BLOB));
END_ENTITY;
```

```
TYPE typespec_enum = ENUMERATION OF (
    dt_boolean,          (* 0 *)
    dt_byte,             (* 1 *)
    dt_short,            (* 2 *)
    dt_long,             (* 3 *)
    dt_longlong,         (* 4 *)
    ieeefloat4,          (* 5 *)
    ieeefloat8,          (* 6 *)
    dt_short_beo,        (* 7 *)
    dt_long_beo,         (* 8 *)
    dt_longlong_beo,     (* 9 *)
    ieeefloat4_beo,      (* 10 *)
    ieeefloat8_beo,      (* 11 *)
    dt_string,           (* 12 *)
    dt_bytestr,          (* 13 *)
    dt_blob,             (* 14 *)
    dt_boolean_flags_beo, (* 15 *) (* mark only endianness of flags
*)
    dt_byte_flags_beo,   (* 16 *) (* mark only endianness of flags
*)
    dt_string_flags_beo, (* 17 *) (* mark only endianness of flags
*)
    dt_bytestr_beo,      (* 18 *)
    dt_sbyte,            (* 19 *)
    dt_sbyte_flags_beo,  (* 20 *)
    dt_ushort,           (* 21 *)
    dt_ushort_beo,       (* 22 *)
    dt_ulong,            (* 23 *)
    dt_ulong_beo );      (* 24 *)
END_TYPE;
```

(*

Explanation of dt_XXX_beo: The endianness of flags (their type is always "dt_short") is the same as for the values.

If the values are booleans/bytes/strings, endianness for values is insignificant, but it is given to ensure proper reading of the flags.

*)

```
TYPE seq_rep_enum = ENUMERATION OF (
    explicit,            (* 0 *)
    implicit_constant,   (* 1 *)
    implicit_linear,     (* 2 *)
    implicit_saw,        (* 3 *)
    raw_linear,          (* 4 *)
    raw_polynomial,      (* 5 *)
    formula,             (* 6 *) (* no longer used since 5.1 *)
    external_component,  (* 7 *)
    raw_linear_external, (* 8 *)
    raw_polynomial_external, (* 9 *)
    raw_linear_calibrated, (*extension March 2003*) (* 10 *)
```

```

        raw_linear_calibrated_external                      (* 11 *)
        (* extendible!! *) );
END_TYPE;

(*
The seq_rep_enum determines what can be expected to be in the attributes
of AoLocalColumn. The following table gives an overview:
attribute
seq_rep_enum generation_parameters values      external_component
explicit      - - -                      final values      - - -
implicit      gener.param                  gener.param          - - -
raw            gener.param                  raw values           - - -
ext_comp      - - -                      values from file ext.comp.description
raw+extern    gener.param                  raw values ext.comp.description

The generation parameters shall contain for
explicit      ---
implicit_constant  constant value (offset)
implicit_linear   start_value+increment
implicit_saw      start_value+increment+number_of_values_per_saw
raw_linear        offset + factor
raw_polynomial    N (order) + coeff0 + coeff1 + ... + coeffN
external_component ---
raw_linear_external      const.value+gradient (offset+factor)
raw_polynomial_external  N (order) + coeff0 + coeff1 + ... + coeffN
raw_linear_calibrated    offset + factor + calibration
raw_linear_calibrated_external  offset + factor + calibration

The data shall contain for
explicit      all the data
implicit_constant  constant value (offset)
implicit_linear   start_value+increment (offset+factor)
implicit_saw      start_value+increment+number_of_values_per_saw
raw_linear        raw data
raw_polynomial    raw data
external_component all the data
raw_linear_external raw data
raw_polynomial_external  raw data
raw_linear_calibrated    raw data
raw_linear_calibrated_external raw data

The external_component is given for
external_component
raw_linear_external
raw_polynomial_external
raw_linear_calibrated_external
and empty for all other choices.
*)

ENTITY AoLocalColumn      (* BID = 39 *)
SUBTYPE OF (asam_base_entity);
    flags                  : LIST [0:?] OF T_SHORT;
    global_flag             : OPTIONAL T_SHORT;
    independent             : T_SHORT;      (* MANDATORY ATTRIBUTE *)
    minimum                 : OPTIONAL T_DOUBLE;
    maximum                 : OPTIONAL T_DOUBLE;

```



```

sequence_representation : seq_rep_enum; (* MANDATORY ATTRIBUTE *)
generation_parameters   : LIST [0:?] OF T_DOUBLE;
raw_datatype            : OPTIONAL datatype_enum;
values                  : OPTIONAL value_sequence;
external_component      : LIST [0:?] OF AoExternalComponent;
                        (*CHILD*)

INVERSE
    submatrix            : AoSubmatrix for local_columns;
                        (* MANDATORY ATTRIBUTE *) (*FATHER*)
    measurement_quantity : AoMeasurementQuantity for local_columns;
                        (* MANDATORY ATTRIBUTE *) (*INFO_TO*)

UNIQUE
    UR1: SELF\asam_base_entity.id;

WHERE
    WR1: (SIZEOF (QUERY (loc <* SELF.submatrix.local_columns
        |loc.independent=1) ) <= 1) AND
        (SIZEOF (QUERY (loc <* SELF.submatrix.local_columns
        |loc.independent=1) ) >= 0) ;
    (*
        Explanation of WR1:
        Only zero or one local column per submatrix can be independent
        (have the flag "independent"=1) .
    *)
    WR1a: (
        (SELF.independent>=0 AND SELF.independent<=1)
        AND (
            (SELF.measurement_quantity.datatype=DT_DATE) OR
            (SELF.measurement_quantity.datatype=DT_BYTE) OR
            (SELF.measurement_quantity.datatype=DT_SHORT) OR
            (SELF.measurement_quantity.datatype=DT_LONG) OR
            (SELF.measurement_quantity.datatype=DT_LONGLONG)
        )
    OR
        (SELF.measurement_quantity.datatype=DT_FLOAT) OR
        (SELF.measurement_quantity.datatype=DT_DOUBLE) )
    )
    XOR
    ( SELF.independent=0 ) ;
    (*
        Explanation of WR1a:
        Only numeric (except complex) or time data can be in an
        independent
        column (have the flag "independent"=1) .
    *)

    WR2: (NOT EXISTS (SELF.flags) ) XOR
        (SIZEOF (SELF.flags) =submatrix.number_of_rows) ;
    (*
        Explanation of WR2:
        Whenever the attribute "flags" is given, its length must be
        number_of_rows. (See also WR6)
    *)

    WR3: (SELF.sequence_representation = implicit_constant)
        XOR
        (SELF.sequence_representation = implicit_linear)
        XOR

```

```

        (SELF.sequence_representation = implicit_saw)
        XOR
        (sequence_length (SELF.values) = (*explicit sequences*)
        local_column_list_length ( (* and raw data*)
            SELF.submatrix.number_of_rows,
            SELF.measurement_quantity.rank,
            SELF.measurement_quantity.dimension) );
    (*
    Explanation of WR3:
    In the attribute "values" there is a value_sequence, which has
    in turn an attribute "values" which is a list.
    This where rule is to ensure the right length of that list. For
    this we need two information items:
    1. from AoSubmatrix given in the (inv) attribute "submatrix":
        its attribute "number_of_rows" (an integer);
    2. from AoMeasurementQuantity given in the (inv)
        attributemeasurement_quantity:
        2a. its attribute "rank" (an integer)
        2b. its attribute "dimension" (a list of integer)
    Now the required list length can be calculated, the algorithm
    is contained in the function "local_column_list_length":
        list_length = number_of_rows;
        FOR i=1...rank (step 1) [nothing happens if rank=0]
            list_length = list_length * dimension[i-1];
        END_FOR;
    The required list length is then compared to the actual list length
    provided by function "sequence_length", which is also called here.
    In case of implicit sequences the function shall not be evaluated.
    *)

WR4: ( ( SELF.sequence_representation = explicit ) AND
        ( ( EXISTS (SELF.values) ) AND
          ( NOT ( EXISTS (SELF.generation_parameters) ) )
        )
      )
    XOR
    (
      ( (SELF.sequence_representation = implicit_constant)
        XOR (SELF.sequence_representation = implicit_linear)
        XOR (SELF.sequence_representation = implicit_saw)
      )
      AND
      ( EXISTS (SELF.generation_parameters) )
    )
    XOR
    (
      ( (SELF.sequence_representation = raw_linear)
        XOR (SELF.sequence_representation = raw_polynomial)
        XOR (SELF.sequence_representation =
              raw_linear_calibrated)
      )
      AND
      ( ( EXISTS (SELF.values) ) AND
        ( EXISTS (SELF.generation_parameters) )
      )
    )
  )

```

```

XOR
(
  ( (SELF.sequence_representation = external_component)
    XOR (SELF.sequence_representation = raw_linear_external)
    XOR (SELF.sequence_representation =
      raw_linear_calibrated_external)
    XOR (SELF.sequence_representation =
      raw_polynomial_external)
  )
  AND
  (EXISTS (SELF.external_component) )
);
(*)
Explanation of WR4:
For explicit and raw data the attribute "values" is used;
for implicit and raw data the attribute
"generation_parameters" is used.
For data on external components the
attribute "external_component" is used, others may be used
according to the type.
*)

WR5: ( ( (SELF.sequence_representation = implicit_constant)
  XOR (SELF.sequence_representation = implicit_linear)
  XOR (SELF.sequence_representation = implicit_saw)
  XOR (SELF.sequence_representation = raw_linear)
  XOR (SELF.sequence_representation =
    raw_linear_calibrated)
  XOR (SELF.sequence_representation = raw_polynomial)
  XOR (SELF.sequence_representation = raw_linear_external)
  XOR (SELF.sequence_representation =
    raw_linear_calibrated_external)
  XOR (SELF.sequence_representation =
    raw_polynomial_external)
  )
  AND
  ( 'numeric_sequence' IN TYPEOF (SELF.values) )
  )
  XOR ( SELF.sequence_representation = explicit)
  XOR ( 'string_sequence' IN TYPEOF (SELF.values) );
(*)
Explanation of WR5:
For explicit data the type of the sequence may be any;
for implicit and raw data the sequence must be numeric.
*)

WR6: (EXISTS (SELF.global_flag) ) XOR (EXISTS (SELF.flags) );
(*)
Explanation of WR6:
Either flags or global_flag is given. (See also WR2)
*)

WR7: (NOT EXISTS (SELF.raw_datatype) ) XOR
  (SELF.raw_datatype <> dt_unknown AND
  SELF.raw_datatype <> dt_id);
END_ENTITY;

```

```

ENTITY value_sequence
ABSTRACT SUPERTYPE OF (ONEOF (
    numeric_sequence,
    textual_sequence) );
END_ENTITY;

ENTITY textual_sequence
ABSTRACT SUPERTYPE OF (ONEOF (
    bytestr_sequence,
    (*blob_sequence, (removed 12.12.2002) *)
    string_sequence) )
SUBTYPE OF (value_sequence);
END_ENTITY;

ENTITY numeric_sequence
ABSTRACT SUPERTYPE OF (ONEOF (
boolean_sequence,
    byte_sequence,
    short_sequence,
    long_sequence,
    longlong_sequence,
    float_sequence,
    double_sequence,
    complex_sequence,
    dcomplex_sequence,
    time_sequence) )
SUBTYPE OF (value_sequence);
END_ENTITY;

ENTITY boolean_sequence
SUBTYPE OF (numeric_sequence);
    values : LIST [1:?] OF T_BOOLEAN;
END_ENTITY;

ENTITY byte_sequence
SUBTYPE OF (numeric_sequence);
    values : LIST [1:?] OF T_BYTE;
END_ENTITY;

ENTITY short_sequence
SUBTYPE OF (numeric_sequence);
    values : LIST [1:?] OF T_SHORT;
END_ENTITY;

ENTITY long_sequence
SUBTYPE OF (numeric_sequence);
    values : LIST [1:?] OF T_LONG;
END_ENTITY;

ENTITY longlong_sequence
SUBTYPE OF (numeric_sequence);
    values : LIST [1:?] OF T_LONGLONG;
END_ENTITY;

ENTITY float_sequence

```

```

SUBTYPE OF (numeric_sequence) ;
    values : LIST [1:?] OF T_FLOAT;
END_ENTITY;

ENTITY double_sequence
SUBTYPE OF (numeric_sequence) ;
    values : LIST [1:?] OF T_DOUBLE;
END_ENTITY;

ENTITY complex_sequence
SUBTYPE OF (numeric_sequence) ;
    values : LIST [2:?] OF T_FLOAT;
    (* pairs of real and imaginary part are stored consecutively *)
WHERE
    WR1: SIZEOF (values) MOD 2 = 0;
END_ENTITY;

ENTITY dcomplex_sequence
SUBTYPE OF (numeric_sequence) ;
    values : LIST [2:?] OF T_DOUBLE;
    (* pairs of real and imaginary part are stored consecutively *)
WHERE
    WR1: SIZEOF (values) MOD 2 = 0;
END_ENTITY;

ENTITY time_sequence
    (* Derive no further subtypes! *)
SUBTYPE OF (numeric_sequence) ;
    values : LIST [1:?] OF T_DATE;
END_ENTITY;

ENTITY string_sequence
    (* Derive no further subtypes! *)
SUBTYPE OF (textual_sequence) ;
    values : LIST [1:?] OF T_STRING;
END_ENTITY;

ENTITY bytestr_sequence
    (* Derive no further subtypes! *)
SUBTYPE OF (textual_sequence) ;
    values : LIST [1:?] OF T_BYTESTR;
END_ENTITY;

    (*
        The 3 hierarchical entities
        AoUnitUnderTestAbstract,
        AoTestSequenceAbstract,
        AoTestEquipmentAbstract
        are constructed in the same way. The toplevel is always given by a separate
        entity, an arbitrary number of other levels may follow. Hierarchical
        relation on instance level is ensured by references in both directions.
    *)

ENTITY AoUnitUnderTestAbstract
ABSTRACT SUPERTYPE OF (ONEOF (
    AoUnitUnderTest,

```

```

        AoUnitUnderTestPart ) )
SUBTYPE OF (asam_base_entity);
    children : SET [0:?] OF AoUnitUnderTestPart; (*CHILD*)
INVERSE
    measurement : SET[0:?] OF AoMeasurement for units_under_test;
        (*INFO_REL*)
UNIQUE
    UR1: SELF\asam_base_entity.id;
WHERE
    WR1: unique_name_and_version (children);
        (*WR2: EXISTS (version_date); *)
END_ENTITY;

ENTITY AoUnitUnderTest (* BID=21 *)
SUBTYPE OF (AoUnitUnderTestAbstract);
INVERSE
    environment : SET [0:1] OF AoEnvironment for uuts; (*FATHER*)
UNIQUE
    UR1: SELF\asam_base_entity.name,
        SELF\asam_base_entity.version;
END_ENTITY;

ENTITY AoUnitUnderTestPart (* BID=22 *)
SUBTYPE OF (AoUnitUnderTestAbstract);
INVERSE
    parent_unit_under_test : AoUnitUnderTest for children; (*FATHER*)
    parent_unit_under_test_part : AoUnitUnderTestPart for children;
        (*FATHER*)
WHERE
    WR1: (NOT EXISTS (parent_unit_under_test) ) XOR
        (NOT EXISTS (parent_unit_under_test_part) );
END_ENTITY;

ENTITY AoTestSequenceAbstract
ABSTRACT SUPERTYPE OF (ONEOF (
    AoTestSequence,
    AoTestSequencePart ) )
SUBTYPE OF (asam_base_entity);
    children : SET [0:?] OF AoTestSequencePart; (*CHILD*)
INVERSE
    measurement : SET[0:?] OF AoMeasurement for sequences;
        (*INFO_REL*)
UNIQUE
    UR1: SELF\asam_base_entity.id;
WHERE
    WR1: unique_name_and_version (children);
        (*WR2: EXISTS (version_date); *)
END_ENTITY;

ENTITY AoTestSequence (* BID=25 *)
SUBTYPE OF (AoTestSequenceAbstract);
INVERSE
    environment : SET [0:1] OF AoEnvironment for sequences; (*FATHER*)
UNIQUE
    UR1: SELF\asam_base_entity.name,
        SELF\asam_base_entity.version;

```

END_ENTITY;

ENTITY AoTestSequencePart (* BID=26 *)

SUBTYPE OF (AoTestSequenceAbstract);

INVERSE

parent_sequence : AoTestSequence for children; (*FATHER*)

parent_sequence_part : AoTestSequencePartfor children; (*FATHER*)

WHERE

WR1: (NOT EXISTS (parent_sequence)) XOR

(NOT EXISTS (parent_sequence_part));

END_ENTITY;

ENTITY AoTestEquipmentAbstract

ABSTRACT SUPERTYPE OF (ONEOF (

AoTestEquipment,

AoTestEquipmentPart))

SUBTYPE OF (asam_base_entity);

children : SET [0:?] OF AoTestEquipmentPart; (*CHILD*)

INVERSE

measurement : SET[0:?] OF AoMeasurement for equipments;

(*INFO_REL*)

measurement_quantities : SET[0:?] OF AoMeasurementQuantity for
channel; (*INFO_FROM*)

UNIQUE

UR1: SELF\asam_base_entity.id;

WHERE

WR1: unique_name_and_version (children);

(*WR2: EXISTS (version_date); *)

END_ENTITY;

ENTITY AoTestEquipment (* BID=23 *)

SUBTYPE OF (AoTestEquipmentAbstract);

INVERSE

environment : SET [0:1] OF AoEnvironment for equipments;

(*FATHER*)

UNIQUE

UR1: SELF\asam_base_entity.name,

SELF\asam_base_entity.version;

END_ENTITY;

ENTITY AoTestEquipmentPart (* BID=24 *)

SUPERTYPE OF (ONEOF (

AoTestDevice))

SUBTYPE OF (AoTestEquipmentAbstract);

INVERSE

parent_equipment : AoTestEquipment for children; (*FATHER*)

parent_equipment_part : AoTestEquipmentPartfor children; (*FATHER*)

WHERE

WR1: (NOT EXISTS (parent_equipment)) XOR

(NOT EXISTS (parent_equipment_part));

(* This formulation of inverse attribute ensures that
oneparent_equipment is always required. It is either of
typeAoTestEquipment or of type AoTestEquipmentPart. The firstof
the two has a relation to AoEnvironment.*)

END_ENTITY;

```

(*)
    The following entity AoTestDevice shall be the link for ASAM GDI.
    Further subtypes (application elements) can be derived from it in any
    number of levels.
*)

ENTITY AoTestDevice (* BID=37 *)
SUBTYPE OF (AoTestEquipmentPart);
WHERE
    WR1: SIZEOF (QUERY (child <* SELF\AoTestEquipmentPart.children |
        (NOT ('AoTestDevice' IN TYPEOF (child)) ) ) ) = 0;
    (* The children of AoTestDevice must have type
       AoTestDevice (and not AoTestEquipment...) *)
END_ENTITY;

ENTITY AoQuantity (* BID=11 *)
SUBTYPE OF (asam_base_entity);
successors : SET [0:?] OF AoQuantity; (*CHILD*)
    (* The following 6 attributes are OPTIONAL since Version 5.1 *)
    (* They were MANDATORY ATTRIBUTES since Version 4.0 *)

    default_unit : OPTIONAL AoUnit; (*INFO_TO*)
    default_rank : OPTIONAL T_LONG;
    default_dimension : LIST [0:?] OF T_LONG;
    default_datatype : OPTIONAL datatype_enum;
    default_type_size : OPTIONAL T_LONG;
        (* for strings/bytestreams: maximum_length *)
    default_mq_name : OPTIONAL T_STRING;
INVERSE
    predecessor : SET [0:1] OF AoQuantity FOR successors; (*FATHER*)
    groups : SET [0:?] OF AoQuantityGroup for quantities;
        (*INFO_REL*)
    measurement_quantities: SET [0:?] OF AoMeasurementQuantity for
        quantity; (*INFO_FROM*)
UNIQUE
    UR1: SELF\asam_base_entity.id;
    UR2: SELF\asam_base_entity.name;
WHERE
    WR1: SIZEOF (SELF.default_dimension) = SELF.default_rank;
    WR2: (NOT EXISTS (default_datatype)) XOR
        (SELF.default_datatype <> dt_unknown
        ANDSELF.default_datatype <> dt_id);
    WR3: unique_name_and_version (successors);
END_ENTITY;

ENTITY AoUnit (* BID=13 *)
SUBTYPE OF (asam_base_entity);
    phys_dimension : AoPhysicalDimension; (*INFO_TO*)
        (* MANDATORY ATTRIBUTE *)
    factor : T_DOUBLE; (* MANDATORY ATTRIBUTE *)
    offset : T_DOUBLE; (* MANDATORY ATTRIBUTE *)
INVERSE
    groups : SET [0:?] OF AoUnitGroup for units; (*INFO_REL*)
    quantities : SET [0:?] OF AoQuantity for default_unit;
        (*INFO_FROM*)
    measurement_quantities : SET [0:?] OF AoMeasurementQuantity

```



```

        for unit;    (*INFO_FROM*)    (* MANDATORY ATTRIBUTE
*)
    parameters : SET [0:?] OF AoParameter for unit;    (*INFO_FROM*)
UNIQUE
    UR1: SELF\asam_base_entity.id;
    UR2: SELF\asam_base_entity.name,phys_dimension;
END_ENTITY;

(*
    factor is the (value using SI unit) when (value using this unit) = 1
    offset is the (value using SI unit) when (value using this unit) = 0
    :
    (value using this unit) multiplied by (factor) = (value using SI unit)
    (value using this unit) plus (offset) = (value using SI unit)
    (value using this unit) multiplied by (factor)
    plus (offset) = (value using SI unit)
    :
    Examples:
    (4200) 1/min= (4200 * factor) 1/s      [factor=0.016667]
              = (70) 1/s
    (20) deg.C = (20 + offset) K           [offset=273.15]
              = (293.15) K
    (68) deg.F = (68 * factor + offset) K   [factor=0.55555]
              = (293,15) K                 [offset=255.372]
    (200) mm = (200 * factor) m = 0.2 m     [factor=0.001]
*)

ENTITY AoPhysicalDimension (* BID=15 *)
SUBTYPE OF (asam_base_entity);
    length_exp           : T_LONG;    (* MANDATORY ATTRIBUTE *)
    mass_exp             : T_LONG;    (* MANDATORY ATTRIBUTE *)
    time_exp             : T_LONG;    (* MANDATORY ATTRIBUTE *)
    current_exp          : T_LONG;    (* MANDATORY ATTRIBUTE *)
    temperature_exp      : T_LONG;    (* MANDATORY ATTRIBUTE *)
    molar_amount_exp     : T_LONG;    (* MANDATORY ATTRIBUTE *)
    luminous_intensity_exp : T_LONG;  (* MANDATORY ATTRIBUTE *)
    length_exp_den       : OPTIONAL T_LONG;
    mass_exp_den         : OPTIONAL T_LONG;
    time_exp_den         : OPTIONAL T_LONG;
    current_exp_den      : OPTIONAL T_LONG;
    temperature_exp_den  : OPTIONAL T_LONG;
    molar_amount_exp_den : OPTIONAL T_LONG;
    luminous_intensity_exp_den : OPTIONAL T_LONG;
INVERSE
    units : SET[0:?] OF AoUnit FOR phys_dimension;    (*INFO_FROM*)
UNIQUE
    UR1: SELF\asam_base_entity.id;
    UR2: SELF\asam_base_entity.name;
WHERE
    (* ensure default_value=1 *)
    WR1: EXISTS (length_exp_den) OR (length_exp_den=1);
    WR2: EXISTS (mass_exp_den) OR (mass_exp_den=1);
    WR3: EXISTS (time_exp_den) OR (time_exp_den=1);
    WR4: EXISTS (current_exp_den) OR (current_exp_den=1);
    WR5: EXISTS (temperature_exp_den) OR (temperature_exp_den=1);
    WR6: EXISTS (molar_amount_exp_den) OR (molar_amount_exp_den=1);

```

```

        WR7: EXISTS (luminous_intensity_exp_den) OR
            (luminous_intensity_exp_den=1);
        (* ensure denominators not equal zero *)
        WR8 : length_exp_den<>0;
        WR9 : mass_exp_den<>0;
        WR10: time_exp_den<>0;
        WR11: current_exp_den<>0;
        WR12: temperature_exp_den<>0;
        WR13: molar_amount_exp_den<>0;
        WR14: luminous_intensity_exp_den<>0;
    END_ENTITY;

    ENTITY AoQuantityGroup (* BID=12 *)
    SUBTYPE OF (asam_base_entity);
        quantities : SET [0:?] OF AoQuantity;    (*INFO_REL*)
                                                    (* MANDATORY ATTRIBUTE *)
    UNIQUE
        UR1: SELF\asam_base_entity.id;
        UR2: SELF\asam_base_entity.name;
    END_ENTITY;

    ENTITY AoUnitGroup (* BID=14 *)
    SUBTYPE OF (asam_base_entity);
        units : SET [0:?] OF AoUnit;    (* MANDATORY ATTRIBUTE *) (*INFO_REL*)
    UNIQUE
        UR1: SELF\asam_base_entity.id;
        UR2: SELF\asam_base_entity.name;
    END_ENTITY;

    ENTITY AoLog (* BID=43 *)
    SUBTYPE OF (asam_base_entity);
        date : T_DATE;
        children : SET [0:?] OF AoLog;    (*CHILD*)
    INVERSE
        parent : SET [0:1] OF AoLog for children;    (*FATHER*)
    UNIQUE
        UR1: SELF\asam_base_entity.id;
        UR2: SELF\asam_base_entity.name,
            date,
            parent;
    WHERE
        WR1: unique_name_and_version (children);
    END_ENTITY;

    ENTITY AoAny (* BID=0 *)
    SUBTYPE OF (asam_base_entity);
        (* Allowed to build a hierachy within the AoAny. *)
        children : SET [0:?] OF AoAny;    (*CHILD*)
    INVERSE
        parent : SET [0:1] OF AoAny for children;    (*FATHER*)
    UNIQUE
        UR1: SELF\asam_base_entity.id;
        UR2: SELF\asam_base_entity.name,
            SELF\asam_base_entity.version,
            parent;
        (* if parent does not exists then unique (name,version) *)

```

```

WHERE
    WR1: unique_name_and_version (children) ;
END_ENTITY;

(*)
    Also AoAny should not be instantiated, instead entities should
    be derived from it.
*)

(*-----SECURITY SECTION-----*)

ENTITY AoUser (* BID=34 *)
SUBTYPE OF (asam_base_entity) ;
    password : T_STRING; (* MANDATORY ATTRIBUTE *)
INVERSE
    groups : SET [0:?] Of AoUserGroup for users; (*INFO_REL*)
                                                (* MANDATORY ATTRIBUTE *)
UNIQUE
    UR1: SELF\asam_base_entity.id;
    UR2: SELF\asam_base_entity.name;
END_ENTITY;

ENTITY AoUserGroup (* BID=35 *)
SUBTYPE OF (asam_base_entity) ;
    users : SET [0:?] Of AoUser; (*INFO_REL*)
                                                (* MANDATORY ATTRIBUTE *)
    superuser_flag : T_SHORT; (* MANDATORY ATTRIBUTE *)
UNIQUE
    UR1: SELF\asam_base_entity.id;
    UR2: SELF\asam_base_entity.name,
        SELF\asam_base_entity.version;
END_ENTITY;

ENTITY ACL (* not asam_base_entities !! *)
ABSTRACT SUPERTYPE OF (ONEOF (
    ACLI,
    ACLA,
    ACLTemplate
) ) ;
    users : AoUserGroup; (*INFO_TO*)
    appl_element_name : T_STRING;
    rights : T_SHORT;
END_ENTITY;

ENTITY ACLI
SUBTYPE OF (ACL) ;
    instance_id : T_LONGLONG;
END_ENTITY;

ENTITY ACLA
SUBTYPE OF (ACL) ;
    attribute_name : T_STRING;
    (* if empty: Whole application element with all attributes *)
END_ENTITY;

ENTITY ACLTemplate

```

```

SUBTYPE OF (ACL) ;
    instance_id : T_LONGLONG;
    ref_appl_elem_name: T_STRING;
    (* The table SVCTPLI contains:
        USERGROUPID   Reference to a UserGroup instance
        AID            Application element id
        RIGHTS         Rights value (bitmasked)
        IID            Application instance id
        REFAID         Referencing application element
    *)
END_ENTITY;

ENTITY SecurityLevel;
    appl_element_name : T_STRING;
    level : T_SHORT;
    (*
        bit decimal      ACL on      Protection on
        0      1      appl elem    all instances
        1      2      inst elem    instances individually
        2      4      elem attrs   attributes individually
        Combination by adding the decimal values.
    *)
END_ENTITY;

ENTITY InitialRightsAt;
    (* This is to be contained in table SVCATTR, column ACLREF *)
    appl_element_name : T_STRING;
    attribute_name : T_STRING;
END_ENTITY;

(*)
----- FUNCTIONS -----*)
(*)
    The following function is called from AoLocalColumn. It is used to
    determine the actual size of its value sequences.
*)

FUNCTION sequence_length (
    seq : value_sequence
) : INTEGER;

LOCAL
    list_length : INTEGER;
END_LOCAL;

    list_length := 0;
    IF 'boolean_sequence' IN TYPEOF (seq) THEN
        list_length := SIZEOF (seq\boolean_sequence.values);
    END_IF;

    IF 'byte_sequence' IN TYPEOF (seq) THEN
        list_length := SIZEOF (seq\byte_sequence.values);
    END_IF;

    IF 'short_sequence' IN TYPEOF (seq) THEN
        list_length := SIZEOF (seq\short_sequence.values);
    END_IF;

```

```

IF 'long_sequence' IN TYPEOF (seq) THEN
    list_length := SIZEOF (seq\long_sequence.values);
END_IF;

IF 'longlong_sequence' IN TYPEOF (seq) THEN
    list_length := SIZEOF (seq\longlong_sequence.values);
END_IF;

IF 'float_sequence' IN TYPEOF (seq) THEN
    list_length := SIZEOF (seq\float_sequence.values);
END_IF;

IF 'double_sequence' IN TYPEOF (seq) THEN
    list_length := SIZEOF (seq\double_sequence.values);
END_IF;

IF 'complex_sequence' IN TYPEOF (seq) THEN
    list_length := SIZEOF (seq\complex_sequence.values) / 2;
END_IF;

IF 'dcomplex_sequence' IN TYPEOF (seq) THEN
    list_length := SIZEOF (seq\dcomplex_sequence.values) / 2;
END_IF;

IF 'time_sequence' IN TYPEOF (seq) THEN
    list_length := SIZEOF (seq\time_sequence.values);
END_IF;

IF 'string_sequence' IN TYPEOF (seq) THEN
    list_length := SIZEOF (seq|string_sequence.values);
END_IF;

IF 'bytestr_sequence' IN TYPEOF (seq) THEN
    list_length := SIZEOF (seq\bytestr_sequence.values);
END_IF;

RETURN (list_length);
END_FUNCTION;

```

(*

The following function is called from AoLocalColumn. It is used to determine the required size of its value sequences.

*)

```

FUNCTION local_column_list_length (
    number_of_rows : INTEGER;
    rank           : INTEGER;
    dimension      : LIST OF INTEGER
) : INTEGER;

LOCAL
    list_length : INTEGER;
    i           : INTEGER;
    temp        : INTEGER;
END_LOCAL;

list_length := number_of_rows;

```

```

    REPEAT i := 1 TO rank BY 1; (*nothing happens if rank=0*)
        temp := list_length * dimension[i-1];
        list_length := temp;
    END_REPEAT;
    RETURN (list_length);
END_FUNCTION;

```

(*

The following function is called if the members of a set should be distinct in name+version. Since "version" is optional, the function also works without it, then it checks only that "name" is different.

The function is here given in Express. It is understood that an implementation in a normal programming language will look different. But the action of the function should be the same:

- take the set passed in the argument "instances"
- look if any two members of the set have the same name
 - * if NO return TRUE
 - * if yes look if those members differ at least in the version:
 - if YES return TRUE otherwise return FALSE

It is planned to call this function only in cases where the uniqueness of set members matters, i.e. when building hierarchical trees.

*)

```

FUNCTION unique_name_and_version (
    instances : AGGREGATE OF asam_base_entity
) : BOOLEAN;

LOCAL
    set_size      : INTEGER;
    uniqueness     : BOOLEAN;
    version1       : STRING;
    version2       : STRING;
    inst1          : asam_base_entity;
    inst2          : asam_base_entity;
END_LOCAL;

uniqueness := TRUE;
set_size := SIZEOF (instances);
REPEAT i := 1 TO set_size BY 1;
    version1 := ''; (*modified 13.12.2002*)
    inst1 := instances[i];
    IF EXISTS (inst1.version) THEN
        version1 := inst1.version;
    END_IF;
    REPEAT j := i+1 TO set_size BY 1;
        inst2 := instances[j];
        IF inst1.name = inst2.name THEN
            version2 := ''; (*modified 13.12.2002*)
            IF EXISTS (inst2.version) THEN
                version2 := inst2.version;
            END_IF;
            IF version1 = version2 THEN
                uniqueness := FALSE;
                RETURN (uniqueness);
            END_IF;
        END_IF;
    END_REPEAT;
END_REPEAT;

```

```

        RETURN (uniqueness);
END_FUNCTION;

(*
  Similar function, limited to attribute "name" only.
*)

FUNCTION unique_name (
  instances : AGGREGATE OF asam_base_entity
) : BOOLEAN;
LOCAL
  set_size      : INTEGER;
  uniqueness    : BOOLEAN;
  inst1         : asam_base_entity;
  inst2         : asam_base_entity;
END_LOCAL;
  uniqueness := TRUE;
  set_size := SIZEOF (instances);
  REPEAT i := 1 TO set_size BY 1;
    inst1 := instances[i];
    REPEAT j := i+1 TO set_size BY 1;
      inst2 := instances[j];
      IF inst1.name = inst2.name THEN
        uniqueness := FALSE;
        RETURN (uniqueness);
      END_IF;
    END_REPEAT;
  END_REPEAT;
  RETURN (uniqueness);
END_FUNCTION;

END_SCHEMA;

```

7.9.4 用 EXPRESS-G 图表达的基础模型

以下是前述 Express 模型的图形表达，即这些图形是从主模型中派生出来的。当这两种表述之间存在差异时，以文本 EXPRESS 模型为准。请注意，图中没有表达出规则和函数，尽管它们对模型的使用很重要。

EXPRESS-G 语言的说明见图 55。EXPRESS-G 语言符号说明见图 56。

父类和子类型的关系，如果标“(abs)”则父类不能被实例化。

所有的关系都有一个方向，该方向由该行末尾的球表示。

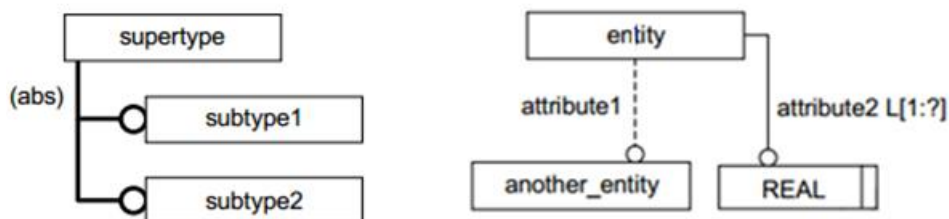


图55 EXPRESS-G 语言的说明

属性关系：

——实线表示强制属性；

——虚线表示可选属性；

——线上都标有属性名称。

L[a: b] = 表示至少有 a 个最多有 b 个成员的列表。

S[a: b] = 表示至少有 a 个最多有 b 个成员的集合。

INV 表示反向的属性（即相反方向的属性）。

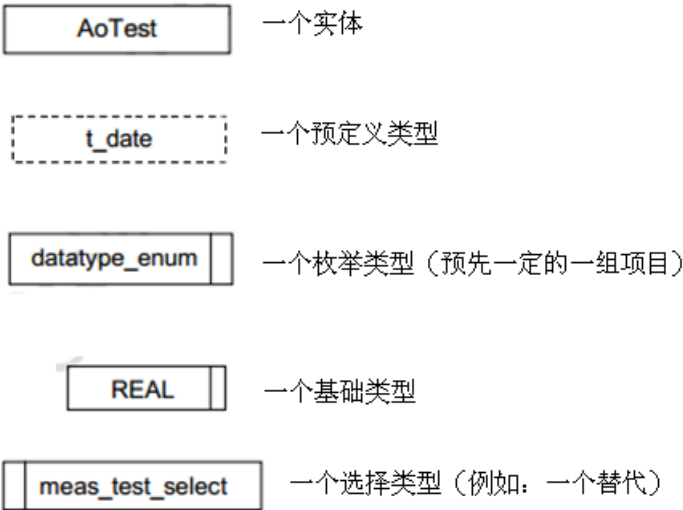


图56 EXPRESS-G 符号说明

在图57、图58中，LIST[...] 由L[...]代表，SET[: ...]由S[...]代表。

在图57-图65（从“Environment”到“Parameter, Log, External Component”）表示ODS 基础模型各个特定部分，图中部分内容有重叠。

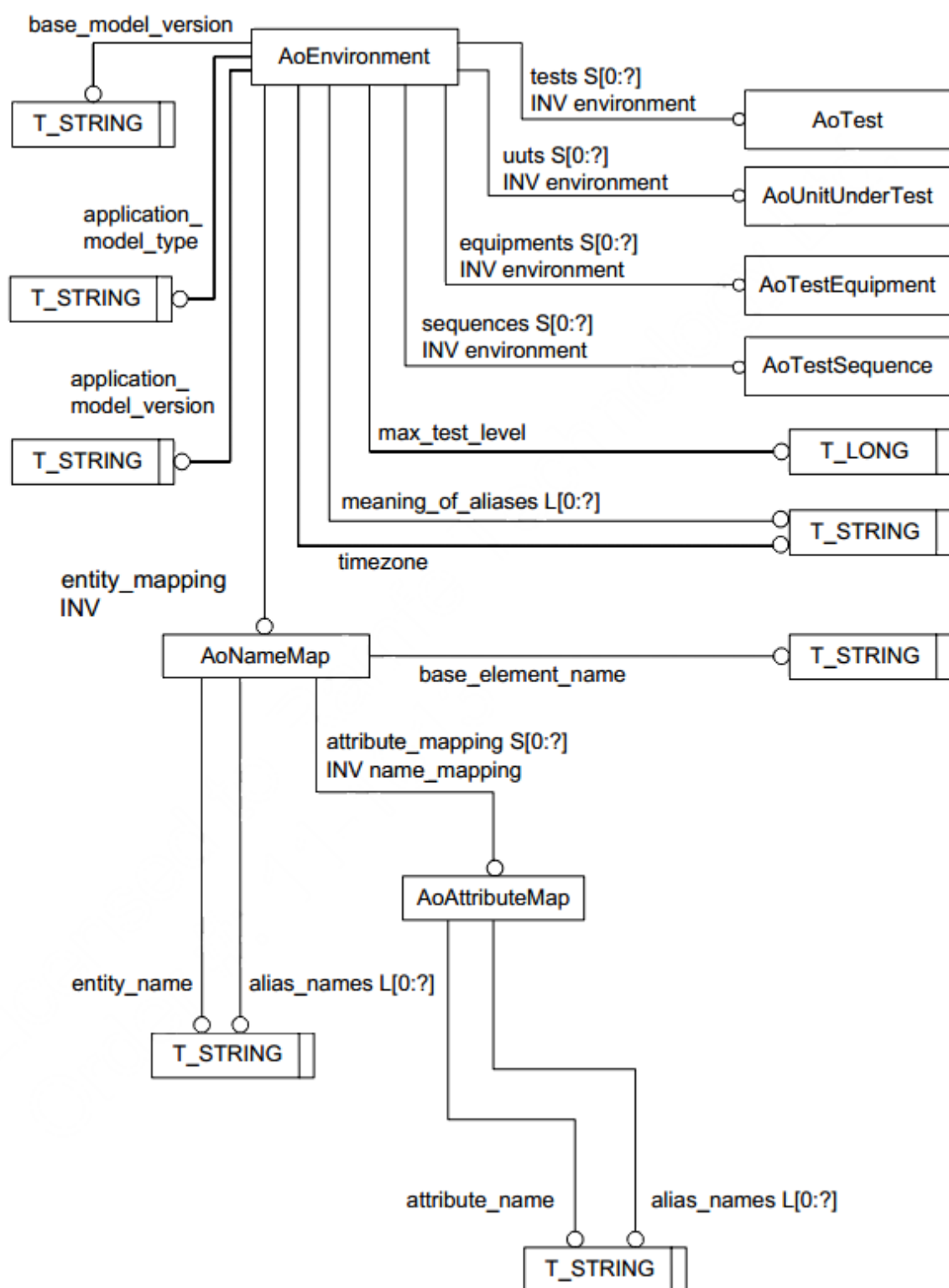


图57 EXPRESS-G 图 环境

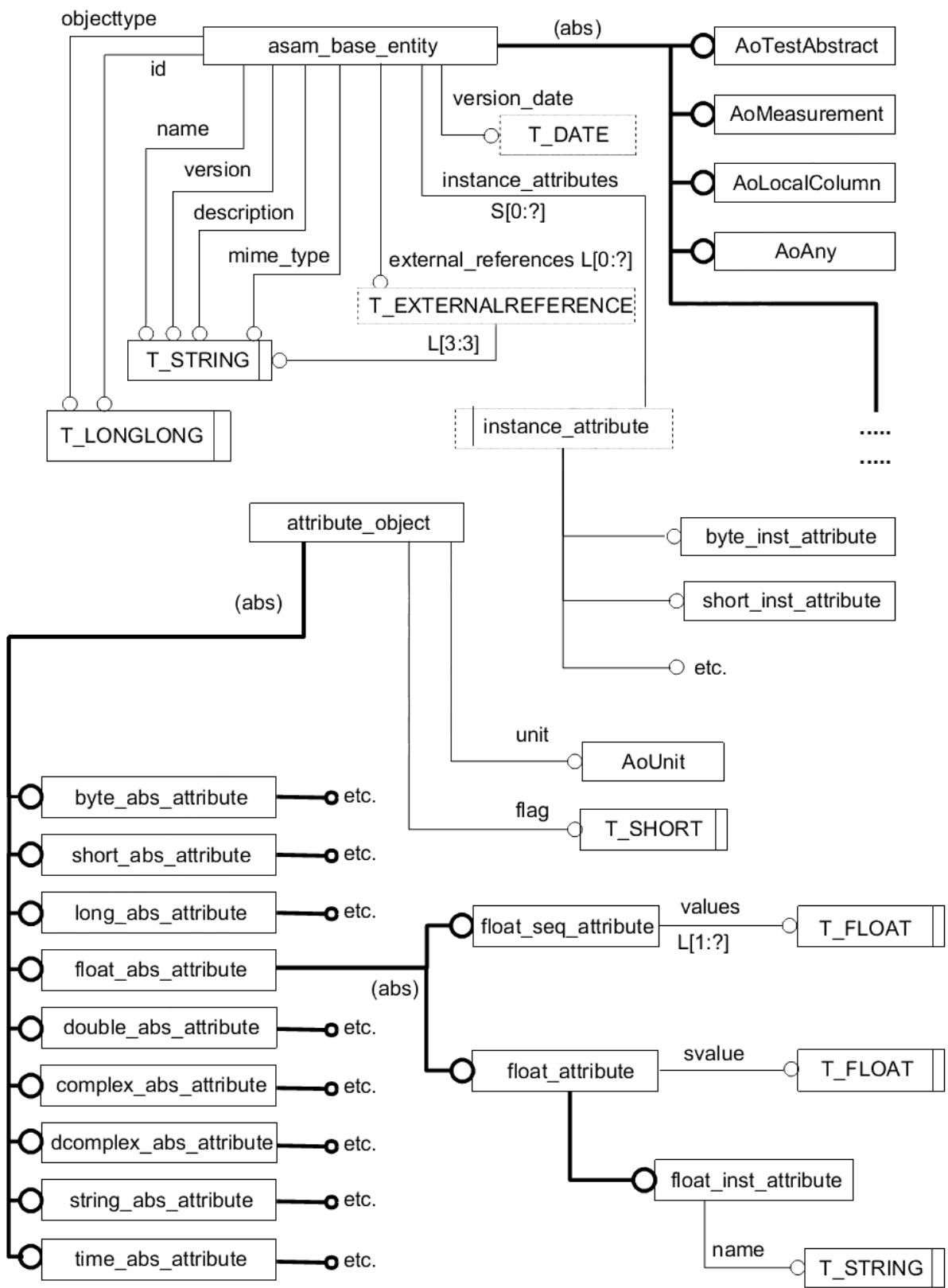


图58 EXPRESS-G 图 顶层实体和属性对象

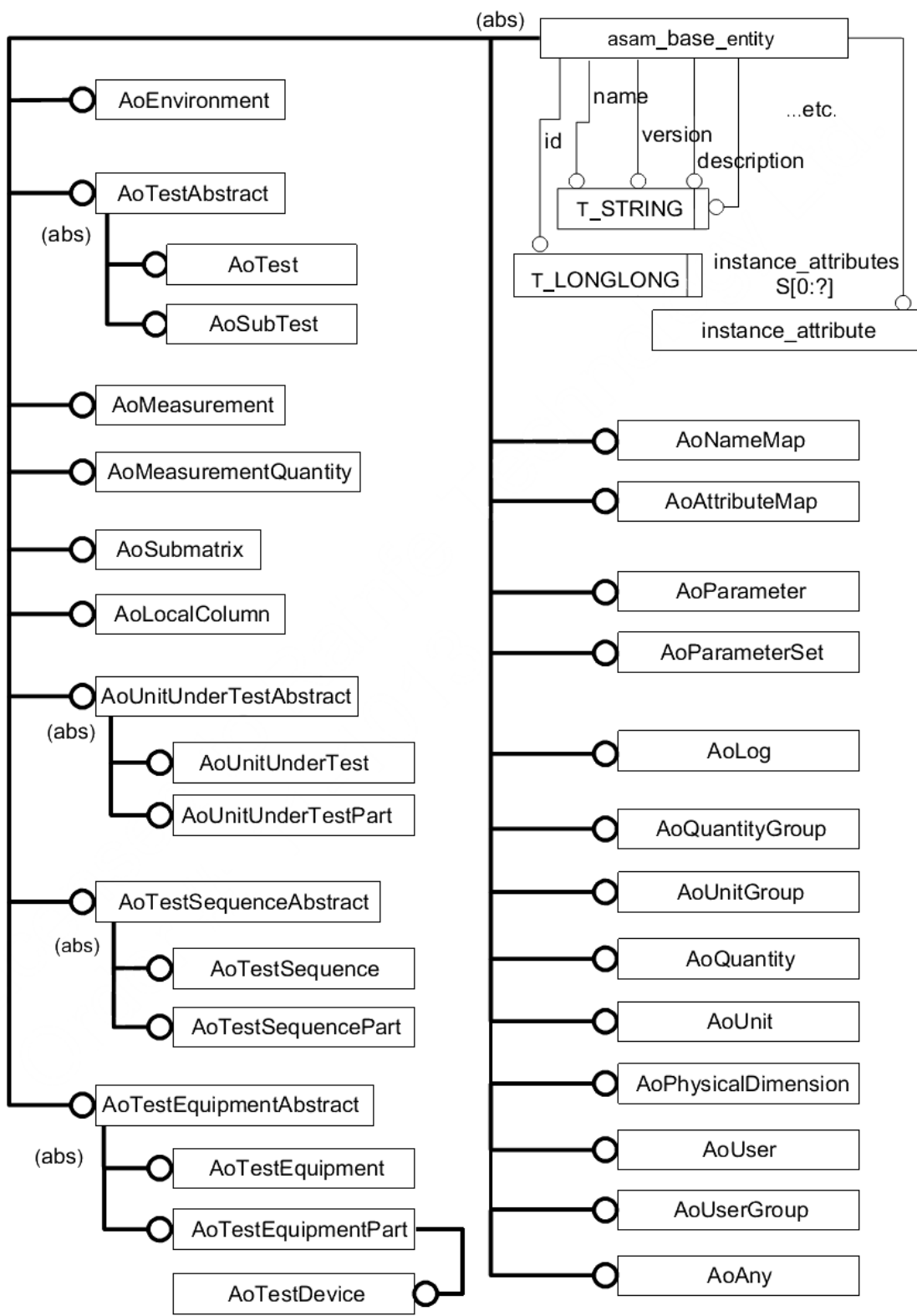


图59 EXPRESS-G 图 基本实体

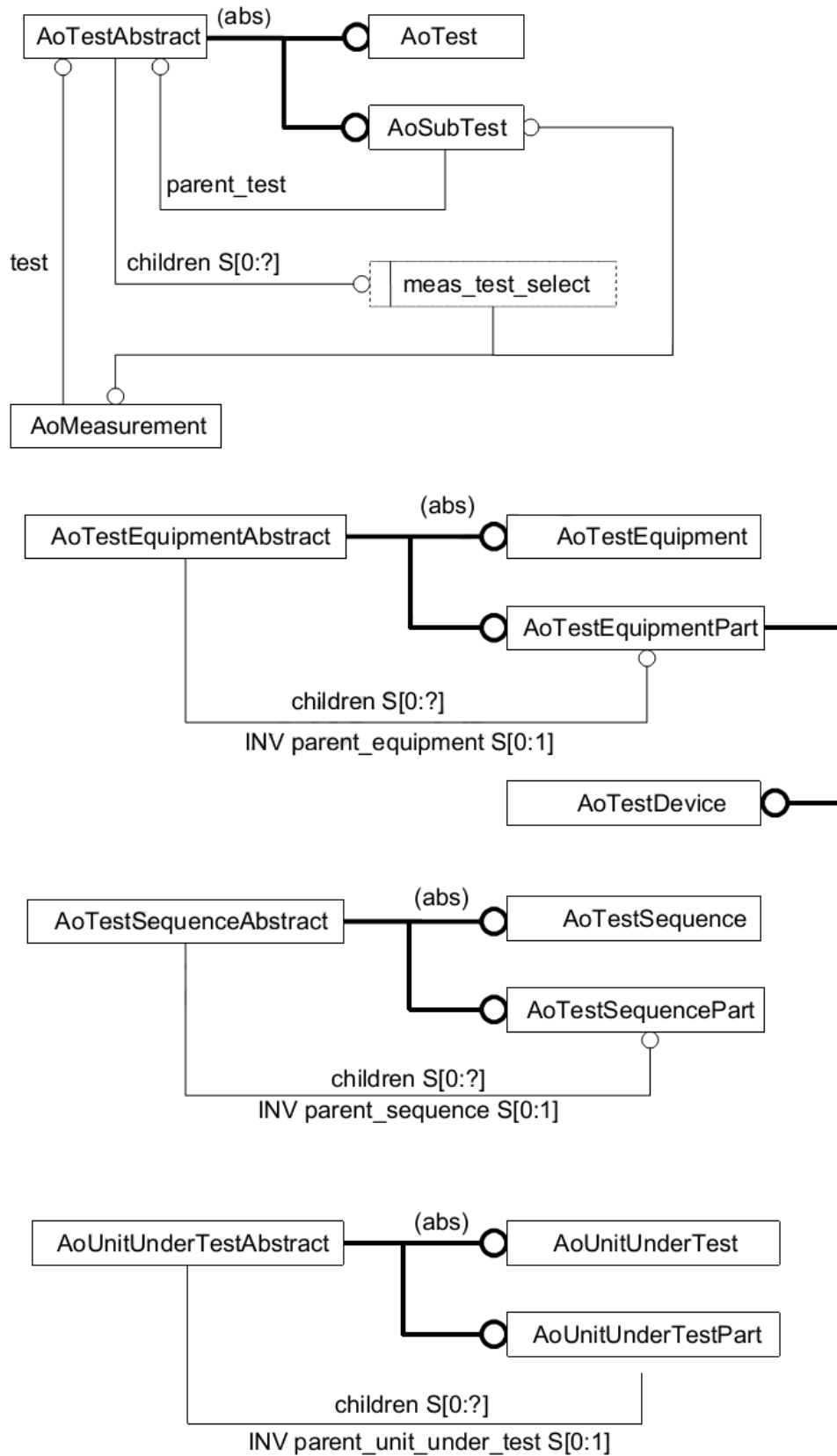


图60 EXPRESS-G 图 分层实体

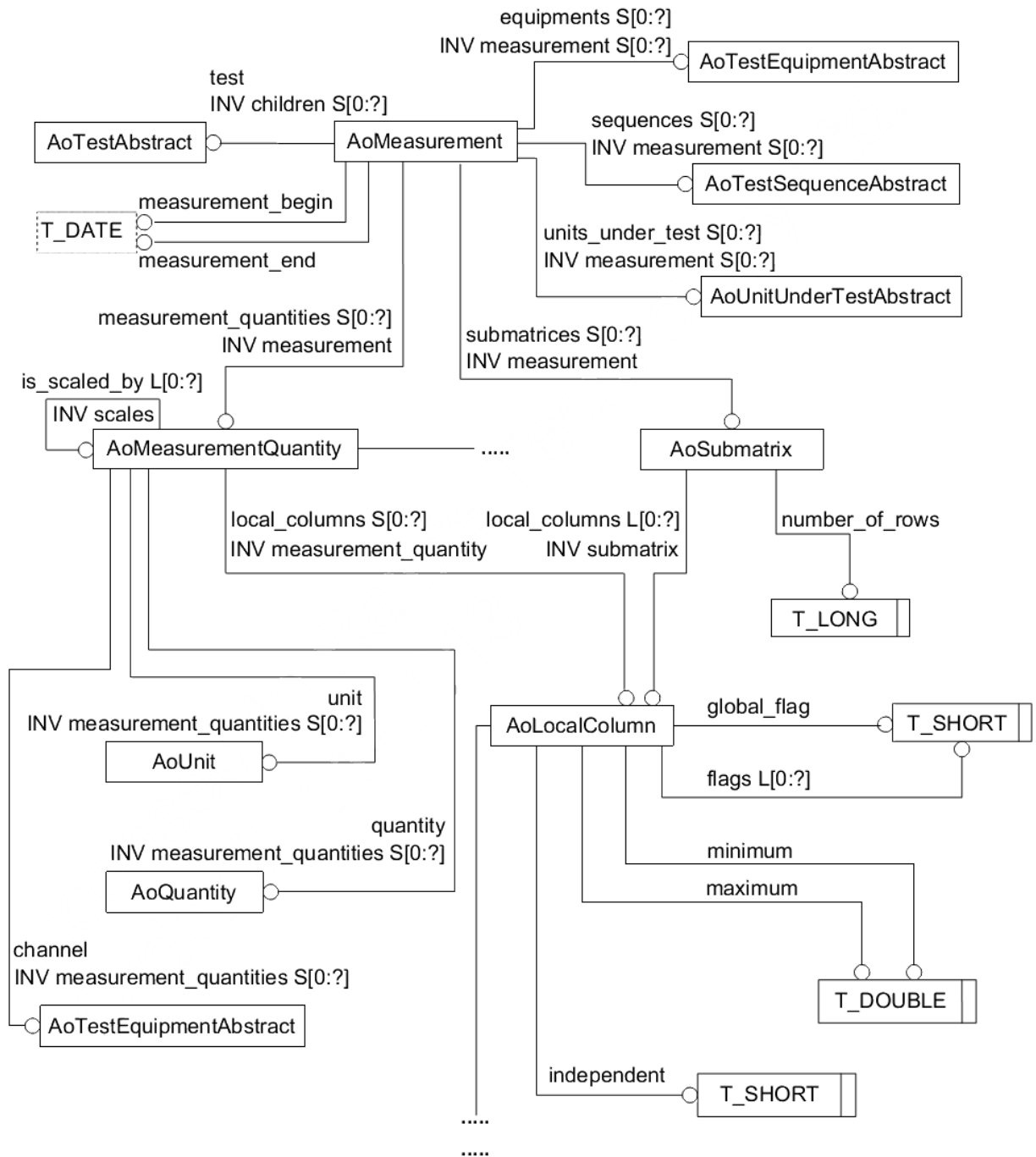


图61 EXPRESS-G 图 测量和子矩阵

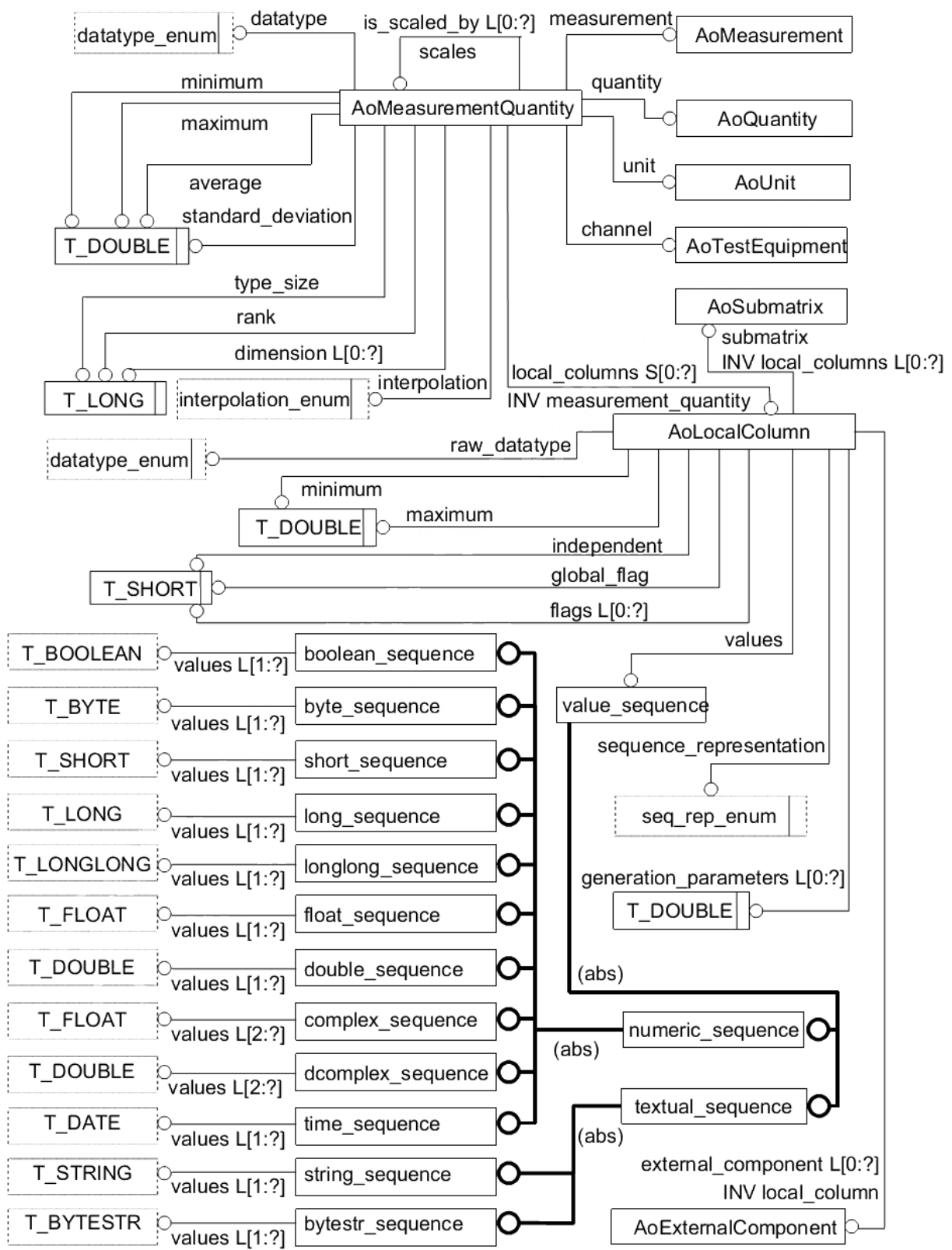


图62 EXPRESS-G 图 测量数量和局部列

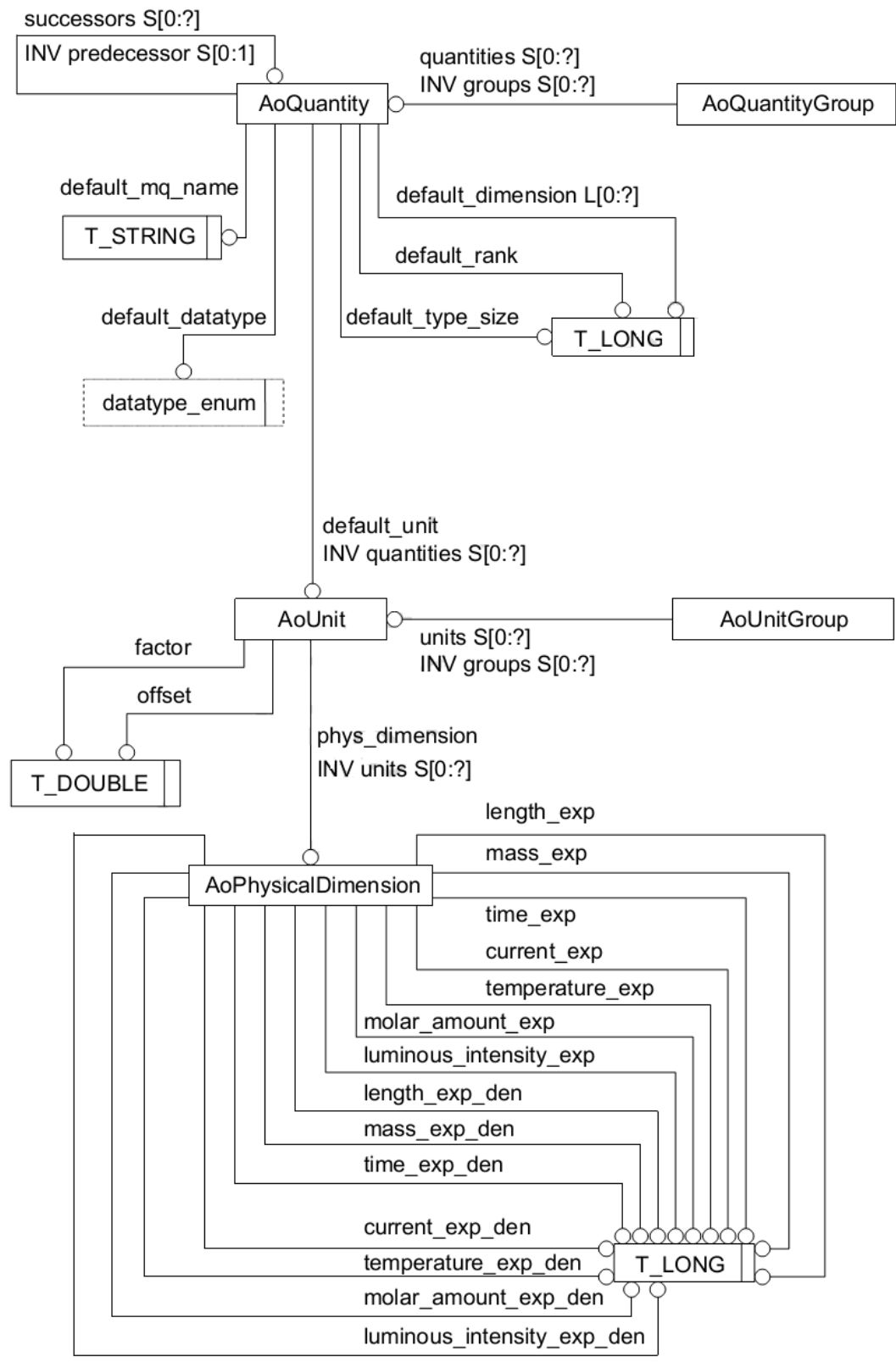


图63 EXPRESS-G 图 量和单位

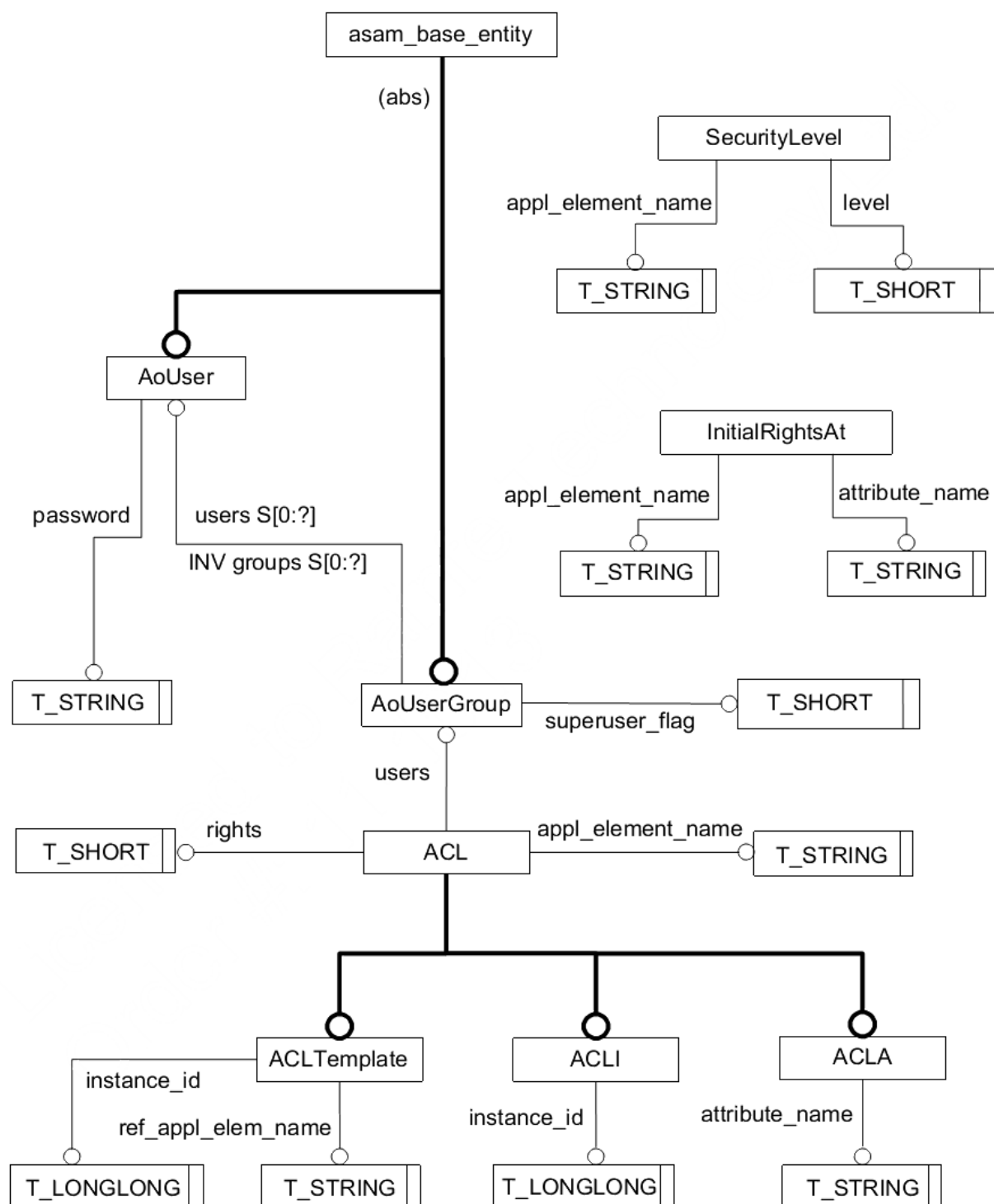


图64 EXPRESS-G 图 安全

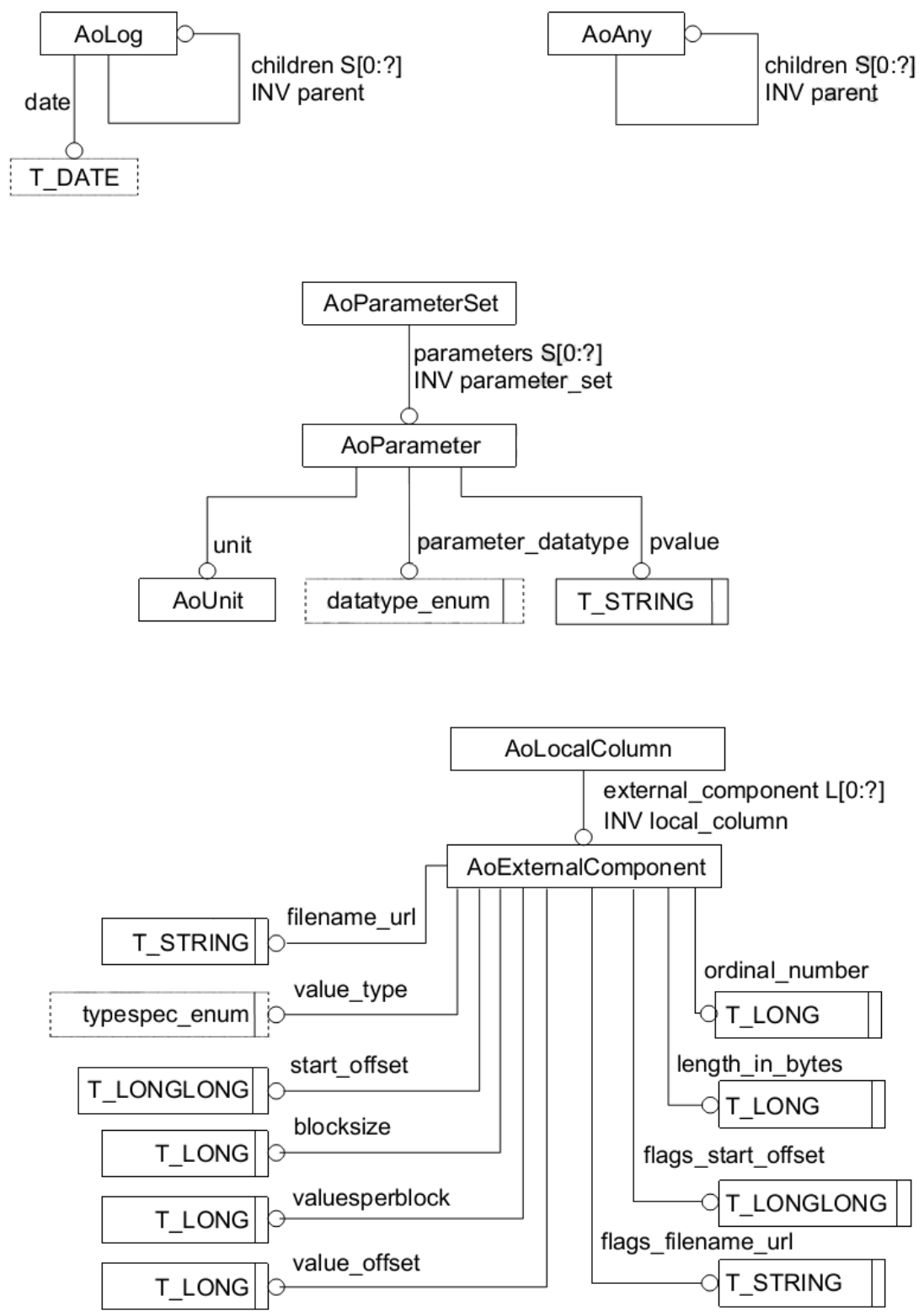


图65 EXPRESS-G 图 参数、日志和外部组件

8 标准模型

TODS通过对《ISO/PAS 22720 ASAM Open Data Services 5.0》进行扩展,提供试验数据领域开放数据交换标准。专为试验数据提供专用的、高级别的ODS标准。TODS 完全兼容《ISO/PAS 22720 ASAM Open Data Services 5.0》标准,在TODS标准中可通过《ISO/PAS 22720 ASAM Open Data Services 5.0》标准进行数据交换,同时在实现了TODS标准的系统中可通过TODS标准进行数据交换。

8.1 TODS 标准模型介绍

本节详细介绍TODS标准模型组成、TODS标准元素间的关系、通用扩展规则。

图 66 显示 TODS 的标准模型。

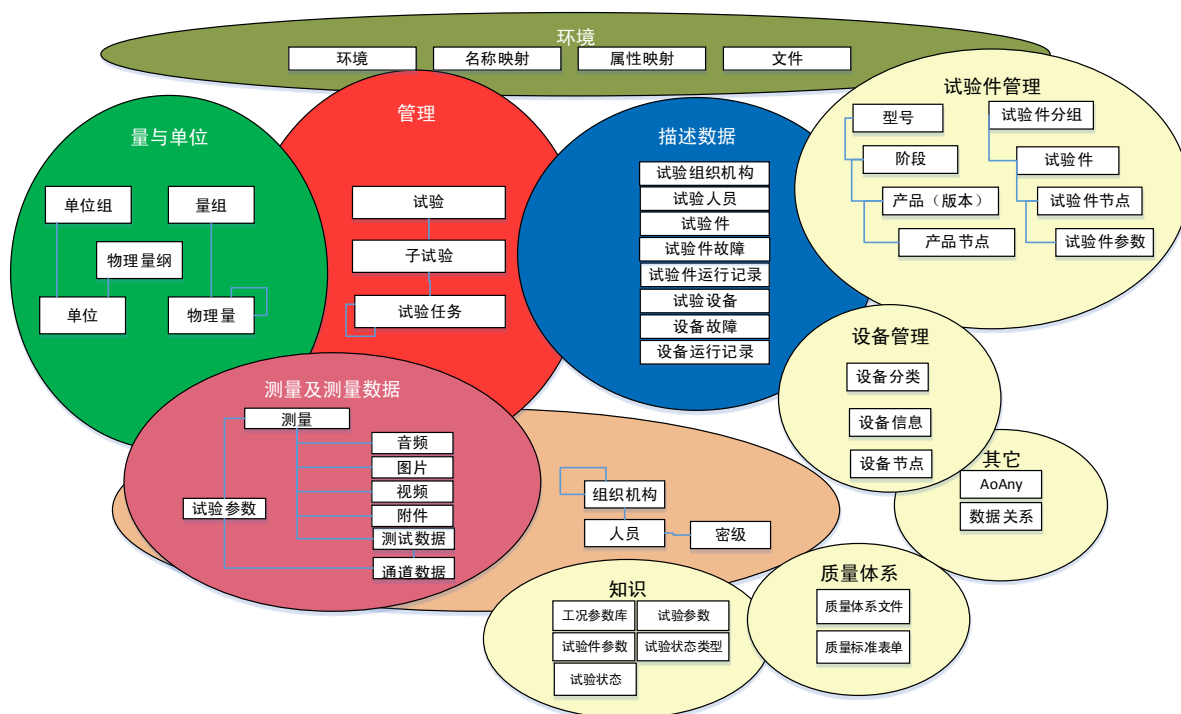


图66 TODS 标准模型图

基础模型包括以下基本元素,从这些基本元素中可以派生出应用元素。这些元素根据用法进行分组。

8.1.1 扩展规则

任何一个TODS模型都应该能够根据测试的具体情境在测量执行后纳入测量数据。因为这些数据出现在许多公司或部门,所以无法将它们映射到同一个模型上。再者,事实证明在不同的公司、部门或不同的应用之间交换数据是很有必要的。

为了方便在不同的数据库之间进行数据交换,这些数据库中的数据模型差异不能太大,即使是为了满足某些实际应用或公司的特殊要求而导致数据模型之间存在差异,这种差异也不能太大。

在 TODS 中,这种差别得到了解决。我们创建了一个标准模型,然后再从这个标准模型扩展出其它特殊用途的模型。这些扩展类型包括:

- 属性的传递（用作特殊用途的数据从基础模型中获得属性和联系）；
- 规则的传递（基础模型给特殊用途的数据提供规则，后者必须服从）。

这种扩展出来的模型通常是在某个方面的典型应用，因此被称作扩展模型。它的各个部分被称为扩展元素，而标准模型部分称为标准元素，基础模型的各个部分被称为基本元素。

8.2 环境元素的描述

环境元素见表102。

表102 环境元素

序号	标准元素	显示名	基础元素	备注
1	TdmEnvironment	环境	AoEnvironment	
2	TdmNameMap	名称映射	AoNameMap	
3	TdmAttributeMap	属性映射	AoAttributeMap	
4	TdmFile	文件	AoFile	

8.2.1 TdmEnvironment

TdmEnvironment 描述见表103。

表103 TdmEnvironment 描述

名称	显示名	物理名	基础属性/ 关系	性质	TODS 必选	长度	ODS 数据类 型	允许 空	备注
GID	主键	GID				36	DT_STRING	否	
ID	标识	ID	ID	必			DT_LONGLON G	否	
NAME	名称	NAME	NAME	必		50	DT_STRING	否	
DESCRIPTION	描述	DESCRIPTI ON	DESCRIPTI ON			200	DT_STRING		对 服 务 器 的 描 述
VERSION	版本	VERSION	VERSION			50	DT_STRING	否	服 务 器 的 版 本
VERSION_DATE	日期	VERSION_D ATE	VERSION_D ATE				DT_DATE	否	
BASE_MODEL_VERSION	基础模型版本	BASE_MODE L_VERSION	BASE_MODE L_VERSION			50	DT_STRING	否	
APPLICATION_MODEL_VERSION	应用模型版本	APPLICATI ON_MODEL_ VERSION	APPLICATI ON_MODEL_ VERSION			50	DT_STRING	否	

8.2.2 TdmNameMap

TdmNameMap 的描述见表104。

表104 TdmNameMap 的描述

名称	显示名	物理名	基础属性/ 关系	性质	TODS 必 选	长度	ODS数据 类型	允 许 空	备注
GID	主键	GID				36	DT_STRI NG	否	
ID	标识	ID	ID	必			DT_LONG LONG	否	
NAME	名称	NAME	NAME	必		50	DT_STRI NG	否	
DESCRIPTION	描述	DESCRIPT ION	DESCRIPTI ON			200	DT_STRI NG		
ENTITY_NAME	元素名称	ENTITY_N AME	ENTITY_NA ME	必		50	DT_LONG LONG	否	
ALIAS_NAME	替代名称	ALIAS_NA ME	ALIAS_NAM E	必		50	DT_STRI NG	否	

8.2.3 TdmAttributeMap

TdmAttributeMap 的描述见表105。

表105 TdmAttributeMap 的描述

名称	显示名	物理名	基 础 属 性/关系	性质	TODS 必 选	长度	ODS 数据 类型	允许空	备注
GID	主键	GID				36	DT_STRI NG	否	
ID	标识	ID	ID	必			DT_LONG LONG	否	
NAME	名称	NAME	NAME	必		50	DT_STRI NG	否	
DESCRIPTION	描述	DESCRIPTI ON	DESCRIP TION			200	DT_STRI NG		
ATTRIBUTE_NAME	属性名称	ATTRIBUTE _NAME	ATTRIBU TE_NAME	必		50	DT_STRI NG	否	
ALIAS_NAMES	属性别名	ALIAS_NAM ES	ALIAS_N AMES	必		50	DT_STRI NG	否	
元素关系									
关系名		关系类型	基 础 关 系名	性质	TODS 必 选	关联元素		范围	备注
REF_NAME_FATHER		FATHER	NAME_MA PPING	必		TDMNAM EMAP		1:1	

8.2.4 TdmFile

TdmFile 的描述见表106。

表106 TdmFile 的描述

名称	显示名	物理名	基础属性 / 关系	性质	TODS 必选	长度	ODS 数据类型	允许空	备注
GID	主键	GID				36	DT_STRING	否	
ID	标识	ID	ID	必			DT_LONGLONG	否	
NAME	名称	NAME	NAME	必		50	DT_STRING	否	
DESCRIPTION	描述	DESCRIPTION	DESCRIPTION			200	DT_STRING		
CREATED	创建时间	ORIGINAL_FILEDATE	AO_CREATED				DT_DATE		
LAST_MODIFIED	修改日期	LAST_MODIFIED	AO_LAST_MODIFIED				DT_DATE		
LOCATION	文件地址	FILEPATH	AO_LOCATION			50	DT_STRING		
SIZE	文件大小	CONTENTSIZE	AO_SIZE			64	DT_LONGLONG	否	
ORIGINAL_FILENAME	文件名称	ORIGINAL_NAME	AO_ORIGINAL_FILENAME			50	DT_STRING	否	
元素关系									
关系名		关系类型	基础关系名	性质	TODS 必选	关联元素		范围	备注
REF_FILE_FATHER		FATHER	AO_FILE_PARENT	必		TDMFILE		0:1	
REF_FILE_CHILD		CHILD	AO_FILE_CHILDREN	必		TDMFILE		0:n	

8.3 管理元素的描述

管理元素见表107。

表107 管理元素

序号	标准元素	显示名	基础元素	备注
1	TdmTest	试验	AoTest	
2	TdmSubTest	子试验	AoSubTest	
3	TdmTestTask	试验任务	AoAny	

8.3.1 TdmTest

TdmTest 的描述见表108。

表108 TdmTest 的描述

名称	显示名	物理名	基础属性/关系	性质	TODS 必选	长度	ODS 数据类型	允 许 空	备注
GID	主键	ID				36	DT_STRING	否	
ID	标识	SEQUENCE	ID	必			DT_LONGLONG	否	
NAME	名称	NAME	NAME	必		50	DT_STRING	否	
DESCRIPTION	描述	DESCRIPT ION	DESCRIPTION			200	DT_STRING		
CREATED	创 建 时 间	CREATEDA TE	CREATED				DT_DATE		
CREATED_BY	创建人	CREATED_ BY	CREATED_BY			50	DT_STRING		
CODE	试 验 编 号	CODE					DT_STRING		
PLANSTARTTIME	计 划 开 始时间	PLANSTAR TTIME					DT_DATE		
PLANENDTIME	计 划 结 束时间	PLANENDT IME					DT_DATE		
REALSTARTTIME	实 际 开 始时间	REALSTAR TTIME					DT_DATE		
REALENDTIME	实 际 结 束时间	REALENDT IME					DT_DATE		
STATE	状态	STATE					DT_LONG		0: 未 开 时, 1: 开 始,

名称	显示名	物理名	基础属性/关系	性质	TODS 必选	长度	ODS 数据类型	允 许 空	备注
									2: 暂停, 3: 结束
元素关系									
关系名	关系类型	基础关系名	性质	TODS 必 选	关联 元素		范围		备注
REL_TEST_SUBTEST_CHILD	CHILD	CHILDREN	必		TdmSubTest or TdmMeasurement		0:n		
REL_TEST_MEASUREMENT_CHILD									

8.3.2 TdmSubTest

TdmSubTest的描述见表109。

表109 TdmSubTest 的描述

名称	显示名	物理名	基础属性/关系	性质	TODS 必选	长度	ODS 数据类型	允 许 空	备注
GID	主键	ID				36	DT_STRING	否	
ID	标识	SEQUENCE	ID	必			DT_LONGLONG	否	
NAME	名称	NAME	NAME	必		50	DT_STRING	否	
DESCRIPTION	描述	DESCRIPTION	DESCRIPTION			200	DT_STRING		
CREATED	创建时间	CREATEDATE	CREATED				DT_DATE		
CREATED_BY	创建人	CREATED_BY	CREATED_BY			50	DT_STRING		
CODE	试验编号	CODE					DT_STRING		
PLANSTARTTIME	计划开	PLANST					DT_DATE		

名称	显示名	物理名	基础属性/ 关系	性质	TODS 必选	长度	ODS 数据 类型	允 许 空	备注
	始时间	ARTTIME							
PLANENDTIME	计 划 结 束时间	PLANENDTIME					DT_DATE		
REALSTARTTIME	实 际 开 始时间	REALSTARTTIME					DT_DATE		
REALENDTIME	实 际 结 束时间	REALENDTIME					DT_DATE		
STATE	状态	STATE					DT_LONG		0: 未开时, 1: 开始, 2: 暂停, 3: 结 束 4:强制结束
元素关系									
关系名	关 系 类 型	基础关系名	性质	TODS 必选	关联元素		范围	备注	
REL_TEST_FATHER	FATHER	PARENT_TEST	必		TdmTest		1:1		
REL_SUBTEST_FATHER					or TdmSubTest				
REL_SUBTEST_CHILD	CHILD	CHILDREN	必		TdmSubTest		0:n		
REL_SUBTEST_MEASUREMENT_CHILD					or TdmMeasurement				

8.3.3 TdmTestTask

TdmTestTask的描述见表110。

表110 TdmTestTask 的描述

名称	显示名	物理名	基础属性 / 关系	性质	TODS 必选	长度	ODS 数据类型	允许空	备注
GID	主键	ID				36	DT_STRING	否	
ID	标识	NID	ID	必			DT_LONG LONG	否	
NAME	名称	NAME	NAME	必		50	DT_STRING	否	
DESCRIPTION	描述	NOTE	DESCRIPTION			200	DT_STRING		
CREATED	创建时间	CREATEDATE	CREATED				DT_DATE		
CREATED_BY	创建人	CREATED_BY	CREATED_BY			50	DT_STRING		
CODE	试验编号	CODE				50	DT_STRING		
PLANSTARTTIME	计划开始时间	PLANSTARTTIME					DT_DATE		
PLANENDTIME	计划结束时间	PLANENDTIME					DT_DATE		
REALSTARTTIME	实际开始时间	REALSTARTTIME					DT_DATE		
REALENDTIME	实际结束时间	REALENDTIME					DT_DATE		
STATE	状态	STATE					DT_LONG		0：未开时，1：开始，2：暂停，3：结束
OUTLINELEVEL	层级	OUTLINELEVEL					DT_LONG		
OUTLINENUMBER	大纲号	OUTLINENUMBER					DT_STRING		
LIMITTYPE	限制类型	LIMITTYPE					DT_STRING		
COMPLETEPERCENT	完成百分比	COMPLETEPERCENT					DT_LONG		

		RCENT							
SERIALNO	序号	SERIALNO					DT_LONG		
FIRSTLEVEL	优先级	FIRSTLEVEL					DT_LONG		
SUMMARY	任务摘要	SUMMARY					DT_LONG		
LIMITTIME	限制时限	LIMITTIME					DT_STRING		
TIMELIMIT	工期（天）	TIMELIMIT					DT_LONG		
INTERVALTIME	时间间隔	INTERVALTIME					DT_LONG		
ISMARKER	是否里程碑	ISMARKER					DT_LONG		0: 否 1: 是
元素关系									
关系名	关系类型	基础关系名	性质	TOD S 必 选	关联元素		范围	备注	
REL_TEST_FATHER	FATHER			必	TdmTest or TdmSubTest	PROJEC TID	1:1	所 属 试 验、子试 验	
REL_TESTTASK_FATHER	FATHER				TdmTestTask		0:1		
REL_TESTTASK_CHILD	CHILD				TdmTestTask	PARENT ID	0:n		
REL_TABOBJECT_CHILD	CHILD				TdmTableObject		0:n		
REL_WEBOFFICEFILE_CHILD	CHILD				TdmWebOfficeFile		0:n		

REL_VEDIO_CHILD	CHILD				TdmVedio		0:n	
REL_PICTURE_CHILD	CHILD				TdmPicture		0:n	
REL_AFFIX_CHILD	CHILD				TdmAffix		0:n	
REL_SUBMATRIX	CHILD				TdmSubmatrix		0:n	

8.4 测量及测量数据元素的描述

测量及测量数据元素见表111。

表111 测量及测量数据元素的函数

序号	标准元素	显示名	基础元素	备注
1	TdmMeasurementQuantity	测量参数	AoMeasurementQuantity	
2	TdmMeasurement	测量	AoMeasurement	
3	TdmVideoFile	视频文件	AoAny	
4	TdmPictureFile	图片文件	AoAny	
5	TdmAffixFile	附件文件	AoAny	
6	TdmSubmatrix	测量数据	AoSubmatrix	
7	TdmStructLocalColumn	测量行列化列数据	AoLocalColumn	

8.4.1 TdmMeasurementQuantity

TdmMeasurementQuantity 的描述见表112。

表112 TdmMeasurementQuantity 的描述

名称	显示名	物理名	基础属性/关系	性质	TODS必选	长度	ODS数据类型	允许空	备注
GID	主键	GID				36	DT_STRING	否	
ID	标识	ID	ID	必			DT_LONGLONG	否	
NAME	名称	NAME	NAME	必		50	DT_STRING	否	
DESCRIPTION	描述	DESCRI	DESCRIPTI			200	DT_STRING		

名称	显示名	物理名	基础属性/ 关系	性质	TODS 必选	长度	ODS 数据 类型	允许 空	备注
		PTION	ON						
CREATED	创 建 时 间	CREATE DATE	CREATED				DT_DATE		
CREATED_BY	创建人	CREATE D_BY	CREATED_B Y			50	DT_STRING		
DATATYPE	数 据 类 型	DATATY PE	DATATYPE	必			DT_ENUM	否	
SYMBOL	参数符 号					200	DT_STRING	true	
元素关系									
关系名	关系类型	基础关系名	性 质	TODS 必选	关联元素		范围	备注	
MEASUREMENT_FAT HER	FATHER	MEASUREMENT	必		TdmMeasurem ent	ACTIONKEY	0:1		
UNIT	INFO_TO	UNIT	必		TdmUnit		0:1		
QUANTITY	INFO_TO	QUANTITY	必		TdmQuantity		0:1		
LOCAL_COLUMNS Or DATA_LOCAL_COLU MNS Or STRUCT_LOCAL_CO LUMNS	INFO_REL	LOCAL_COLUMNS	必		TdmLocalCom nu Or TdmDataLoca lColumn Or TdmStructLo calColumn		0:n		
REL_PARAMTYPE	INFO_REL				TdmParamTyp e	TYPEID	0:1	参 数 类 型 (TBL_PARA MTYPE)	

8.4.2 TdmMeasurement

TdmMeasurement的描述见表113。

表113 TdmMeasurement 的描述

名称	显示名	物理名	基础属性/ 关系	性质	TODS 必选	长度	ODS 数据 类型	允许 空	备注
GID	主键	GID				36	DT_STRIN G	否	
ID	标识	ID	ID	必			DT_LON GLONG	否	
NAME	名称	NAME	NAME	必		50	DT_STRIN G	否	

名称	显示名	物理名	基础属性/ 关系	性质	TODS 必选	长度	ODS 数据 类型	允许 空	备注
							NG		
DESCRIPTION	描述	DESC RIPTI ON	DESCRIPT ION			200	DT_STRI NG		
CREATED	创 建 时 间	CREA TEDA TE	CREATED				DT_DATE		
CREATED_BY	创建人	CREA TED_ BY	CREATED _BY			50	DT_STRI NG		
MEASUREMENT_BEG IN	实 际 开 始时间	REALST ARTTIM E	MEASUREME NT_BEGIN				DT_DATE		
MEASUREMENT_END	实 际 结 束时间	REALEN DTIME	MEASUREME NT_END				DT_DATE		
元素关系									
关系名	关系类型	基础关 系名	性质	TODS 必选	关联元素		范围	备注	
TEST_FATHER Or SUBTEST_FATHER	FATHER	TEST	必		TdmTest or TdmSubT est		1:1		
MEASUREMENT_QUANTIT IES_ CHILDREN	CHILD	MEASURE MENT_QU ANTITIE S	必		TdmMeas ureMent Quantit y		0:n		
SUBMATRICES_CHILDRE N	CHILD	SUBMATR ICES	必		TdmSubm atrix		0:n		
EQUIPMENTS	INFO_REL	EQUIPME NTS			TdmTest Equipme nt		0:n		
UNITS_UNDER_TESTS	INFO_REL	UNITS_U NDER_TE ST			TdmUnit UnderTe st		0:n		
TABLEOBJECT_CHILDRE N	CHILD				TdmTabl eObject		0:n		
WEBOFFICEFILE_CHILD REN	CHILD				TdmWebO fficeFi le		0:n		
AFFIXFILE_CHILDREN	CHILD				TdmAffi		0:n		

名称	显示名	物理名	基础属性/ 关系	性质	TODS 必选	长度	ODS 数据 类型	允许 空	备注
						x			
PICTUREFILE_CHILDREN	CHILD					TdmPicture		0:n	
VEDIOFILE_CHILDREN	CHILD					TdmVideo		0:n	

8.4.3 TdmVedioFile

TdmVedioFile的描述见表114。

表114 TdmVedioFile 的描述

名称	显示名	物理名	基础属性/ 关系	性质	TODS 必选	长度	ODS 数据 类型	允许 空	备注
GID	主键	ID				36	DT_STRING	否	
ID	标识	SEQUENCE	ID	必			DT_LONGLONG	否	
NAME	名称	DATAFILENAME	NAME	必		50	DT_STRING	否	
DESCRIPTION	描述	DESCRIPTION	DESCRIPTION			200	DT_STRING		
CREATED	创建时间	SUBMITTEDATE	CREATED				DT_DATE		
CREATED_BY	创建人	SUBMITPERSON	CREATED_BY			50	DT_STRING		
元素关系									
关系名	关系类型	基础关系名	性质	TODS 必选	关联元素		范围	备注	
MEASUREMENT_FATHER Or TEST_FATHER Or SUBTEST_FATHER Or TESTTASK_FATHER	FATHER				TdmMeasurement or TdmTest or TdmSubTest Or TdmTestTask	PROJID	1:1		
FILE_CHILDREN	CHILD	DATAFILE			TdmFile	DATAFILE	0:n	文件	

8.4.4 TdmPictureFile

TdmPictureFile 的描述见表115。

表115 TdmPictureFile 的描述

名称	显示名	物理名	基础属性/ 关系	性质	TODS 必选	长度	ODS 数据 类型	允许 空	备注
GID	主键	ID				36	DT_STRIN G	否	
ID	标识	SEQU ENCE	ID	必			DT_LON GLONG	否	
NAME	名称	DATA FILEN AME	NAME	必		50	DT_STRIN G	否	
DESCRIPTION	描述	DESC RIPTI ON	DESCRIP TION			200	DT_STRIN G		
CREATED	创建时 间	SUBM ITEDA TE	CREATED				DT_DATE		
CREATED_BY	创建人	SUBM ITPER SON	CREATED _BY			50	DT_STRIN G		
元素关系									
关系名	关系类型	基础关 系名	性质	TODS 必选	关联元素		范围	备注	
MEASUREMENT_FATHER Or TEST_FATHER Or SUBTEST_FATHER Or TESTTASK_FATHER	FATHER				TdmMeas urement or TdmTest or TdmSubT est Or TdmTest Task	PROJID	1:1	所属试验、 所属子试 验、所属测 量	
FILE_CHILDREN	CHILD	DATAFIL E			TdmFile	DATAFIL E	0:n	文件	

8.4.5 TdmAffixFile

TdmAffixFile的描述见表116。

表116 TdmAffixFile 的描述

名称	显示名	物理名	基础属性/ 关系	性质	TODS 必选	长度	ODS 数据 类型	允许 空	备注
GID	主键	ID				36	DT_STR ING	否	
ID	标识	SEQU ENCE	ID	必			DT_LON GLONG	否	
NAME	名称	DATA FILEN AME	NAME	必		50	DT_STR ING	否	
DESCRIPTION	描述	DESC RIPTI ON	DESCRIPT ION			200	DT_STR ING		
CREATED	创建时 间	SUBM ITEDA TE	CREATED				DT_DATE		
CREATED_BY	创建人	SUBM ITPER SON	CREATED _BY			50	DT_STR ING		
元素关系									
关系名	关系类型	基础关 系名	性质	TODS 必选	关联元素		范围	备注	
MEASUREMENT_FATHER Or TEST_FATHER Or SUBTEST_FATHER Or TESTTASK_FATHER	FATHER				TdmMeas urement or TdmTest or TdmSubT est Or TdmTest Task	PROJID	1:1	所属试验、 所属子试 验、所属测 量、所属任 务	
FILE_CHILDREN	CHILD	DATAFIL E			TdmFile	DATAFIL E	0:n	文件	

8.4.6 TdmSubmatrix

TdmSubmatrix的描述见表117。

表117 TdmSubmatrix 的描述

名称	显示名	物理名	基础属性/ 关系	性质	TODS 必选	长度	ODS 数据 类型	允许 空	备注
GID	主键	GID				36	DT_STRI NG	否	
ID	标识	ID	ID	必			DT_LON GLONG	否	
NAME	名称	DATA FILEN AME	NAME	必		50	DT_STRI NG	否	
DESCRIPTION	描述	DESC RIPTI ON	DESCRIPT ION			200	DT_STRI NG		
CREATED	创建时 间	CREA TED	CREATED				DT_DATE		
CREATED_BY	创建人	CREA TED_ BY	CREATED _BY			50	DT_STRI NG		
NUMBER_OF_R OWS	测量点 数目	NUMB ER_O F_RO WS	NUMBER _OF_ROW S	必			DT_LONG	否	
元素关系									
关系名	关系类型	基础关 系名	性质	TODS 必选	关联元素		范围	备注	
MEASUREMENT_FATHER Or TEST_FATHER Or SUBTEST_FATHER Or TESTTASK_FTATHER	FATHER	MEASURE MENT	必		TdmMeas urement Or TdmTest Or TdmSubT est Or TdmTest Task	PROJID	1:1	所属测量	
LOCAL_COLUMNS	CHILD	LOCAL_C OLUMNS	必		TdmData LocalCo lumn		0:n		
DATAFILESTRUBMATRIX	SUBTYPE				TdmData FileSub matrix		0:1		
STRUCTSUBMATRIX	SUBTYPE				TdmStru		0:1		

名称	显示名	物理名	基础属性/ 关系	性质	TODS 必选	长度	ODS 数据 类型	允许 空	备注
						ctSubma trix			

8.4.7 TdmLocalColumn

TdmLocalColumn 的描述见表118。

表118 TdmLocalColumn 的描述

名称	显示名	物理名	基础属性/ 关系	性质	TODS 必选	长度	ODS 数据 类型	允许 空	备注
GID	主键	GID				36	DT_STRIN G	否	
ID	标识	ID	ID	必			DT_LON GLONG	否	
NAME	名称	NAME	NAME	必		50	DT_STRIN G	否	
DESCRIPTION	描述	DESC RIPTI ON	DESCRIPT ION			200	DT_STRIN G		
CREATED	创建时 间	SUBM ITDAT E	CREATED				DT_DATE		
CREATED_BY	创建人	CREA TED_ BY	CREATED _BY			50	DT_STRIN G		
GLOBAL_FLAG	总体标 记	GLOB AL_FL AG	GLOBAL_ FLAG				DT_SHOR T	否	
元素关系									
关系名	关系类型	基础关 系名	性质	TODS 必选	关联元素		范围	备注	
SUBMATRIX_FATHER	FATHER	SUBMA TRIX	必		TdmSub matrix		1:1		
MEASUREMENT_QUANTITY	INFO_TO	MEASU REMEN T_QUA NTITY	必		TdmMe asureme ntQuanti ty		1:1		

8.5 描述数据的描述

描述数据的元素见表119。

表119 描述数据的元素

序号	标准元素	显示名	基础元素	备注
1	TdmTestUserGroup	试验组织机构	AoUserGroup	
2	TdmTestUser	试验人员	AoUser	
3	TdmUnitUnderTest	试验件	AoUnitUnderTest	
4	TdmUUTTrouble	试验件故障	AoAny	
5	TdmUUTRecord	试验件运行记录	AoAny	
6	TdmTestEquipment	试验设备	AoTestEquipment	
7	TdmEquipmentTrouble	设备故障	AoAny	
8	TdmEquipmentRecord	设备运行记录	AoAny	

8.5.1 TdmTestUserGroup

TdmTestUserGroup 的描述见表120。

表120 TdmTestUserGroup 的描述

名称	显示名	物理名	基础属性 /关系	性质	TODS 必选	长度	ODS 数 据类型	允许 空	备注
GID	主键	ID				36	DT_STR ING	否	
ID	标识	SEQUENCE	ID	必			DT_LON GLONG	否	
NAME	名称	ORGNAME	NAME	必		50	DT_STR ING	否	
DESCRIPTION	描述	DESCRIPTI ON	DESCRIPT ION			200	DT_STR ING		
CREATED	创建时间	CREATED ATE	AO_CRE ATED				DT_DA TE		
CREATED_BY	创建人	CREATED _BY	AO_CRE ATED_B Y			50	DT_ST RING		
元素关系									
关系名		关系类型	基础关 系名	性质	TODS 必选	关联元素		范围	备注
REF_USERGROUP_FATHER		FATHER				TdmTe stUse rGrou p		0:1	

REF_USERGROUP_CHILD	CHILD				TdmTestUserGroup	PARENTORGID	0:n	父节点标识
REF_TESTUSER_CHILD	CHILD				TdmTestUser		0:n	

8.5.2 TdmTestUser

TdmTestUser 的描述见表121。

表121 TdmTestUser 的描述

名称	显示名	物理名	基础属性/关系	性质	TODS 必选	长度	ODS 数据类型	允许空	备注
GID	主键	ID				36	DT_STRING	否	
ID	标识	SEQUENCE	ID	必			DT_LONGLONG	否	
NAME	名称	NAME	NAME	必		50	DT_STRING	否	
DESCRIPTION	描述	NOTE	DESCRIPTION			200	DT_STRING		
CREATED	创建时间	CREATE DATE	AO_CREATED				DT_DATE		
CREATED_BY	创建人	CREATE D_BY	AO_CREATED_BY			50	DT_STRING		
元素关系									
关系名	关系类型	基础关系名	性质	TODS 必选	关联元素		范围	备注	
REL_TESTUSERGROUP_FATHER	FATHER				TdmTestUserGroup	ORGID	1:1	所属组织结构	

8.5.3 TdmUnitUnderTest

TdmUnitUnderTest 的描述见表122。

表122 TdmUnitUnderTest 的描述

序号	名称	物理表名称	显示名	基础模型	备注
1	TdmUnitUnderTest	TDMUNITUNDERTEST	试验件	AoUnitUnderTest	包含 一个 TdmUUTObj 拷贝，并引用一个 Obj
2	TdmUUTTrouble	TDMUUTTROUBLE	试验件故障	AoAny	
3	TdmUUTRecord	TDMUUTRECORD	试验件运行记录	AoAny	

8.5.4 TdmUUTTrouble

TdmUUTTrouble 的描述见表 123。

表123 TdmUUTTrouble 的描述

名称	显示名	物理名	基础属性/关系	性质	TODS 必选	长度	ODS 数据类型	允许空	备注
GID	主键	GID				36	DT_STRING	否	
ID	标识	ID	ID	必			DT_LONGLONG	否	
NAME	名称	NAME	NAME	必		50	DT_STRING	否	
DESCRIPTION	描述	DESCRIPTION	DESCRIPTION			200	DT_STRING		
CREATED	创建时间	CREATED	AO_CREATED				DT_DATE		
CREATED_BY	创建人	CREATED_BY	AO_CREATED_BY			50	DT_STRING		
元素关系									
关系名	关系类型	基础关系名	性质	TODS 必选	关联元素		范围	备注	
REL_UUTUNDERTEST_FATHER	FATHER				TdmUnitUnderTest	UUTID	1:1	所属试验件	

8.5.5 TdmUUTRecord

TdmUUTRecord 的描述见表124。

表124 TdmUUTRecord 的描述

名称	显示名	物理名	基础属性/ 关系	性质	TODS 必选	长度	ODS 数据 类型	允 许 空	备注
GID	主键	GID				36	DT_STRIN G	否	
ID	标识	ID	ID	必			DT_LON GLONG	否	
NAME	名称	RSNA ME	NAME	必		50	DT_STRIN G	否	
DESCRIPTION	描述	DESC RIPTI ON	DESCRIPT ION			200	DT_STRIN G		
CREATED	创 建 时 间	CREA TEDA TE	AO_CREA TED				DT_DATE		
CREATED_BY	创建人	CREA TED_ BY	AO_CREA TED_BY			50	DT_STRIN G		
元素关系									
关系名	关 系 类 型	基础关系名	性质	TODS 必选	关联元素		范围	备注	
REL_UUTUNDERTES T_FATHER	FATHER				TdmUn itUnd erTes t	UUTID	1:1	所属试验件	

8.5.6 TdmTestEquipment

TdmTestEquipment的描述见表125。

表125 TdmTestEquipment 的描述

名称	显示名	物理名	基础属性/ 关系	性质	TODS 必选	长度	ODS 数据 类型	允 许 空	备注
GID	主键	ID				36	DT_STRIN G	否	
ID	标识	SEQUEN CE	ID	必			DT_LONGL ONG	否	
NAME	名称	RSNAME	NAME	必		50	DT_STRIN G	否	
DESCRIPTION	描述	DESCRI PTION	DESCRIPT ION			200	DT_STRIN G		
VERSION	版本	VERSIO N	VERSION			50	DT_STRIN G		
VERSION_DATE	日期	CREATE	VERSION_				DT_DATE		

名称	显示名	物理名	基础属性 /关系	性质	TODS 必选	长度	ODS 数据 类型	允 许 空	备注
		DATE	DATE						
MIME_TYPE	多用途互 联网邮件 扩展类型	MIME_T YPE	MIME_TYP E			50	DT_STRIN G		
OBJECTTYPE	元素标识	OBJECT TYPE	OBJECTTY PE				DT_LONGL ONG		
CREATED	创建时间	CREATE TIME	AO_CREAT ED				DT_DATE		
CREATED_BY	创建人	CREATE D_BY	AO_CREAT ED_BY			50	DT_STRIN G		
LAST_MODIFIED	修改日期	LAST_M ODIFIE D	AO_LAST_ MODIFIED				DT_DATE		
LAST_MODIFIED_ BY	修改人	LAST_M ODIFIE D_BY	AO_LAST_ MODIFIED _BY			50	DT_STRIN G		
RSCODE	资源编号	RSCODE				50	DT_STRIN G	否	
BAR_CODE	条码编号	BAR_CO DE				100	DT_STRIN G		
元素关系									
关系名	关系类型	基础关系名	性质	TODS 必选	关联元素		范围	备注	
REL_ENVIRONMEN T_FATHER	FATHER	ENVIRONMENT			TdmEnvironment		0:1		
REL_MEASUREMEN T	INFO_REL	MEASUREMENT			TdmMeasurement		0:n		
REL_MEASUREMEN T_QUANTITIES	INFO_FROM	MEASUREMENT_ QUANTITIES			TdmMeasurementQ uantity		0:n		
REL_EQUIPMENTT ROUBLE_CHILD	CHILD				TdmEqui pmentTr ouble		0:n	设备故障	
REL_EQUIPMENTR ECORD_CHILD	CHILD				TdmEqui pmentRe cord		0:n	设备运行记 录	

8.5.7 TdmEquipmentTrouble

TdmEquipmentTrouble 的描述见表126。

表126 TdmEquipmentTrouble 的描述

名称	显示名	物理名	基础属性/ 关系	性质	TODS 必选	长度	ODS 数据 类型	允 许 空	备注
GID	主键	GID				36	DT_STRIN G	否	
ID	标识	ID	ID	必			DT_LONGL ONG	否	
NAME	名称	NAME	NAME	必		50	DT_STRIN G	否	
DESCRIPTION	描述	DESCRI PTION	DESCRIPTI ON			200	DT_STRIN G		
VERSION	版本	VERSIO N	VERSION			50	DT_STRIN G		
VERSION_DATE	日期	CREATE DATE	VERSION_D ATE				DT_DATE		
MIME_TYPE	多用途 互联网 邮件扩 展类型	MIME_T YPE	MIME_TYPE			50	DT_STRIN G		
OBJECTTYPE	元素标 识	OBJECT TYPE	OBJECTTYP E				DT_LONGL ONG		
CREATED	创建时 间	CREATE D	AO_CREATE D				DT_DATE		
CREATED_BY	创建人	CREATE D_BY	AO_CREATE D_BY			50	DT_STRIN G		
LAST_MODIFIED	修改日 期	LAST_M ODIFIE D	AO_LAST_M ODIFIED				DT_DATE		
LAST_MODIFIED_B Y	修改人	LAST_M ODIFIE D_BY	AO_LAST_M ODIFIED_B Y			50	DT_STRIN G		
元素关系									
关系名	关 系 类 型	基础关系名	性质	TODS 必选	关联元素		范围	备注	
REL_TESTEQUIPME NT_FATHER	FATHER				TdmTe stEqu ipmen t	TEID	1:1		

8.5.8 TdmEquipmentRecord

TdmEquipmentRecord 的描述见表127。

表127 TdmEquipmentRecord 的描述

名称	显示名	物理名	基础属性/ 关系	性质	TODS 必选	长度	ODS 数据 类型	允许 空	备注
GID	主键	GID				36	DT_STRIN G	否	
ID	标识	ID	ID	必			DT_LONGL ONG	否	
NAME	名称	RSNAME	NAME	必		50	DT_STRIN G	否	
DESCRIPTION	描述	DESCRI PTION	DESCRIPTI ON			200	DT_STRIN G		
VERSION	版本	VERSIO N	VERSION			50	DT_STRIN G		
VERSION_DATE	日期	CREATE DATE	VERSION_D ATE				DT_DATE		
MIME_TYPE	多用途 互联网 邮件扩 展类型	MIME_T YPE	MIME_TYPE			50	DT_STRIN G		
OBJECTTYPE	元素标 识	OBJECT TYPE	OBJECTTYP E				DT_LONGL ONG		
CREATED	创建时 间	CREATE DATE	AO_CREATE D				DT_DATE		
CREATED_BY	创建人	CREATE D_BY	AO_CREATE D_BY			50	DT_STRIN G		
LAST_MODIFIED	修改日 期	LAST_M ODIFIE D	AO_LAST_M ODIFIED				DT_DATE		
LAST_MODIFIED_BY	修改人	LAST_M ODIFIE D_BY	AO_LAST_M ODIFIED_B Y			50	DT_STRIN G		
元素关系									
关系名	关系类 型	基础关系名	性质	TODS 必选	关联元素		范围	备注	
REL_TESTEQUIPME NT_FATHER	FATHER				TdmTe stEqu ipmen t	TEID	1:1		

8.6 量与单位的元素描述

量与单位的元素见表128。

表128 量与单位的元素

序号	标准元素	显示名	基础元素	备注
1	TDM_UnitGroup	单位组	AoUnitGroup	
2	TDM_Unit	单位	AoUnit	
3	TDM_PhysicalDimension	物理量纲	AoPhysicalDimension	
4	TDM_QuantityGroup	量组	AoQuantityGroup	
5	TDM_Quantity	物理量	AoQuantity	

8.6.1 TUnitGroup

TUnitGroup 的描述见表129。

表129 TUnitGroup 的描述

名称	显示名	物理名	基础属性/关系	性质	TODS必选	长度	ODS 数据类型	允许空	备注
GID	主键	GID				36	DT_STRING	否	
ID	标识	ID	ID	必			DT_LONGLONG	否	
NAME	名称	NAME	NAME	必		50	DT_STRING	否	
DESCRIPTION	描述	DESCRIPTION	DESCRIPTION			200	DT_STRING		
元素关系									
关系名		关系类型	基础关系名	性质	TODS 必选	关联元素		范围	备注
REL_UNITS		INFO_REL	units	必		TdmUnit		0:n	

8.6.2 TUnit

TUnit 的描述见表130。

表130 TUnit 的描述

名称	显示名	物理名	基础属性/关系	性质	TODS 必选	长度	ODS 数据类型	允许空	备注
GID	主键	GID				36	DT_STRING	否	
ID	标识	ID	ID	必			DT_LONGLONG	否	
NAME	名称	NAME	NAME	必		50	DT_STRING	否	
DESCRIPTION	描述	DESCRIPTION	DESCRIPTION			200	DT_STRING		
FACTOR	因子	FACTOR	FACTOR	必			DT_DOUBLE		数学概念：整数

名称	显示名	物理名	基础属性/关系	性质	TODS 必选	长度	ODS 数据 类型	允许 空	备注
									n 除 以 m，结果是无余数的整数，那么我们称 m 是 n 的因子
OFFSET	偏移量	OFFSET	OFFSET	必			DT_DOUBLE		以 某 标 准 作 为 参 照 系，经 过 偏 移 量 得 到 新 的 标 准
元素关系									
关系名	关系类型	基础关系名	性质	TODS 必选	关联元素		范围	备注	
REL_PHYS_DIMENSION	INFO_TO	PHYS_DIMENSION	必		TdmPhysicalDimension		1:1		
REL_GROUPS	INFO_REL	GROUPS	必		TdmUnitGroup		0:n		
REL_MEASUREMENT_QUANTITIES	INFO_FROM	MEASUREMENT_QUANTITIES	必		TdmMeasurementMentQuantity		0:n		
REL_PARAMETERS	INFO_FROM	PARAMETERS			TdmParameters		0:n		
REL_QUANTITIES	INFO_FROM	QUANTITIES	必		TdmQuantity		0:n		

8.6.3 TPhysicalDemension

TPhysicalDemension 的描述见表131。

表131 TPhysicalDemension 的描述

名称	显示名	物理名	基础属性/关系	性质	TODS 必选	长度	ODS 数据 类型	允许 空	备注
GID	主键	GID				36	DT_STRIN	否	

名称	显示名	物理名	基础属性/ 关系	性质	TODS 必选	长 度	ODS 数据 类型	允 许 空	备注
							G		
ID	标识	ID	ID	必			DT_LONGL ONG	否	
NAME	名称	NAME	NAME	必		50	DT_STRIN G	否	
DESCRIPTION	描述	DESCRI PTION	DESCRIPTI ON			200	DT_STRIN G		
LENGTH_EXP	长度指 数	LENGTH _EXP	LENGTH_EX P	必			DT_LONG	否	用来表示空间 上距离的大 小，标准单位 是“米（m）”
MASS_EXP	质 量 指 数	MASS_E XP	MASS_EXP	必			DT_LONG	否	物体所含物质 的多少单位 “千克（Kg）”
TIME_EXP	时 间 指 数	TIME_E XP	TIME_EXP	必			DT_LONG	否	事件发生到结 束的时刻间隔 单位“秒（s）”
CURRENT_EXP	电 流 指 数	CURREN T_EXP	CURRENT_E XP	必			DT_LONG	否	表示电流强弱 的物理量 单位：“安培 （A）”
TEMPERATURE_EXP	温 度 指 数	TEMPER ATURE_ EXP	TEMPERATU RE_EXP	必			DT_LONG	否	表示物体冷热 程度的物理量 单位：“开尔文 （K）”
MOLAR_AMOUNT_EXP	摩 尔 量 指数	MOLAR_ AMOUNT _EXP	MOLAR_AMO UNT_EXP	必			DT_LONG	否	指单位物质的 量的物质所具 有的质量，单 位“摩 尔 量 （mol）” $n=m/M$
LUMINOUS_INTENSITY_EXP	发光强 度指数	LUMINO US_INT ENSITY _EXP	LUMINOUS_ INTENSITY _EXP	必			DT_LONG	否	光源在某一方 向立体角内之 光通量大小， 单位“坎德拉 （cd）”
ANGLE_EXP	角度	ANGLE_ EXP		必			DT_LONG	否	
LENGTH_EXP_DEN	长 度 指	LENGTH _EXP_D	LENGTH_EX P_DEN				DT_LONG		

名称	显示名	物理名	基础属性/ 关系	性质	TODS 必选	长 度	ODS 数据 类型	允 许 空	备注
	数（分 母）	EN							
MASS_EXP_DEN	质 量 指 数（分 母）	MASS_E XP_DEN	MASS_EXP_ DEN				DT_LONG		
TIME_EXP_DEN	时 间 指 数（分 母）	TIME_E XP_DEN	TIME_EXP_ DEN				DT_LONG		
CURRENT_EXP_DEN	电 流 指 数（分 母）	CURREN T_EXP_ DEN	CURRENT_E XP_DEN				DT_LONG		
TEMPERATURE_EXP_ DEN	温 度 指 数（分 母）	TEMPER ATURE_ EXP_DE N	TEMPERATU RE_EXP_DE N				DT_LONG		
MOLAR_AMOUNT_EXP_ _DEN	摩 尔 指 数（分 母）	MOLAR_ AMOUNT _EXP_D EN	MOLAR_AMO UNT_EXP_D EN				DT_LONG		
LUMINOUS_INTENSI TY_EXP_DEN	光 强 指 数（分 母）	LUMINO US_INT ENSITY _EXP_D EN	LUMINOUS_ INTENSITY _EXP_DEN				DT_LONG		
ANGLE_EXP_DEN	角 度 指 数（分 母）	ANGLE_ EXP_DE N					DT_LONG	否	
元素关系									
关系名	关系类型	基础关系名	性质	TODS 必选	关联元素		范 围	备注	
REL_UNITS	INFO_FROM	UNITS	必		TdmUn it		0:n		

8.6.4 TQuantityGroup

TQuantityGroup 的描述见表132。

表132 TQuantityGroup 的描述

名称	显示名	物理名	基础属性/ 关系	性质	TODS 必选	长度	ODS 数据 类型	允许 空	备注
GID	主键	GID				36	DT_STR ING	否	
ID	标识	ID	ID	必			DT_LON GLONG	否	
NAME	名称	NAME	NAME	必		50	DT_STR ING	否	
DESCRIPTION	描述	DESC RIPTI ON	DESCRIPT ION			200	DT_STR ING		
元素关系									
关系名	关系类型	基础关系名	性质	TODS 必选	关联元素		范围	备注	
REL_QUANTITIES	INFO_REL	QUANTITIES	必		TdmQu ntity		0:n		

8.6.5 TQuantity

TQuantity 的描述见表 133。

表133 TQuantity 的描述

名称	显示名	物理名	基础属性/ 关系	性质	TODS 必选	长度	ODS 数据类 型	允许 空	备注
GID	主键	GID				36	DT_STRING	否	
ID	标识	ID	ID	必			DT_LONGLONG	否	
NAME	名称	NAME	NAME	必		50	DT_STRING	否	
DESCRIPT ION	描述	DESCRIPTION	DESCRIPTION			200	DT_STRING		
元素关系									
关系名	关系类型	基础关系名	性质	TODS 必选	关联元素		范围	备注	
REL_UNIT	INFO_TO	DEFAULT_UNIT	必		TdmUn it	UNIT	1:1	默 认 单 位	
REL_QUANTITY_ CHILD	CHILD	SUCCESSORS	必		TdmQu antit y		0:n		
REL_QUANTITY_	FATHER	PREDECESSOR	必		TdmQu	PAREN	0:1		

名称	显示名	物理名	基础属性/关系	性质	TODS 必选	长度	ODS 数据类型	允许 空	备注
FATHER							antity	TID	
REL_MEASUREMENT_QUANTITIES	INFO_FROM	MEASUREMENT_QUANTITIES					TdmMeasurementQuantity	0:n	
REL_GROUPS	INFO_REL	GROUPS					TdmQuantityGroup	0:n	

8.7 试验件管理的描述

试验件管理元素见表134。

表134 试验件管理元素

序号	标准元素	显示名	基础元素	备注
1	TdmProductModel	型号表	AoAny	
2	TdmProductPhase	产品阶段	AoAny	
3	TdmProduct	产品（版本）	AoAny	
4	TdmProductNode	产品节点	AoAny	
5	TdmUUTobjGroup	试验件分组	AoAny	
6	TdmUUTobj	试验件	AoAny	
7	TdmUUTobjComponent	试验件节点	AoAny	
8	TdmUUTobjParameter	试验件参数信息	AoAny	

8.7.1 TdmProductModel

TdmProductModel 的描述见表135。

表135 TdmProductModel 的描述

名称	显示名	物理名	基础属性/关系	性质	TODS 必选	长度	ODS 数据类型	允许 空	备注
GID	主键	GID				36	DT_STRING	否	
ID	标识	ID	ID	必			DT_LONGLONG	否	
NAME	名称	NAME	NAME	必		50	DT_STRING	否	

名称	显示名	物理名	基础属性/ 关系	性质	TODS 必选	长度	ODS 数据 类型	允许 空	备注
DESCRIPTION	描述	REMARKS	DESCRIPT ION			200	DT_STRIN G		
CREATED	创建时间	CREATE TIME	AO_CREAT ED				DT_DATE		
CREATED_BY	创建人	USERID	AO_CREAT ED_BY			50	DT_STRIN G		
CODE	代号	CODE	CODE			50	DT_STRIN G		产品型号的 英文编号。 值必须唯一
元素关系									
关系名	关系类型	基础关系名	性质	TODS 必 选	关联元素		范围	备注	
REL_MODEL_FATHER	FATHER				TdmPr oduct Model		0:1		
REL_MODEL_CHILD	CHILD				TdmPr oduct Model	PAREN TID	0:n		
REL_PHASE_CHILD	CHILD				TdmPr oduct Phase		0:n		
REL_PRODUCT_CHILD	CHILD				TdmPr oduct		0:n		
REL_UUTOBJ_CHILD	CHILD				TdmUU TObj		0:n		

8.7.2 TdmProductPhase

TdmProductPhase 的描述见表136。

TdmProduct的描述的描述见表137。

表136 TdmProductPhase 的描述

名称	显示名	物理名	基础属性/ 关系	性质	TODS 必选	长度	ODS 数据 类型	允 许 空	备注
GID	主键	GID				36	DT_STRIN G	否	
ID	标识	ID	ID	必			DT_LONGL ONG	否	
NAME	名称	NAME	NAME	必		50	DT_STRIN G	否	

名称	显示名	物理名	基础属性/ 关系	性质	TODS 必选	长度	ODS 数据 类型	允 许 空	备注
DESCRIPTION	描述	REMARK S	DESCRIPTI ON			200	DT_STRIN G		
CREATED	创 建 时 间	CREATE D	AO_CREATE D				DT_DATE		
CREATED_BY	创建人	CREATE D_BY	AO_CREATE D_BY			50	DT_STRIN G		
PHASENO	阶 段 序 号	PHASEN O	PHASENO				DT_LONG		表示第几个 阶段
元素关系									
关系名	关 系 类 型	基础关系名		性质	TODS 必选	关联元素		范围	备注
REL_MODEL_FATHE R	FATHER					TdmPr oduct Model	MODEL ID	1:1	
REL_PRODUCT_CHI LD	CHILD					TdmPr oduct		0:n	

表137 TdmProduct 的描述

名称	显示名	物理名	基础属性/ 关系	性质	TODS 必选	长度	ODS 数据 类型	允 许 空	备注
GID	主键	GID				36	DT_STRIN G	否	
ID	标识	ID	ID	必			DT_LONGL ONG	否	
NAME	名称	NAME	NAME	必		50	DT_STRIN G	否	
DESCRIPTION	描述	REMARK S	DESCRIPTI ON			200	DT_STRIN G		
CREATED	创 建 时 间	CREATE DATE	AO_CREATE D				DT_DATE		
CREATED_BY	创建人	CREATO R	AO_CREATE D_BY			50	DT_STRIN G		
PRODUCTVERSION	版本	PRODUC TVERSI ON	PRODUCTVE RSION			50	DT_STRIN G		
PRODUCTCODE	代号	PRODUC TCODE	PRODUCTCO DE			50	DT_STRIN G		自动生成， 生成规则 “型号代号 +产品版本”

名称	显示名	物理名	基础属性/ 关系	性质	TODS 必选	长度	ODS 数据 类型	允 许 空	备注
元素关系									
关系名	关 系 类 型	基础关系 名	性质	TODS 必选	关联元素		范围	备注	
REL_MODEL_FATHER	FATHER				TdmProduct Model	MODEL ID	0:1		
REL_PHASE_FATHER	FATHER				TdmProduct Phase	PHASE ID	0:1		
REL_PRODUCTNODE _CHILD	CHILD				TdmProduct Node		0:n		
REL_UUTOBJ_CHIL D	CHILD				TdmUUTObj		0:n		
REL_PRODUCTNODE	INFO_RE L				TdmProduct Node		0:1		

8.7.3 TdmProductNode

TdmProductNode 的描述见表138。

表138 TdmProductNode 的描述

名称	显示名	物理名	基础属性/ 关系	性质	TODS 必选	长度	ODS 数据 类型	允 许 空	备注
GID	主键	GID				36	DT_STRIN G	否	
ID	标识	ID	ID	必			DT_LONGL ONG	否	
NAME	名称	NAME	NAME	必		50	DT_STRIN G	否	
DESCRIPTION	描述	REMARK S	DESCRIPTI ON			200	DT_STRIN G		
CREATED	创 建 时 间	CREATE D	AO_CREATE D				DT_DATE		
CREATED_BY	创建人	CREATE D_BY	AO_CREATE D_BY			50	DT_STRIN G		
元素关系									
关系名	关 系 类 型	基础关系名	性质	TODS 必选	关联元素		范围	备注	
REL_TDMPRODUCT_ FATHER	FATHER		必		TdmPr oduct	PRODU CTID	1:1		
REL_TDMPRODUCT_ REL	INFO_RE L				TdmPr oduct	RELPR ODUCT	0:1		

						ID		
REL_TDMUUTOBJCOMPONENT_REL	INFO_REL				TdmUUTObjComponent		0:n	

8.7.4 TdmUUTObjGroup

TdmUUTObjGroup 的描述见表139。

表139 TdmUUTObjGroup 的描述

名称	显示名	物理名	基础属性/ 关系	性质	TODS 必选	长度	ODS 数 据 类型	允 许 空	备注
GID	主键	GID				36	DT_STRIN G	否	
ID	标识	ID	ID	必			DT_LONGL ONG	否	
NAME	名称	NAME	NAME	必		50	DT_STRIN G	否	
DESCRIPTION	描述	DESCRI PTION	DESCRIPTI ON			200	DT_STRIN G		
CREATED	创 建 时 间	CREATE D	AO_CREATE D				DT_DATE		
CREATED_BY	创建人	CREATE D_BY	AO_CREATE D_BY			50	DT_STRIN G		
元素关系									
关系名	关 系 类 型	基础关系名		性质	TODS 必选	关联元素		范围	备注
REL_GROUP_FATHE R	FATHER					TdmUU TobjG roup		0:1	
REL_GROUP_CHILD	CHILD					TdmUU TobjG roup	PAREN TID	0:n	
REL_UUTOBJ_CHIL D	CHILD					TdmUU Tobj		0:n	

8.7.5 TdmUUTObj

TdmUUTObj 的描述见表140。

表140 TdmUUTObj 的描述

名称	显示名	物理名	基础属性/关系	性质	TODS 必选	长度	ODS 数据类型	允许空	备注
GID	主键	GID				36	DT_STRING	否	
ID	标识	ID	ID	必			DT_LONGLONG	否	
NAME	名称	NAME	NAME	必		50	DT_STRING	否	
DESCRIPTION	描述	NOTE	DESCRIPTION			200	DT_STRING		
CREATED	创建时间	CREATED	AO_CREATE				DT_DATE		
CREATED_BY	创建人	CREATED_BY	AO_CREATE_BY			50	DT_STRING		
CODE	编号	CODE				50	DT_STRING		
元素关系									
关系名	关系类型	基础关系名	性质	TODS 必选	关联元素		范围	备注	
REL_MODEL_FATHER	FATHER				TdmProductModel	MODEL ID	1:1		
REL_PRODUCT_FATHER	FATHER				TdmProduct	PRODUCTID	1:1		
REL_TYPE_FATHER	FATHER				TdmUUTObjGroup	TYPE	1:1		
REL_PARAM_CHILD	CHILD				TdmUUTObjParameter		0:n		
REL_COMPONENT_CHILD	CHILD				TdmUUTObjComponent		0:n		
REL_COMPONENT	INFO_REL				TdmUUTObjComponent		0:1		
REL_ATTACH_CHILD	CHILD				TdmFile	ATTACH	0:n	附件，引用【文件】表	
REL_PIC_CHILD	CHILD				TdmFile	PIC	0:n	图片，引用	

名称	显示名	物 理 名	基础属性/ 关系	性质	TODS 必选	长度	ODS 数据 类型	允 许 空	备注
						le			【文件】表
REL_DRAW_CHILD	CHILD					TdmFille	DRAW	0:n	图纸，引用 【文件】表

8.7.6 TdmUUTObjComponent

TdmUUTObjComponent 的描述见表141。

表141 TdmUUTObjComponent 的描述

名称	显示名	物 理 名	基础属性/ 关系	性质	TODS 必选	长度	ODS 数据 类型	允 许 空	备注
GID	主键	GID				36	DT_STRING	否	
ID	标识	ID	ID	必			DT_LONGLONG	否	
NAME	名称	NAME	NAME	必		50	DT_STRING	否	
DESCRIPTION	描述	DESCRIPTION	DESCRIPTION			200	DT_STRING		
CREATED	创建时间	CREATED	AO_CREATED				DT_DATE		
CREATED_BY	创建人	CREATED_BY	AO_CREATED_BY			50	DT_STRING		
元素关系									
关系名	关系类型	基础关系名	性质	TODS 必选	关联元素		范围	备注	
REL_PRODUCTNODE	INFO_REL				TdmProductNode	PRODUCTNODEID	0:1		
REL_UUTOBJ	INFO_REL				TdmUUTObj		0:1		
REL_UUTOBJ_FATHER	FATHER				TdmUUTObj		1:1		

8.7.7 TdmUUTObjParameter

TdmUUTObjParameter 的描述见表142。

表142 TdmUUTObjParameter 的描述

名称	显示名	物理名	基础属性/关系	性质	TODS 必选	长度	ODS 数据类型	允许空	备注
GID	主键	ID				36	DT_STRING	否	
ID	标识	SEQUENCE	ID	必			DT_LONGLONG	否	
NAME	名称	NAME	NAME	必		50	DT_STRING	否	
DESCRIPTION	描述	DESCRIPTION	DESCRIPTION			200	DT_STRING		
CREATED	创建时间	CREATED	AO_CREATE_D				DT_DATE		
CREATED_BY	创建人	CREATED_BY	AO_CREATE_D_BY			50	DT_STRING		
元素关系									
关系名	关系类型	基础关系名	性质	TODS 必选	关联元素		范围	备注	
REL_UUTOBJ_FATHER	FATHER				TdmUUTobj	PARTID	1:1		
REL_QUANTITY	INFO_REL				TdmQuantity	QUANTITYID	0:1	物理量	
REL_UNIT	INFO_REL				TdmUnit	UNIT	0:1	单位	

8.8 设备管理的描述

设备管理的元素见表143。

表143 设备管理的元素

序号	标准元素	显示名	基础元素	备注
1	TdmEquipmentType	设备分类	AoAny	
2	TdmEquipment	设备信息	AoAny	
3	TdmEquipmentComponent	设备节点	AoAny	

8.8.1 TdmEquipmentType

TdmEquipmentType 的描述见表144。

表144 TdmEquipmentType 的描述

名称	显示名	物理名	基础属性/关系	性质	TODS 必选	长度	ODS 数据类型	允许空	备注

GID	主键	ID				36	DT_STRING	否	
ID	标识	SEQUENCE	ID	必			DT_LONGLONG	否	
NAME	名称	NAME	NAME	必		50	DT_STRING	否	
DESCRIPTION	描述	DESCRIPTION	DESCRIPTION			200	DT_STRING		
CREATED	创建时间	CREATE DATE	AO_CREATED				DT_DATE		
CREATED_BY	创建人	CREATE D_BY	AO_CREATED_BY			50	DT_STRING		
LAST_MODIFIED	修改日期	LAST_MODIFIED	AO_LAST_MODIFIED				DT_DATE		
LAST_MODIFIED_BY	修改人	LAST_MODIFIED_BY	AO_LAST_MODIFIED_BY			50	DT_STRING		
元素关系									
关系名	关系类型	基础关系名	性质	TODS 必选	关联元素		范围	备注	
REL_TYPE_CHILD	CHILD				TdmEquipmentType	PARENTID	0:n		
REL_TYPE_FATHER	FATHER				TdmEquipmentType		0:1		
REL_EQUIPMENT	INFO_REL				TdmEquipment		0:n		

8.8.2 TdmEquipment

TdmEquipment 的描述见表145。

表145 TdmEquipment 的描述

名称	显示名	物理名	基础属性/关系	性质	TODS 必选	长度	ODS 数据类型	允许空	备注
GID	主键	GID				36	DT_STRING	否	
ID	标识	ID	ID	必			DT_LONGLONG	否	
NAME	名称	RSNAME	NAME	必		50	DT_STRING	否	
DESCRIPTION	描述	DESCRIPTION	DESCRIPTION			200	DT_STRING		

名称	显示名	物理名	基础属性 /关系	性 质	TODS 必选	长度	ODS 数据 类型	允 许 空	备注
CREATED	创建时间	CREATE DATE	AO_CREAT ED				DT_DATE		
CREATED_BY	创建人	CREATE D_BY	AO_CREAT ED_BY			50	DT_STRIN G		
RSCODE	资源编号	RSCODE				50	DT_STRIN G	否	
BAR_CODE	条形码	BAR_CO DE				500	DT_STRIN G		
STATE	状态	STATE				10	DT_STRIN G	否	null:正常, WXZ:维 修 中, BYZ:保 养中, JDZ: 计 量 中, BFZ:报废审 批中, BF:已 报废
USESTATE	使用状态	USESTA TE				10	DT_STRIN G	否	1,null: 空 闲, 3:使用 中
ISDELETE	是否删除	ISDELE TE					DT_BOOLE AN	否	是 否 删 除 0, 1
MANUFACTUREDAT E	建造日期	MANUFA CTURED ATE					DT_DATE		
STARTUSINGDATE	启用日期	STARTU SINGDA TE					DT_DATE		
REMARK	备注	REMARK				512	DT_STRIN G		
TECH_SPEC	技术规格	TECH_S PEC				50	DT_STRIN G		
CHECKCYCLE	计量周期	CHECKC YCLE					DT_SHORT		
SERVICECYCLE	保养周期	SERVIC ECYCLE					DT_SHORT		
FREQUENCY	使用次数	FREQUE NCY					DT_SHORT		
SERVICECYCLE_U NIT	保养周期 单位	SERVIC ECYCLE _UNIT				50	DT_STRIN G		

名称	显示名	物理名	基础属性 /关系	性质	TODS 必选	长度	ODS 数据 类型	允 许 空	备注
CHECKCYCLE_UNI T	计量周期 单位	CHECKC YCLE_U NIT				50	DT_STRIN G		
元素关系									
关系名	关系类型	基础关系名	性质	TODS 必选	关联元素		范围	备注	
REL_TYPE	INFO_REL				TdmEquipment Type	RS_TY PEID	1:1		
REL_TESTINGDES K	INFO_REL				TdmTestingde sk	RSID	0:1		
REL_EQUIPMENTC OMPONENT	INFO_REL				TdmEquipment Component		0:n		
REL_USERGROUP_ ORG	INFO_REL				TdmUserGroup	ORG	0:1	所 属 部 门	

8.8.3 TdmEquipmentComponent

TdmEquipmentComponent 的描述见表146。

表146 TdmEquipmentComponent 的描述

名称	显示名	物理名	基础属性/ 关系	性质	TODS 必选	长度	ODS 数据 类型	允 许 空	备注
GID	主键	GID				36	DT_STR ING	否	
ID	标识	ID	ID	必			DT_LONGL ONG	否	
NAME	名称	NAME	NAME	必		50	DT_STRIN G	否	
DESCRIPTION	描述	DESCRI PTION	DESCRIPTI ON			200	DT_STRIN G		
CREATED	创 建 时 间	CREATE D	AO_CREATE D				DT_DATE		
CREATED_BY	创建人	CREATE D_BY	AO_CREATE D_BY			50	DT_STRIN G		
元素关系									
关系名	关 系 类	基础关系名	性质	TODS 必选	关联元素		范围	备注	

名称	显示名	物理名	基础属性/ 关系	性质	TODS 必选	长度	ODS 数据 类型	允 许 空	备注
	型								
REL_EQUIPMENT	INFO_REL					TdmEquipment	RSID	0:1	
REL_EQUIPMENT	FATHER					TdmEquipment	FRSID	1:1	

8.9 知识元素的描述

知识元素的见表147。

表147 知识元素

序号	标准元素	显示名	基础元素	备注
1	TdmWsParamLibrary	工况参数库	AoParameter	
2	TdmTestParamLibrary	试验参数	AoParameter	
3	TdmUTParamLibrary	试验件参数	AoParameter	
4	TdmStateType	试验状态类型	AoAny	
5	算法库	算法库	AoParameter	

8.9.1 TdmWsParamLibrary

TdmWsParamLibrary 的描述见表148。

表148 TdmWsParamLibrary 的描述

名称	显示名	物 理 名	基础属性/ 关系	性质	TODS 必选	长度	ODS 数据 类型	允 许 空	备注
GID	主键	GID				36	DT_STRING	否	
ID	标识	ID	ID	必		50	DT_LONGLONG	否	
NAME	名称	NAME	NAME	必		50	DT_STRING	否	
Symbol	参 数 符 号	Symbo l		必		50	DT_STRING	否	
DataType	数 据 类 型	DataT ype		必		10	DT_STRING	否	
EnumValue	枚举	EnumV alue				50	DT_STRING		
ISBuiled	是 否 已 创建	ISBui lded					DT_LONGLONG		

ISDelete	是否删除	ISDelete					DT_LONGLONG		
Creator	创建人	Creator		必		36	DT_STRING	否	
CreateDateTime	创建时间	CreateDateTime		必			DT_DATE		
Width	参数在列表上的宽度	Width				50	DT_LONGLONG		
Remark	描述	Remark	DESCRIPTION			256	DT_STRING		
Unit	单位	Unit				36	DT_STRING		
元素关系									
关系名	关系类型	基础关系名	性质	TODS 必选	关联元素		范围	备注	
REL_UNITS	INFO_REL	units	必		TdmUnit		0:n		

8.9.2 TdmTestParamLibrary

TdmTestParamLibrary 的描述见表149。

表149 TdmTestParamLibrary 的描述

名称	显示名	物理名	基础属性/关系	性质	TODS 必选	长度	ODS 数据类型	允许空	备注
GID	主键	ID				36	DT_STRING	否	
ID	参数编号	SN	ID	必			DT_LONGLONG	否	
NAME	名称	NAME	NAME	必		50	DT_STRING	否	
DESCRIPTION	描述	DESCRIPTION	DESCRIPTION			200	DT_STRING		
CREATED	创建时间	CREATEDATE	CREATED				DT_DATE		
CREATED_BY	创建人	CREATED_BY	CREATED_BY			50	DT_STRING		
元素关系									
关系名	关系类型	基础关系名	性质	TODS 必选	关联元素		范围	备注	
REL_QUANTITY	INFO_REL				TdmQuantity	QUANTITYID	0:1	物理量	
REL_QUANTITYUNIT	INFO_REL				TdmUnit	QUANTITYUNIT	0:1	单位	

8.9.3 TdmUUTParamLibrary

TdmUUTParamLibrary 的描述见表150。

表150 TdmUUTParamLibrary 的描述

名称	显示名	物理名	基础属性/关系	性质	TODS 必选	长度	ODS 数据类型	允许空	备注
GID	主键	ID				36	DT_STRING	否	
ID	参数编号	SN	ID	必			DT_LONGLONG	否	
NAME	名称	NAME	NAME	必		50	DT_STRING	否	
DESCRIPTION	描述	DESCRIPTION	DESCRIPTION			200	DT_STRING		
CREATED	创建时间	CREATEDATE	CREATED				DT_DATE		
CREATED_BY	创建人	CREATED_BY	CREATED_BY			50	DT_STRING		
元素关系									
关系名	关系类型	基础关系名	性质	TODS 必选	关联元素		范围	备注	
REL_QUANTITY	INFO_REL				TdmQuantity	QUANTITYID	0:1	物理量	
REL_QUANTITYUNIT	INFO_REL				TdmUnit	QUANTITYUNIT	0:1	单位	

8.9.4 TdmStateTpye

TdmStateTpye 的描述见表151。

表151 TdmStateTpye 的描述

属性	中文名称	基础属性/关系	ODS 类型	性质	备注
ID	主键				
TYPE	状态类型				外键，引用“状态类型信息表【TSL_STATETYPE】”主键。
SYMBOL	状态标识符号				
RATE	状态权值				
REMARK	状态说明				
IDCOLOR	状态标识颜色				
NOTICE	操作注意事项				

8.9.5 算法库

算法库的描述见表152。

表152 算法库的描述

属性	中文名称	基础属性/关系	ODS 类型	性质	备注
ID	主键 ID				
PKNAME	算法名称				
CHARACTER	算法特点				
ADVERTENCE	注意事项				
BUSINESS	适用业务				
PKLANGUAGE	算法语言				
PROFESSION	所属专业				
REMARK	功能描述				
USELIMIT	使用限制				
PKTYPE	算法类型				
FUNC	算法功能				
PKKEY	算法编号				
INTERFACE	接口说明				
SOFTWARESETTING	软件运行环境				
SERVEROBJ	服务对象				
HARDWARESETTING	硬件运行环境				
APPMODULE	应用模块				

8.10 质量体系元素的描述

质量体系元素见表153。

表153 质量体系元素

序号	标准元素	显示名	基础元素	备注
1	质量体系文件	质量体系文件	AoAny	
2	质量标准表单	质量标准表单	AoAny	

8.10.1 质量体系文件

质量体系文件描述见表154。

表154 质量体系文件的描述

属性	中文名称	基础属性/关系	ODS 类型	性质	备注
ID	主键				
DOCCODE	文件 ID				
DOCTYPE	文件类型				两类（质量体系表单） QualityStandardForm（质量体系附件） QualityStandardFile
SUBMITTIME	提交时间				

属性	中文名称	基础属性/关系	ODS 类型	性质	备注
DEPART	部门				引用 部门 信息 表 【DPS_ORGANIZATION】主键
SUBMITER	创建人				引用 “ 人 员 信 息 表 【DPS_USER】” 主键

8.10.2 质量标准表单

质量标准表单的描述见表155。

表155 质量标准表单的描述

属性	中文名称	基础属性/关系	ODS 类型	性质	备注
ID	主键				
FORMID	表单 ID				引用自定义表单表【 fs_form】对应版本信息【 fs_form_editions 】主键 ID
KEYVALUE	值				
CONTENTKEY	数据标识				

8.11 安全元素的描述

安全元素见表156。

表156 安全元素

序号	标准元素	显示名	基础元素	备注
1	TdmUserGroup	组织机构	AoUserGroup	
2	TdmUser	人员	AoUSER	
3	TdmSecurityGrade	密级	AoAny	

8.11.1 TdmUserGroup

TdmUserGroup 的描述见表157。

表157 TdmUserGroup 的描述

名称	显示名	物理名	基础属性 /关系	性质	TODS 必选	长度	ODS 数 据类型	允 许 空	备注
GID	主键	ORGID				36	DT_ST RING	否	
ID	标识	ID	ID	必			DT_LON GLONG	否	
NAME	名称	ORGDISPL	NAME	必		50	DT_ST	否	

名称	显示名	物理名	基础属性/关系	性质	TODS 必选	长度	ODS 数 据类型	允 许 空	备注
		AYNAME					RING		
DESCRIPTION	描述	DESCRIPTI ON	DESCRIPT ION			200	DT_STR ING		
SUPERUSER_FLAG	SUPERUSER 标识	SUPERUSER _FLAG	SUPERUSE R_FLAG	必			DT_SHO RT	否	超 级 管 理 员 标 示,需设 置 一 个 兼 容 性 常 值
CREATEDATE	创建日期	CREATED ATE	AO_CRE ATE				DT_DA TE	否	
元素关系									
关系名		关系类型	基础关 系名	性质	TODS 必选	关联元素		范围	备注
REF_USERS		INFO_REL	USERS	必		TDMUS ER	ORGID	0:n	
REF_USERGROUP_FATHER		FATHER				TDMU SERG ROUP	PARE NTOR GID	0:1	
REF_USERGROUP_CHILD		CHILD				TDMU SERG ROUP		0:n	

8.11.2 TdmUser

TdmUser 的描述见表158。

表158 TdmUser 的描述

名称	显示名	物理名	基础属性/关 系	性质	TODS 必选	长度	ODS 数据 类型	允许空	备注
GID	主键	USERID				36	DT_STR ING	否	G U I D
ID	标识	ID	ID	必			DT_LONG LONG	否	
NAME	名称	USERNAME	NAME	必		50	DT_STR ING	否	
PASSWORD	密码	PASSWORD	PASSWORD	必		50	DT_STR ING	否	

名称	显示名	物理名	基础属性/关系	性质	TODS 必选	长度	ODS 数据类型	允许空	备注
DESCRIPTION	描述	DESCRIPTION	DESCRIPTION			200	DT_STRING		
CREATEDATE	创建日期	CREATEDATE	AO_CREATE				DT_DATE	否	
LASTMODIFYDATE	修改日期	LASTMODIFYDATE	AO_LAST_MODIFIED				DT_DATE		
USERDISPLAYNAME	显示名	USERDISPLAYNAME				50	DT_STRING	否	
CODE	编号	CODE				50	DT_STRING	否	
SEX	性别	SEX					DT_LONG	否	
元素关系									
关系名		关系类型	基础关系名	性质	TODS 必选	关联元素		范围	备注
REF_GROUPS		INFO_REL	GROUPS	必		TDMUSE RGROUPS		0:n	
REF_USERSECURITYGRADE		INFO_REL				TDMSECURITYGRADE		0:1	

8.11.3 TdmSecurityGrade

TdmSecurityGrade 的描述见表159。

表159 TdmSecurityGrade 的描述

名称	显示名	物理名	基础属性/关系	性质	TODS 必选	长度	ODS 数据类型	允许空	备注
GID	主键	GRADEID				36	DT_STRING	否	
ID	标识	ID	ID	必			DT_LONG	否	
NAME	名称	GRADENAME	NAME	必		50	DT_STRING	否	
DESCRIPTION	描述	DESCRIPTI	DESCRIPTI			200	DT_STR		

名称	显示名	物理名	基础属性 / 关系	性质	TODS 必选	长度	ODS 数据类型	允许空	备注
		ON	ON				ING		
ORDERINDEX	密级等级	ORDERINDEX					DT_SHORT	否	
元素关系									
关系名		关系类型	基础关系名	性质	TODS 必选	关联元素		范围	备注
REF_USER		INFO_REL				TDMUSER		0:n	

8.12 其它的描述

其它描述元素汇报160。

表160 其它描述元素

序号	标准元素	显示名	基础元素	备注
1	TdmDataRelation	数据关系	AoAny	

8.12.1 TdmDataRelation

TdmDataRelation的描述见表161。

表161 TdmDataRelation 的描述

名称	显示名	物理名	基础属性 / 关系	性质	TODS 必选	长度	ODS 数据类型	允许空	备注
GID	主键	ID				36	DT_STRING	否	
ID	标识	SEQUENCE	ID	必			DT_LONGLONG	否	
NAME	名称	NAME	NAME	必		50	DT_STRING	否	
DESCRIPTION	描述	DESCRIPTION	DESCRIPTION			200	DT_STRING		
CREATED	创建时间	CREATED	AO_CREATEID				DT_DATE		
CREATED_BY	创建人	CREATED_BY	AO_CREATEID_BY			50	DT_STRING		
DATAID	实例主	DATAID				50	DT_STRING	否	对象的主键

名称	显示名	物理名	基础属性/关系	性质	TODS必选	长度	ODS数据类型	允许空	备注
	键 1	D					G		值
MEMBERNAME	主实例列名	MEMBERNAME				30	DT_STRING	否	对象的主键列名
DATATYPE	实例名称 1	DATATYPE				50	DT_STRING	否	对象名称
SRCDATAID	实例主键 2	SRCDATAID				50	DT_STRING	否	对象的主键值
SRCMEMBERNAME	源实例列名	SRCMEMBERNAME				30	DT_STRING	否	对象的主键列名
SRCDATATYPE	实例名称 2	SRCDATATYPE				50	DT_STRING	否	对象名称

8.13 TODS OO-API（面向对象应用编程接口）

8.13.1 IMapleTRFactory（工厂）

IMapleTRFactory 作为所 TODS OO-API 的入口使用。所有的对象全部由此类完成，同时此类提供 TODS 全局性信息。

8.13.1.1 获取当前系统版本信息

`ErrorCodeGetSystemVersion (out string version,out string ErrString)`

功能：获取当前系统版本信息。服务器将从 `TEnvironment` 实例的基础属性“description”中调出描述信息。

输出参数：version 值的格式：主版本号.次版本号.内部版本号.修订版本号，例如：1.0.1.1

ErrString 错误信息。此参数输出此方法执行时的异常信息，若执行时未发生异常则此值为空字符串。

返回值：为 ErrorCode 的某个值见附录 A。

8.13.1.2 获取基础模型版本信息

`ErrorCode GetBaseModelVersion (out string version,out string ErrString)`

功能：获取基础模型版本信息

输出参数：version 值的格式:主版本号.次版本号.内部版本号.修订版本号，例如：1.0.1.1

ErrString 错误信息。此参数输出此方法执行时的异常信息，若执行时未发生异常则此值为空字符串。

返回值：为 ErrorCode 的某个值见附录 A。

8.13.1.3 获取应用模型版本信息

ErrorCode GetApplModelVersion (out string version, out string ErrString)

功 能：获取应用模型版本信息

输出参数： version 值的格式：主版本号.次版本号.内部版本号.修订版本号，例如：1.0.1.1

ErrString 错误信息。此参数输出此方法执行时的异常信息，若执行时未发生异常则此值为空字符串。

返回值：为 ErrorCode 的某个值见附录 A。

8.13.1.4 创建会话

ErrorCode NewSession (string username, string password, out IMapleTRSession, out string ErrString)

功能：创建会话实例

输入参数： username 表示用户登陆名。

password 表示用户登陆密码。

输出参数： ErrString 错误信息。此参数输出此方法执行时的异常信息，若执行时未发生异常则此值为空字符串。

返回值：为 ErrorCode 的某个值见附录 A。

8.13.2 IMapleTRSession （会话）

8.13.2.1 关闭会话

ErrorCode Close (out string ErrString)

功 能：关闭会话，退出登陆

输出参数： ErrString 错误信息。此参数输出此方法执行时的异常信息，若执行时未发生异常则此值为空字符串。

返回值：为 ErrorCode 的某个值见附录 A。

8.13.2.2 获取 IMapleTRModelAccess

ErrorCode GetMapleTRModelAccess (out IMapleTRModelAccess , out string ErrString)

功能：获取数模型访问

输出参数：实例化有 IMapleTRModelAccess

ErrString 错误信息。此参数输出此方法执行时的异常信息，若执行时未发生异常则此值为空字符串。

返回值：为 ErrorCode 的某个值见附录 A。

8.13.2.3 获取 IMapleTRDBAccess

ErrorCode GetMapleTRModelAccess (out IMapleTRDBAccess , out string ErrString)

功能：获取 DB 数据访问

输出参数：实例化有 IMapleTRDBAccess

ErrString 错误信息。此参数输出此方法执行时的异常信息，若执行时未发生异常则此值为空字符串。

返回值：为 ErrorCode 的某个值见附录 A。

8.13.2.4 获取 IMapleTRFileAccess

ErrorCode GetMapleTRModelAccess (out IMapleTRFileAccess , out string ErrString)

功能：获取文件访问

输出参数：IMapleTRFileAccess 实例化有 IMapleTRFileAccess

ErrString 错误信息。此参数输出此方法执行时的异常信息，若执行时未发生异常则此值为空字符串。

返回值：为 ErrorCode 的某个值见附录 A。

8.13.3 IMapleTRModelAccess（数据模型）

8.13.3.1 获取所有基础模型清单

ErrorCode GetBaseModelNameList (out string modelNameListPackage ,out string ErrString)

功 能：获取所有基础模型清单。

输出参数：ErrString 错误信息。此参数输出此方法执行时的异常信息，若执行时未发生异常则此值为空字符串。

返回值：为 ErrorCode 的某个值见附录 A。

8.13.3.2 获取所有基础模型信息

ErrorCode GetBaseModels (out string modelPackage ,out string ErrString)

功 能：获取所有基础模型表信息。

输出参数：ErrString 错误信息。此参数输出此方法执行时的异常信息，若执行时未发生异常则此值为空字符串。

8.13.3.3 获取制定模型信息

ErrorCode GetModelByName (string modelName,out string modelPackage,out string ErrString)

功 能：根据模型表名称获取基础模型信息。

输入参数：modelName

输出参数：ErrString 错误信息。此参数输出此方法执行时的异常信息，若执行时未发生异常则此值为空字符串。

返回值：为 ErrorCode 的某个值见附录 A。

8.13.3.4 获取关联模型信息

ErrorCode GetModelRelationsByName (string modelName,out string modelPackage,out string ErrString)

功 能：根据模型表名称获取关联模型信息。

输入参数：modelName

modelPackage

输出参数：ErrString 错误信息。此参数输出此方法执行时的异常信息，若执行时未发生异常则此值为空字符串。

返回值：为 ErrorCode 的某个值见附录 A。

8.13.3.5 获取指定关联模型信息

ErrorCode GetModelRelationByMNRN (stringmodelName,string relationalName,out string modelPackage,out string ErrString)

功 能：根据模型表名称和关系名称获取关联模型信息。

输入参数：modelName

relationalName

输出参数：ErrString 错误信息。此参数输出此方法执行时的异常信息，若执行时未发生异常则此值为空字符串。

返回值：为 ErrorCode 的某个值见附录 A。

8.13.3.6 获取所有应用模型信息

ErrorCode GetAppModels (out string modelPackage,out string ErrString)

功 能：获取所有应用模型表信息。

输入参数：modelPackage

输出参数：ErrString 错误信息。此参数输出此方法执行时的异常信息，若执行时未发生异常则此值为空字符串。

返回值：为 ErrorCode 的某个值见附录 A。

8.13.3.7 新建应用模型

ErrorCode AddAppModel (string modelPackage,out string ErrString)

功 能：新建应用模型表信息

输出参数：ErrString 错误信息。此参数输出此方法执行时的异常信息，若执行时未发生异常则此值为空字符串。

返回值：为 ErrorCode 的某个值见附录 A。

8.13.3.8 编辑应用模型信息

ErrorCode EditAppModel (string modelPackage,out string ErrString)

功 能：编辑模型表信息。

输入参数：modelPackage

输出参数：ErrString 错误信息。此参数输出此方法执行时的异常信息，若执行时未发生异常则此值为空字符串。

返回值：为 ErrorCode 的某个值见附录 A。

8.13.3.9 删除应用模型信息

ErrorCode DropAppModelByName (string modelName,out string ErrString)

功 能：根据模型表名称删除模型表信息。

输入参数：modelName 元素名称

输出参数：ErrString 错误信息。此参数输出此方法执行时的异常信息，若执行时未发生异常则此值为空字符串。

返回值：为 ErrorCode 的某个值见附录 A。

8.13.4 IMapleTRDBAccess (DB 数据访问)

8.13.4.1 获取模型实例数据

```
ErrorCodeGetRecord ( string modelName, string keyValue, out string dataPackage, out string ErrString) ;
```

功 能：获取模型实例数据。

输入参数：keyValue 表示 模型表 主键值

ErrString 错误信息。此参数输出此方法执行时的异常信息，若执行时未发生异常则此值为空字符串。

返回值：为 ErrorCode 的某个值见附录 A。

8.13.4.2 获取模型实例数据集

```
ErrorCode GetRecordSet (stringconditionXML , out int totalCount , out string dataPackage, out string ErrString) ;
```

功 能：获取模型实例数据集。

输入参数：conditionXML 表示 Xml 格式条件，格式参照。

输出参数：totalCount 表示 总记录数

返回值：为 ErrorCode 的某个值见附录 A。

8.13.4.3 获取关联模型的实例数据

```
ErrorCode GetRelationnalRDSet ( stringconditionXML , out int totalCount , out string dataPackage, out string ErrString) ;
```

功 能：获取指定模型的关联模型数据。

输入参数：conditionXML 表示 Xml 格式条件，格式参照。

输出参数：totalCount 表示 总记录数。

ErrString 错误信息。此参数输出此方法执行时的异常信息，若执行时未发生异常则此值为空字符串。

返回值：为 ErrorCode 的某个值，请参照附录 A。

8.13.4.4 新建模型实例数据

```
ErrorCode AddRecord (string dataPackage , out string keyValue , out string ErrString) ;
```


功 能：新建模型实例数据。

输出参数：keyValue 表示返回的模型实例数据主键值。

返回值：为 ErrorCode 的某个值见附录 A。

8.13.4.5 编辑模型数据

```
ErrorCode EditRecord (string dataPackage , out string ErrString) ;
```

功 能：编辑模型实例数据。

输入参数：dataPackage。

输出参数：ErrString 错误信息。此参数输出此方法执行时的异常信息，若执行时未发生异常则此值为空字符串。

返回值：为 ErrorCode 的某个值见附录 A。

8.13.4.6 删除模型实例数据

```
ErrorCode DelRecord (string modelName, string [] arrKeyValue , out string ErrString) ;
```

功 能：删除模型实例数据和其包含的实例数据及文件。

输入参数：modelName 元系名称。

arrKeyValue 表示 模型的主键值数组。

输出参数：ErrString 错误信息。此参数输出此方法执行时的异常信息，若执行时未发生异常则此值为空字符串。

返回值：为ErrorCode的某个值。

9 标准的传输格式 (ATF/CLA)

9.1 引言

ODS 可用标准格式存储测量数据。它定义了一个具有通用结构框架的数据模型,可用于数据的建模。它还定义了一组函数,可读写参数和测量结果数据,这些数据被存储在该数据结构中。

这些函数都基于不同的物理存储信息。一种可能性是将数据存储于商业数据库中,同时在数据库表格中表达单个应用模型。另一种可能是,对文件存储系统进行描述说明,使得兼容 ASAM 的应用无需使用数据库。

使用 ASAM 函数访问数据的第三种可能性是,借助于 ASAM 数据结构的元信息用数据库或文件系统表达已有的数据。通过使用元信息,已有的数据存储系统就能兼容 ASAM,并且与前面提到的各种可能方式在功能层面上几乎相同。

针对以上问题,不仅是在一个特定数据结构中如何存储和备份数据的问题,即所谓的环境问题,同样重要的是如何在不同的计算机、应用程序和环境之间进行数据交换。

在相同的环境中,理论上可以从数据库中导入或导出数据,或者在相应的文件系统交换文件。当 ASAM 应用不支持这种数据交换方式时,将存在一定的风险会影响被交换数据的完整性和一致性。

为了避免数据交换中产生上述问题,ASAM ODS 规范提供了一个专用文件格式,即 ASAM 传输格式(ATF)。通过 ATF,可以在不同计算机和环境之间交换整个环境信息或其部分信息。

ATF 可交换所需的结构信息和实例信息以及相应的数据,并以标准格式来存储这些信息。传输格式不保证数据读取应用系统按照自己的数据结构对信息进行分类,只有当数据读取应用系统的数据结构能够获得 ATF 信息时才能保证这一功能。另一方面,始终能确保 ASAM 数据的完整性和兼容性。

该版本的 ASAM ODS 规范包括两种文件格式:传统的 ATF(ATF/CLA)和基于 XML 的 ATF(ATF / XML)。本章将介绍并解释传统的 ASAM 传输格式(ATF/CLA),第 10 章将描述基于 XML 的 ATF 格式。

9.2 ATF/CLA 文件

9.2.1 ATF/CLA 文件的作用

用于传输或归档数据的文件格式与运行程序时用于存取数据的文件格式相比要满足更多的其它要求。

随着工作格式的 ASAM 文件格式[PROT]的访问性能和编辑能力是至关重要的。因为它通常不是真的有可能在一个文件中插入数据,但将在 ASAM 文件中以结构化存储的方式添加数据,该添加数据写入在单独文件的最后部分。

一种数据传输格式不具有这些要求,因为各文件的读写完全在一步中完成,并没有必要关键访问时间。传输格式必须符合其他要求,例如不同的计算机系统之间的可移植性。

被用户所接受和有效的维持传输文件的结构必须简单,容易理解。因此,易于理解的语法有利于 ATF/CLA 文件和不是基于 ASAM 程序的读写。

从上面提到的事实 ATF/CLA 独立必须是能够以独特的方式存储在 ASAM 数据模型中定义的所有信息。这不仅指结构的信息,而且也指数据本身。

目前的版本的 ATF/CLA 文件的规定在不同的计算机平台传输数据的一个纯 ASCII 格式。在这种情况下

下，要考虑只定义了前 127 个字符的 ASCII 标准。如果突变元音（元音变音）被使用时，给出不兼容的可能性，因为这些字符在不同的系统上不同的解释。

因此，不应该以纯文本格式使用特殊字符。在本章中，你会发现一个传输突变元音的解决方案建议。

对于快速增长的全球网，并计划在以后 ATF/CLA 的版本使用“统一字符编码”。统一字符集是一个世界性认可的 16 位字符集，它实现了一个 ASCII 字符集的超集。欲了解更多信息，请参阅书目 [UNICODE]。

ATF/CLA 文件被看作是一个逻辑文件。这意味着，ATF/CLA 文件可以规律性的分布在一个以上的文件。当分别从实例信息储存和传输的结构信息时特别有趣。

它经常不愿意以 ASCII 格式传输测量数据和海量数据，因为这数据量非常大。在这种情况下，有可能以二进制格式存储海量数据，然而不同的计算机体系结构应该尊重小/大尾声法、浮点格式等。

保持引用子文件的主文件应该得到扩展。“ATF”被确认为 ATF/CLA 文件。所有其他文件可能有个别名和扩展名。

9.2.2 ATF/CLA 文件的结构

ATF/CLA 文件由三种不同的块类型构成，第 9.7 节中将描述它们：

- 文件部分；
- 元素应用部分；
- 实例元素部分。

ATF/CLA 文件以关键字 ATF_FILE 开始，后跟版本号，表示该文件所依据的 ATF/CLA 实际版本。三个主要部分是 FILES、APPLELEM 和 INSTELEM。ATF/CLA 文件的结束标记用关键字 ATF_END 表示。为了跟踪错误，在 ATF/CLA 文件的开始部分应注明产生该文件的应用的节点名、用户名和程序名。

ATF/CLA 文件的目的是显示存在于数据源的数据，并注明出处。因此 ATF/CLA 文件应该只包含元数据和应用于数据源中的数据。数据源中无效的元素和属性不应写入 ATF/CLA 文件，即使他们属于 ASAM ODS 基础模型。

因此，所有来自基础元素的应用元素和在文件中实例化的应用元素，必须在应用元素部分进行描述。除了几个安全实体（见第 9.9 节，“ATF/CLA 文件的安全信息”）在 ASAM ODS 中不允许从基础模型进行实体的实例化。因此除了这些之外，所有其他实体都是应用元素。

9.3 ATF/CLA 的语法描述

9.3.1 概述

语法描述主要是以独特的方式定义 ASAM 传输格式。描述的主题是当生成 ATF/CLA 文件时减少模糊性至最低限度甚至避免。

ATF/CLA 文件的语法在一定程度上遵循 C 和 C++ 的语法。为了提高该文件的可读性，括号的使用被减少到最低限度。为了实现括号的功能，每块关键字都有各自的最终块关键字。

C 语言中，所有“空格”（空格、制表符和换纸）以外的字符串会被忽略或作为分隔符。这就要求 ATF/CLA 文件内的所有逻辑行用分号（;）结束。可能的情况下以该方式把很长的逻辑行分成不同的物

理行。

标识符名称必须遵循 ASAM 标准。大写、小写字母是有区别的。

ATF/CLA 文件语法描述图中使用的正则表达式和结构描述已使用。这些语法图没有机器和图形理论的知识，是容易理解的。

语法图由左到右读即可。首先从分支点最左边的路线进行检查。该循环路线(语法图展示了反应落)用箭头标记终点。这些路线一定不能逆着箭头的方向。

语法图可能是嵌套的需要。语法图名称以 *italic*. 希腊字母大写的是指字符集，以小写希腊字母的是指各自的字符集的字符容量。

关键字始终以大写字母。标识符、字符串和字符不转换为大写，可以单独使用大写和小写字母。

9.3.2 本章所用字符集的说明

本章所用字符集如下。

A = 可打印的字符全集 (32-126 和 128-255)

B = 未允许使用的字符

X = 不可打印的字符全集 (0-31 和 127)

$\Delta = \{0-9\}$ 数字 (48 to 57)

H = $\{0-9, A-F, a-f\}$ 十六进制数字 (48 - 57, 65 - 70, 97 - 102)

$\Lambda = \{A-Z, a-z\}$ 字母 (65-90, 97-122)

N = 非字母数字字符 (32-47, 58-64, 91-96, 123-126)

本章中的非可打印字符如下所示 (不要与字符串的转义序列混淆):

<NUL>, <SOH>, <STX>, <ETX>, <EOT>, <ENQ>, <ACK>, <BEL>,
<BS> , <HT> , <LF> , <VT> , <FF> , <CR> , <SO> , <SI> ,
<DLE>, <DC1>, <DC2>, <DC3>, <DC4>, <NAK>, <SYN>, <ETB>,
<CAN>, , <SUB>, <ESC>, <FS> , <GS> , <RS> , <US> ,
<DEL>

为提高可读性，制表符和空格如下所示: <HT>、<SP>。

9.4 本章使用的符号的说明

(1) 可打印的字符全集

α = 正是字符集 A 中的一个字符。

α^n = 正是字符集 A 中的 n 个字符。

α^+ = 任意数量的字符，至少一个字符。

α^* = 任意的字符数量。“无字符”也适用 ($\alpha^+|\epsilon$)。

(2) 不允许的字符全集

β = 正是字符集 B 中的一个字符。

β^n = 正是字符集 B 中的 n 个字符。

β^+ = 任意数量的字符，至少一个字符。

β^* = 任意的字符数量。“无字符”也适用 ($\beta^+|\epsilon$)。

(3) 非可打印的字符全集

- χ =正是非可打印字符集 χ 中的一个字符。
- χ^n =正是非可打印字符集 χ 中的 n 个字符。
- χ^+ =任意数量的字符，至少一个字符。
- χ^* =任意数量的字符。 “无字符”也适用 ($\chi^+|\epsilon$)。

(4) 数字

- δ =正是字符集 Δ 中的一个数字。
- δ^n =正是字符集 Δ 中的 n 个数字。
- δ^+ =任意数量的数字，至少一个数字。
- δ^* =任意的数字数量。 “无数字”也适用 ($\delta^+|\epsilon$)。

(5) 十六进制数

- η =正是字符集 H 中的一个数字。
- η^n =正是字符集 H 中的 n 个数字。
- η^+ =任意数量的数字，至少一个数字。
- η^* =任意数量的数字。 “无数字”也适用 ($\eta^+|\epsilon$)。

(6) 字母

- λ =正是字符集 Λ 中的一个字母。
- λ^n =正是字符集 Λ 中的 n 个字母。
- λ^+ =任意数量的字母，至少一个字母。
- λ^* =任意字母 ($\lambda^+|\epsilon$)。

(7) 非字母数字字符

- v =正是字符集 N 中的一个字符。
- v^n =正是字符集 N 中的 n 个字符。
- v^+ =任意数量的字符，至少一个字符。
- v^* =任意数量的字符 ($v^+|\epsilon$)。

另外，在上述的说明中 ϵ 代表空集。

9.5 ATF/CLA 保留的关键词

表 162 给出了 ATF/CLA 的关键词，不能用作应用元素和实例的标识符。

表162 ATF/CLA 的关键词

APPLATTR	DT_CFLOAT	IEEEFLOAT4
APPLELEM	DT_DATE	IEEEFLOAT8
ATF_END	DT_DOUBLE	INCLUDE
ATF_FILE	DT_FLOAT	INFINITY
BASEATTR	DT_LONG	INIOFFSET
BASETYPE	DT_LONGLONG	INSTELEM
BLOCKSIZE	DT_SHORT	MANY

CARDINALITY	DT_STRING	NAN
COMPONENT	DT_UNKNOWN	REF_TO
DATATYPE	DESCRIPTION	REF_TYPE
DT_BLOB	ENDAPPLELEM	TRUE
DT_BOOLEAN	ENDFILES	UNDEFINED
DT_BYTE	ENDINSTELEM	VALOFFSETS
DT_BYTESTR	FALSE	VALPERBLOCK
DT_CDOUBLE	FILES	

9.6 ATF/CLA 保留关键词的简要描述

本条给出ATF/CLA保留关键词简要说明。

APPLATTR	应用程序属性的定义。
APPLELEM	应用元素开始的定义。
ATF_END	表示 ATF/CLA 文件结束。进一步信息不会被解析器检查。
ATF_FILE	启动 ATF/CLA 文件，必须存在于主文件的开头。
BASEATTR	规定应用程序属性的基本类型。
BASETYPE	规定应用元素的基本类型。
BLOCKSIZE	二进制数据集的一个块的尺寸。
CARDINALITY	规定一个属性可以赋予多少个值。
COMPONENT	参考外部文件。
DATATYPE	规定应用程序和实例的属性的数据类型。
DESCRIPTION	标记 Blob 的开头。
DT_BLOB	大对象块二进制的数据类型。
DT_BOOLEAN	布尔数据类型，允许的值是 TRUE 和 FALSE 。
DT_BYTE	表示字符常量的数据类型。
DT_BYTESTR	字节流的数据类型。
DT_COMPLEX	表示复杂的单精度浮点数（每个部分的 32 位实部和虚部）的数据类型。
DT_DATE	日期/时间数据类型。
DT_COMPLEX	表示复杂双精度浮点数（ 64 位实部和虚部每个）的数据类型。
DT_DOUBLE	表示双精度浮点数（64 位）的数据类型。
DT_LONG	表示 32 位的整数的数据类型。
DT_LONGLONG	表示 64 位的整数的数据类型。
DT_FLOAT	表示单精度浮点数（32 位）的数据类型。
DT_SHORT	表示 16 位整数的数据类型。
DT_STRING	表示字符串（文字常量）的数据类型。
ENDAPPLELEM	结束应用元素的定义。
ENDFILES	结束 ATF/CLA 的一个逻辑文件的物理文件列表。

ENDINSTELEM	结束实例的定义。
FALSE	布尔数据类型的“False”值。
FILES	列出所有属于逻辑文件的物理文件的文件块。
IEEEFLOAT4	表示 4 字节浮点数字的二进制。
IEEEFLOAT8	表示 8 字节浮点数字的二进制。
INCLUDE	插入指定的文件来替代 Include 指令。
INFINITY	标记值“无穷大”。
INIOFFSET	从二进制文件的开始到第一个块的字节错位。
INSTELEM	开始实例定义。
MANY	元素的任意数。
NAN	不是一个数字。
REF_TO	应用元素定义的参考定义。
REF_TYPE	规定实例属性方面的应用元素。
TRUE	布尔数据类型的“true”值。
UNDEFINED	“没有定义”状态复位值的关键字。
VALOFFSETS	二进制文件中从块开始到值的字节错位。
VALPERBLOCK	在二进制文件中每个块值的数量。

9.6.1 ATF/CLA 保留的关键词的结构

ATF/CLA 文件中的关键字必须以字母开头。后面可以跟任意数量的字母、数字和下划线。关键词不允许出现这里没有提到的其他字符。“关键字”语法见图 67。

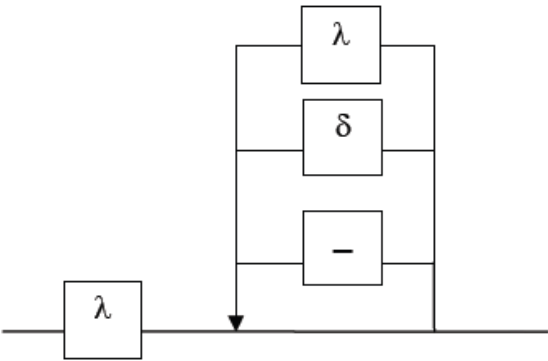


图67 “关键字”语法图

此规则不必执行。它只是 ATF/CLA 关键字的定义规则，因为关键字是从字面列表中提取的。

9.7 ATF/CLA 语法图

9.7.1 空格的结构（空格 WHITESPACE）

回车被定义为<CR><LF>，<CR>和<LF>。请注意：<LF><CR>表示为两个回车（通常不使用）。其在错误的报告中影响加行号。具体见图 68。

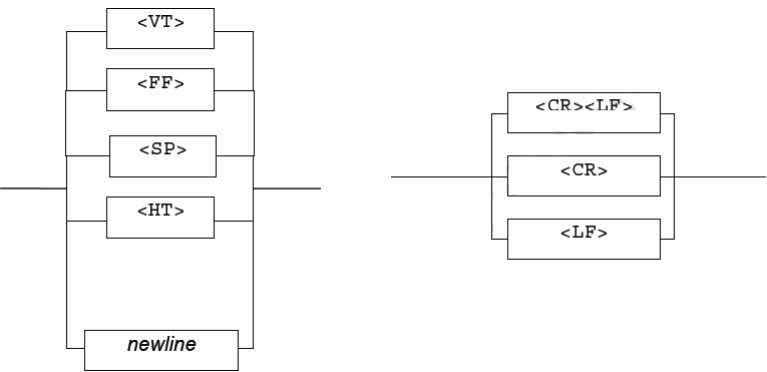


图68 “空格”（左）和“换行”（右）的语法图

语法图空格元素在字符串（“...”）或字符常量（'...'）以外的任何位置被自动忽略。请在第 9.3.1 章概述中找到更多信息。空格如下所示：

- 可能包含一个空白；
- 至少包含一个空白。

9.7.2 分隔符的结构（分隔符 *SEPARATOR*）

“分隔符”语法图见图69。

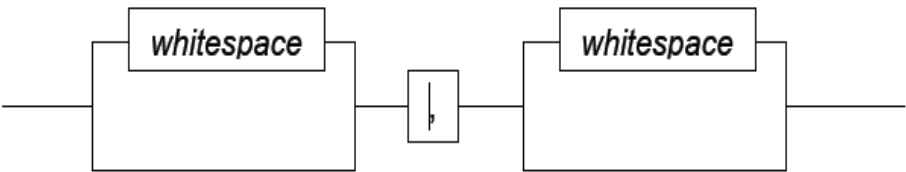


图69 “分隔符”语法图

分隔符如下所示：

- 可以包括一个分隔符；
- 至少包含一个分隔符。

9.7.3 注释的结构（注释 *COMMENT*）

当（物理）行注释结束时，注释包含于“/*”和“*/”或始于“//”。以下规则适用于：

- 注释不能嵌套；
 - 标记字符串或字符常量的注释不解释；
 - “//”之后的“/*”和“*/”没有任何意义；
 - “/*”和“*/”所包围注释的“//”没有意义；
 - 被列为注释分隔符的含义例如序列“文本/*...*/恒定”将解释为两个标识符由隔板分隔。
- “注释”语法图见图 70。

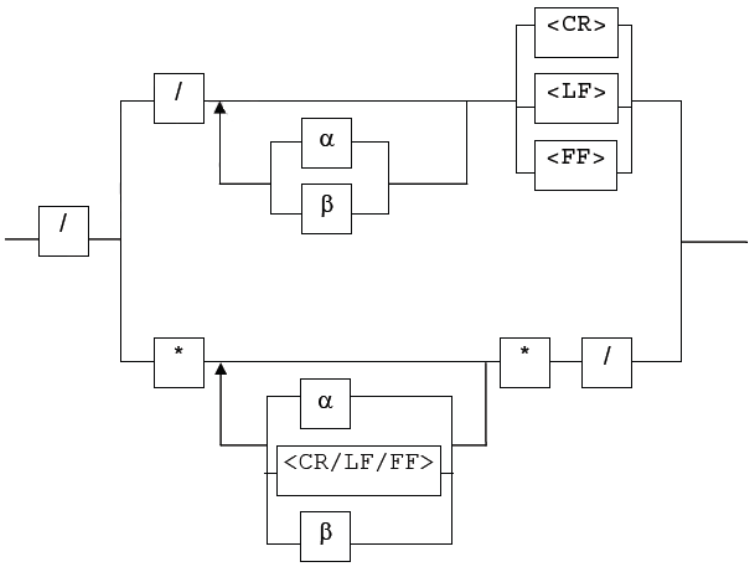


图70 “注释”语法图

9.7.4 转义序列的结构（转义 ESCAPE）

转义序列允许使用于字符串、字符常量和标识符。读取过程中这些序列将被转换为一个字节。转义序列具有以下涵义：

- 在字符串和字符常量插入的非打印字符（如回车换行、换页等）；
- 封装在标识符中的特殊字符（如冒号、分号、括号等），这些特殊字符可能会导致 ATF/CLA 文件的语法含糊不清的地方；
- 7 位 ASCII 字符集以外的独特字符的转换，以不同的方式在不同的机器（例如德语的元音字母变音符号）将被解释。

由于转义序列影响 ATF/CLA 文件的可读性，它们的使用应尽可能地减少。

如果转义序列已经是另一个转义序列的说明结果，那么转义序列的说明将不能替换。这易于构建说明，因为读指针不需要向后定位。“转义”语法图见图 71。表 163 给出了元音编码。

转义序列由 “\” 开始。如果在字符串或字符常量使用了该字符，那必须增加为：“\\”。

以下可打印字符的转义序列允许使用于字符串、字符常量和 ATF/CLA 文件的标识符：

- \<sp>——空格（在标识符内需要）；
- \v——所有非字母数字字符，例如在一个字符串；
- \0xdd——一个任意可打印字符的十六进制表示形式；

不再需要，因为扩展字符集允许用于字符串中（包括了解旧文件和文件）：

- \<ae>—— Umlaut ä;
- \<Ae>—— Umlaut Ä;
- \<oe>—— Umlaut ö;
- \<Oe>—— Umlaut Ö;
- \<ue>—— Umlaut ü;
- \<Ue>—— Umlaut Ü;

- \<sz>——特殊字符 β;
- \<^0>——度 (e.g. °Celsius) 。

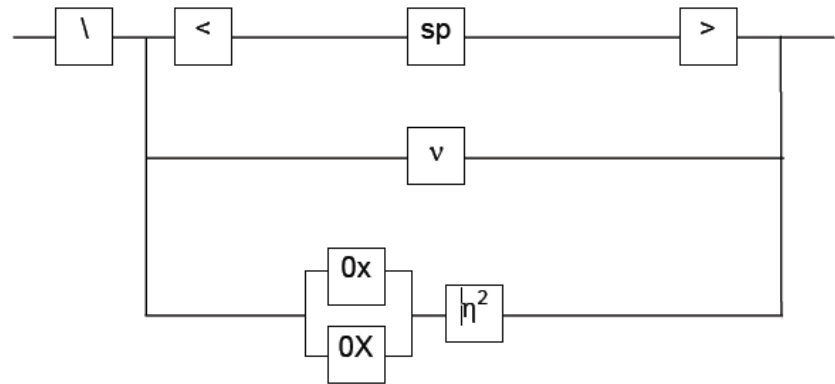


图71 “转义”语法图

表163 元音变音编码（这里仅供参考）

	ISO
ä	228
Ä	196
ö	246
Ö	214
ü	252
Ü	220
ß	223
°	176

表 164 附加的非打印字符的转义序列被允许在字符串中，字符常量和 ATF/CLA 文件的标识符：

表164 非打印字符的转义序列

Sequence	Code	Description	
\b	8	Backspace	<BS>
\f	12	Form feed	<FF>
\n	10	New line	<LF>
\r	13	Carriage return	<CR>
\t	9	Tab	<HT>
\u		Reserved for Unicode extensions	
\v	11	Vertical Tab	<VT>

通过使用正常的转义序列以完成所有其他的替代表示。

对于以数字（0 ... 9）开头的标识符使用 0-9 的替代表示，见图 72。因此，价值<NUL>仅可以用十六进制表示\0×00。

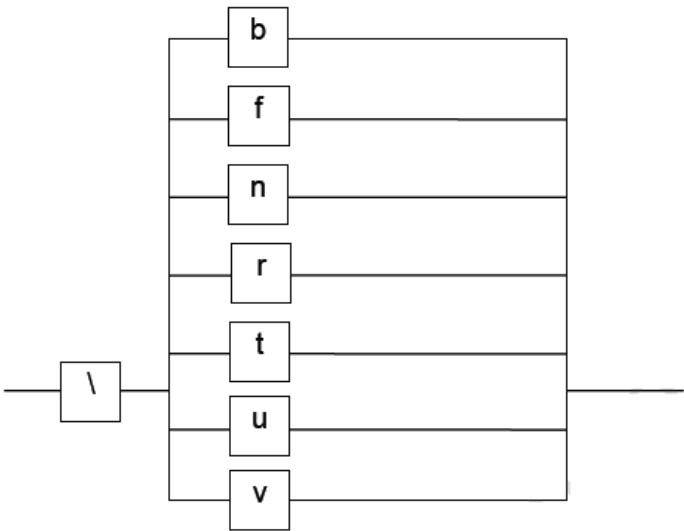


图72 “转义非打印字符”语法图

因为 ATF/CLA 未来的版本可使用 16 位字符表示，应注意在实施过程中。在阅读 ATF/CLA 文件时应检查 ATF/CLA 版本号是否不高于阅读计划的实施过程中所使用的版本。

9.7.5 整数的结构（整数 *INTEGER*）

ATF/CLA 的整数可以使用 8（字节）、16（为 int2 ）、32（int4 的）和 64 位（int8 的）（1、2 、4 和 8 字节）表示。使用长度为 64 位时，必须考虑到今天的计算机有很多只适用于 32 位。这种类型的机器不可能的仿真处理 64 位值，这将导致一个错误消息。

字节数据类型通常被解释为无符号。在识别的字节数据解析器应该能够接受负数（-1 是等于 255 等）。

ATF/CLA 不会有任何的八进制表示，因为八进制值是易于转换为十六进制值，见图 73。忽略前导零。

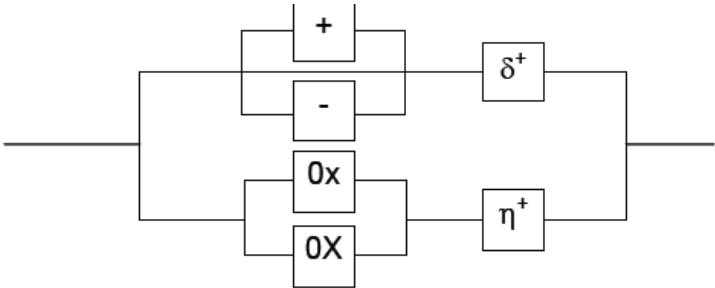


图73 “整数”语法图

9.7.6 浮点数结构 (REAL)

ATF/CLA 用 32 位表示单精度浮点的数字 (real4)，64 位表示双精度浮点 (real8)。确切的数字范围取决于机器的体系结构和机师的话，尾数和指数。

首选 ATF/CLA 格式是“《二进制浮点运算 IEEE 标准》(ANSI / IEEE 754)”。本节介绍浮点数据如何存储在内存中。它总结了不同的 IEEE 存储格式的精度和范围。

浮点格式是一种数据结构，它规定浮点数的精度范围、这些字段的布局 and 它们的算术解释。

浮点存储格式规定浮点格式如何存储于内存中。IEEE 标准定义该格式，但它留给实施者选择存储格式。

汇编语言软件有时会依赖于使用的存储格式，但更高级别的语言通常仅处理浮点数据类型的语言概念。这些类型相对于 IEEE 格式，在不同的高层次的语言有不同的名称。IEEE 754 标准明确规定了单、双浮点格式，它为这两种基本格式定义了一类扩展格式。

以下各节详细描述了用于 IEEE 浮点格式的三种存储格式。

(1) 单精度格式

IEEE 单精度格式由三个字段组成：一个 23 位的小数 f；8 位偏置指数 e；1 位符号 s。这些字段以 32 位连续存储。0: 22 包含 23 位小数 f，0 是最有效的分数位，22 位是最重要的；位 23: 30 包含 8 位偏置指数 e，23 是最有效的分数位，30 位是最重要的；最高位 31 位包含符号位 s。

表 165 显示的三个组成字段 s、e 和 f，在一方面，和其他单格式的位模式所表示的值的值之间的对应关系；u 意味着不关心，即，显示字段的值与单精度格式中的特定位模式的值的测定是不相关的。

表165 IEEE 单精度格式的位模式表示值

Single-Format Bit Pattern	Value
$0 < e < 255$	$(-1)^s \times 2^{e-127} \times 1.f$ (normal numbers)
$e = 0; f \neq 0$ (at least one bit in f is nonzero)	$(-1)^s \times 2^{-126} \times 0.f$ (subnormal numbers)
$e = 0; f = 0$ (all bits in f are zero)	$(-1)^s \times 0.0$ (signed zero)
$s = 0; e = 255; f = 0$ (all bits in f are zero)	+Infinity (positive infinity)
$s = 1; e = 255; f = 0$ (all bits in f are zero)	-Infinity (negative infinity)
$s = u; e = 255; f \neq 0$ (at least one bit in f is nonzero)	NaN (Not-a-Number)

请注意，当 $e < 255$ 时，分配给的单个格式位模式的值形成的插入二进制基数点紧接分数的最显著位的左侧，并插入一个隐含位立即在二进制小数点的左边，从而在二进制位置记数法表示的混合数目（整数加分数，其特征是， $0 \leq \text{分数} < 1$ ）。

由此形成的混合数被称为单精度格式有效。如此命名是因为它的价值没有明确的给出了单格式位模式，但所隐含的偏置指数字段的值的隐含位。

有效数字的前导位（二进制点位到左）的一个正常的数字是 1，而次正规数的有效数字的前导位为 0。在 IEEE 754 中，单格式次正规数称为单一格式的非规格化数。

23 位小数加上隐含前导有效位提供 24 位精度，单格式正规数。

重要的位模式在单存储格式的下表中所示的实施例。最大正正规数是 IEEE 单精度格式的最大有限数表示的。最小正次正规数是最小的正数表示的 IEEE 单精度格式。最小正正规数通常称为下溢阈值。（最大和最小正规数和次正规数的十进制值是近似的；它们是正确的显示的数字的数量。）

单精度存储格式的位模式及其 IEEE 值见表 166。

表166 单精度存储格式的位模式及其 IEEE 值

Common Name	Bit Pattern (Hex)	Decimal Value
+ 0	00000000	0.0
- 0	80000000	-0.0
1	3f800000	1.0
2	40000000	2.0
Maximum normal number	7f7fffff	3.40282347e+38
Minimum positive normal number	00800000	1.17549435e-38
Maximum subnormal number	007fffff	1.17549421e-38
Minimum positive subnormal number	00000001	1.40129846e-45
+ ∞	7f800000	Infinity
- ∞	ff800000	-Infinity
Not-a-Number	7fc00000	NaN

任何满足的定义为 NaN 的众多位模式，可以表示为 NaN（非数字）。在上面的表中显示的 NaN 十六进制值只是可用于表示 NaN 的众多位模式之一。

(2) 双精度格式

IEEE 双精度格式由三个字段组成：一个 52 位小数 **f**、11 位偏置指数 **e** 和 1 位符号 **s**。这些字段连续存储在两个连续地址的 32 位字。

在 SPARC 架构，较高地址的 32 位字包含分数的最低有效位 32 位，而在英特尔和 PowerPC 架构下低地址的 32 位字包含分数的最低有效位 32 位。

如果我们用 **f[31:0]** 表示分数的最低有效位 32 位，0 位是整个分数的最低有效位，31 位是 32 位有效分数的最高有效位。

在其他 32 位字中，0:19 位包含分数的 20 最高有效位 **f[51:32]**，0 位是 20 最高有效位分数的最低有效位，19 是整个分数的最高有效位；20: 30 包含 11 位偏置指数 **e**，20 位是偏置指数的最低有效位，30 是最高有效位；最高顺序位包含符号位 **s**。

这三个字段中的位模式的值决定整个位模式所表示的值。

一方面，下表显示了三个组成字段中位的值之间的对应关系，另一方面，值由双精度格式的位模式所表示；**u** 意味着不关心，因为指定字段的值与特定位模式中的双格式是无关紧要。

IEEE 双精度格式位模式表示的值见表 167。

表167 IEEE 双精度格式位模式表示的值

XXX Double-Format Bit Pattern	Value
0 < e < 2047	$(-1)^s \times 2^{e-1023} \times 1.f$ (normal numbers)
e = 0; f != 0 (at least one bit in f is nonzero)	$(-1)^s \times 2^{-1022} \times 0.f$ (subnormal numbers)
e = 0; f = 0 (all bits in f are zero)	$(-1)^s \times 0.0$ (signed zero)
s = 0; e = 2047; f = 0 (all bits in f are zero)	+Infinity (positive infinity)
s = 1; e = 2047; f = 0 (all bits in f are zero)	-Infinity (negative infinity)
s = u; e = 2047; f != 0 (at least one bit in f is nonzero)	NaN (Not-a-Number)

请注意，当 **e < 2047**，在分数最重要位的左侧立即插入二进制小数点，并在二进制点的左侧立即插入一个隐含位，以形成分配给双精度格式位图案的值。由此形成的数字被称为尾数。如此命名是因为在双格式的位模式没有明确给出它的值，但隐含了偏置指数字段的值。

对于双精度格式，一个正常的数字和一个次正规数之间的差异是，该有效的前导位（位在二进制点的左侧）的正常数字是 **1**，而一个有效的前导位的非正规数为 **0**。在 IEEE 754 中，双精度格式次正规数称为双精度格式非正规数据。

52 位小数加上隐含前导有效位提供了双格式正规的 53 位精度。

双存储格式的重要位模式中的示例如下表所示。在第二列中的位模式以 2 个 8 位的十六进制数显示。对于 SPARC 体系结构，左边的一个是较低地址的 32 位字的值，右侧是较高地址的 32 位字的值，而对于 Intel 和 PowerPC 体系结构，左侧是高地址的字，右侧是较低地址的字。最大正正规数是在 IEEE 双精度格式表示的最大有限数。

最小正次正规数是 IEEE 双精度格式表示的最小正数。最小正正规数通常称为下溢阈值。（最大和最小的正常和次正规数的十进制值是近似的；对于显示的数字的数量它们是正确的。）

双精度存储格式的位模式及其 IEEE 值见表 168。

表168 双精度存储格式的位模式及其 IEEE 值

Common Name	Bit Pattern (Hex)	Decimal Value
+ 0	00000000 00000000	0.0
- 0	80000000 00000000	-0.0
1	3ff00000 00000000	1.0
2	40000000 00000000	2.0
Max normal number	7fefffff ffffffff	1.7976931348623157e+308
Min positive normal number	00100000 00000000	2.2250738585072014e-308
Max subnormal number	000fffff ffffffff	2.2250738585072009e-308
Min positive subnormal number	00000000 00000001	4.9406564584124654e-324
+ ∞	7ff00000 00000000	Infinity
- ∞	fff00000 00000000	-Infinity
Not-a-Number	7ff80000 00000000	NaN

NaN（非数字）可以表示满足 NaN 定义的任何众多位模式。以表 110 中所示的 NaN 十六进制值仅仅是用于表示 NaN 的众多位模式之一。“实数”语法图见图 74。

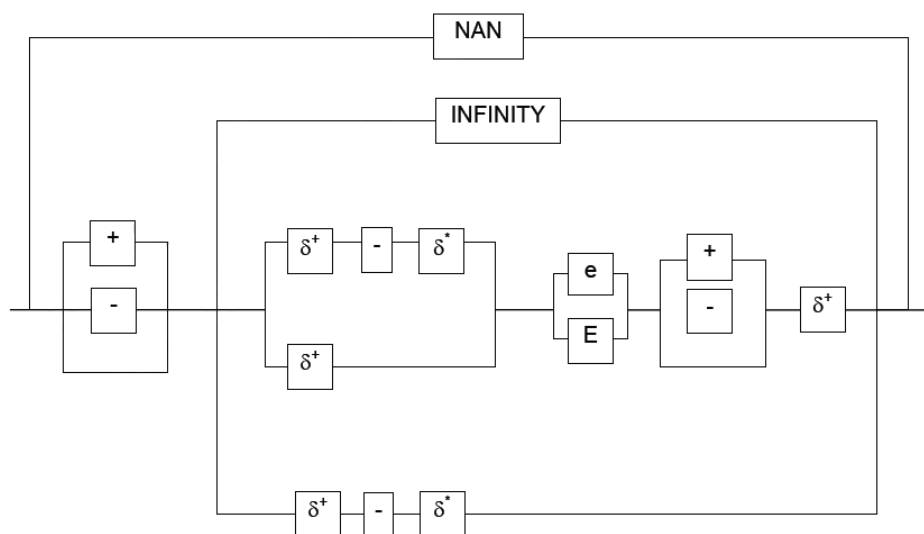


图74 “实数”语法图

9.7.7 复数的结构 (COMPLEX)

实数在图表复数应用的处理方式见图 75。在一个复杂的序列的项的数目是一个真正的序列的两倍。
在序列中的顺序是：

- 第一个实部；
- 第一虚部；
- 第二个实数部分；
- 第二个虚部；
- 第三个实数部分.....。

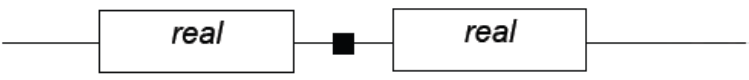


图75 “复数”语法图

9.7.8 字符常量的结构 (CHARACTER)

括在单引号字符常量可能只包含字符。对于非打印字符，使用通常 C 替代表示（见转义序列）。
“字符”语法图见图 76。

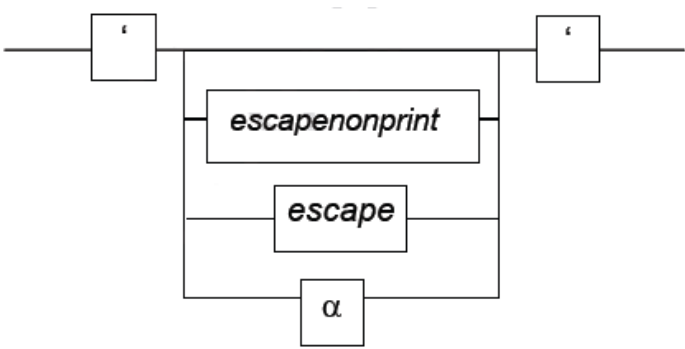


图76 “字符”语法图

值得考虑的是字符常量是 8 位整数值（字节）的另一种表示。内部表示是相同的。

9.7.9 字符串的结构 (STRING)

字符串括在双引号内并包含表示字符的任意数量。也允许是一个空字符串。文字常量“...” + “...”
在未来的版本中计划，但尚未实施。字符串语法图见图 77。

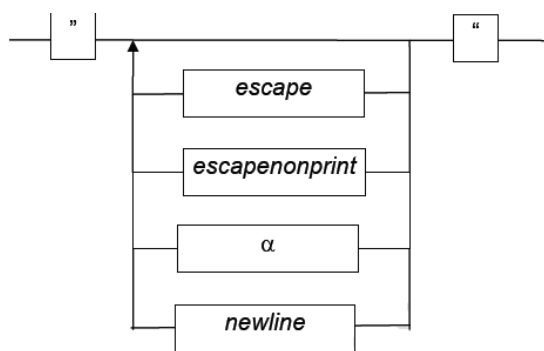


图77 “字符串”语法图

在网络中随时使用转义序列。这些序列用一个反斜杠（\）开始。在未来的版本中将定义新的元素（例如，\A\C）。当在字符串中使用反斜杠时，应该写为“\\”来兼容未来的版本（例如“\自动”变成“\\汽车”）。

请注意，一个字符串内的双引号必须前面加一个反斜杠，否则将被视为结束字符串。

请注意，当使用“.CR.”和“.LF.”使得一个超过几行的长字符串能够容易地编辑和印刷时，属于字符串的换行符必须被编码为\n。

9.7.10 字节流的结构（BYTESTREAM）

使用特殊内容如字符串以表示字节流。只有 16 个十六进制字符允许加上换行以编辑和打印 ATF / CLA 文件。“字节流”语法图见图 78。

请注意，不要写入字节流的长度。这最终需要重新计算读取的字节流长度。

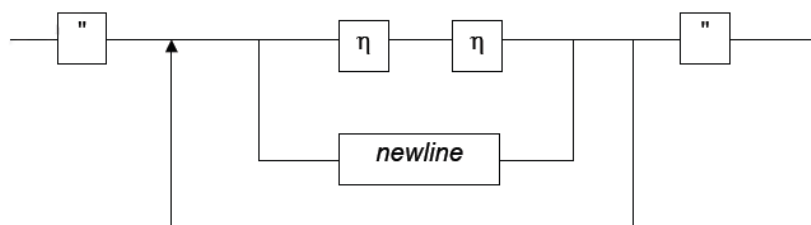


图78 “字节流”语法图

9.7.11 标识符的结构（IDENTIFIER）

标识符姓名的定义也适用 ASAM 标准规则。可能导致语法歧义的标识符的特殊字符使用反斜杠开始。“识别符”语法见图 79。

因为特殊字符的标识符会影响 ATF/CLA 文件的可读性，所以不应该使用该类型的标识符。

标识符名称必须以字母、下划线或转义序列开头。名称的长度并不限于 ATF/CLA 文件。

第 9.5 节规定不得使用保留字作为标识符的列表。然而，标识符和保留字是区分大小写，所有的保留字用大写字母。这为应用元素的名称、属性和关系提供了极大的自由。

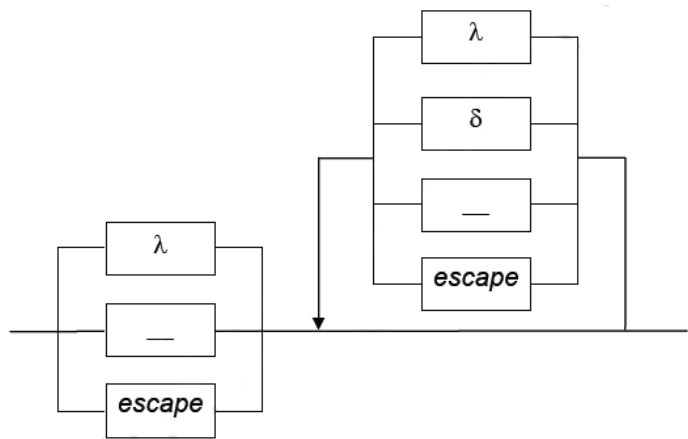


图79 “识别符”语法图

9.7.12 文件名的结构（FILENAME）

ATF/CLA 文件参考的路径和文件名必须以广泛的计算机体系结构运输和解释。因此，一般的定义的路径和文件名的格式是至关重要的，它可以很容易地解释，并转换为目标机器。

在全球范围内的互联网使用的统一资源定位器（URL）已经在不同的平台上应用。通过 URL 的命名约定，将有可能在未来的 ATF/CLA 版本让 ATF/CLA 文件分布于网络。

URL 为目标服务器、计算机的目标系统或服务器名称、端口、目录路径和文件名描述了一个协议。该 URL 具有以下结构：

access_method://server_name[:port]/directory_path/filename

示例：列示例表示一台远程机器上资源的 URL（万维网服务器的主列表）。

利用 HTTP 协议可以定位名称为 www.w3.org 的 Web 服务器，通过标准端口，可以找到 /hypertext/DataSources/WWW 目录，在该目录下包含一份超文本文档 Geographical.html。

网址是 <http://www.w3.org/hypertext/DataSources/WWW/Geographical.html>。

因特网中的每个文件都有 URL 唯一地址。如果协议引用该目录的内容，URL 则以斜线“/”结束。当引用目录中的文件时不用带斜线。

在 URL 中最常用的协议有：

- http: 超文本传输协议（万维网（WWW）的协议）
- file: URL 只包含一个文件名，而不是地址。
- ftp: 进行文件传输的文件传输协议。
- gopher: 进行文件传输的和在 Gopher 系统导航的 Gopher 协议
- mailto: 给收件人发送电子邮件的 Internet 邮件协议。
- news: 轮询文章或新闻组的互联网新闻协议。
- rlogin: 定义机器的远程登录。
- telnet: 给定义的机器通过互联网启动一个终端会话。

网址为上面所描述的被称为绝对 URL，因为它们包含的文件的完整路径名。简化的 URL 格式被称为相对 URL。这种简化的格式作为实际文件引用在同一服务器上的其他文件。相对 URL，ATF/CLA 文

件可以直接引用物理组件而不引用服务器。在目前的 ATF/CLA 版本只支持相对 URL。

绝对 URL 引用在 Internet 上的资源，包括当地的资源。由于后来解释的原因，相对 URL 更适合本地资源。

作为包含 URL 的 ATF/CLA 文件，相对 URL 意味着相同的访问方法、相同的服务器名和相同的目录路径。它描述了与实际 ATF/CLA 文件相关的，ATF/CLA 文件的组件的相对位置关系。

示例：../../matrix_xyz.dat。

相对 URL 可以移动 ATF/CLA 目录结构到在目录树中的任何位置。相对 URL 不必引用实际文件目录，但可能包含 ATF/CLA 文件的根目录的相对参考。这种相对 URL 的变体用一个斜杠 (/) 开始。它看起来像一个绝对的 URL，但不包含访问方法和服务器名称。

两个方向上正确的路径转换和插入正确的分隔符是各自的目标系统读写程序的 ATF/CLA 责任。

如果目标系统（如 DOS 文件名 8.3 的长度）的文件名太长，缩写文件名将遵循以下规则：文件名内的第一个冒号后允许最多 3 个字符。没有这些字符可能是一个冒号。如果有超过 3 个字符或另一个冒号，其余的将被截断。不允许的字符被波浪线替换（破折号）。

冒号前面的字符数进行计数，并截断为 8 个字符。如果该情况文件名不是唯一，则可使用 Windows 95 算法（截断为 6 个字符，使用后 2 个字符进行编号）。

这种情况下，最有可能手动编辑 ATF/CLA 文件。虽然所有需要的文件都集中在 FILES 块，手工编辑应该不太费劲。

为了使文件传输到 UNIX 更容易文件名总是使用小写，操作系统甚至是不区分大小写 (VAX / VMS，MS-WINDOWS)。大写字母的文件名只用于 UNIX 计算机上的路径和文件名应该是大写的情况。

“文件名”语法见图 80。

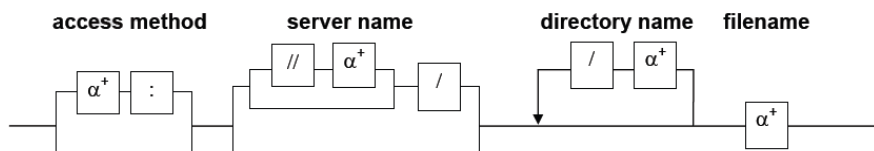


图80 “文件名”语法图

9.7.13 预定义的数据类型 (DATATYPE)

应用和实例属性的数据类型可以自由选择下列预定义数据类型中的 DATATYPE n:

DT_BOOLEAN	逻辑值 (TRUE / FALSE)。
DT_BYTE	单字节 (8 位)，没有符号位。
DT_SHORT	16 位整数，包括一个符号位。
DT_LONG	32 位整数，包括一个符号位。
DT_LONGLONG	64 位整数，包括一个符号位。
DT_FLOAT	32 位浮点数。
DT_DOUBLE	64 位浮点数。
DT_COMPLEX	复杂的 32 位浮点数部分。
DT_DCOMPLEX	复杂的 64 位浮点数部分。

DT_STRING	双引号括起来的字符串。
DT_ENUM	枚举，表示一组有限的带有关键字和包含在方括号中的字符串。
DT_BYTESTR	字节流：二进制数据
DT_BLOB	BinaryLargeObjectBlock：命名的字节流
DT_DATEASAM	日期类型（字符串表示）
DT_UNKNOWN	来自 AoLocalColumn 的特殊类型的基础属性值
DT_EXTERNALREFERENCE	外部参考：三个字符串、每个用双引号括起来或由一个逗号分开： （描述、mimeType、位置）

数据类型的内部表示依赖于所使用的机器类型、数据存储系统和用户的要求。ATF/CLA 文件的数据类型 DT_DATE 以引号括起来的普通字符串进行存储。

适用于数据库或计算的格式转换是相应应用程序的责任。

“数据类型”语法见图 81。

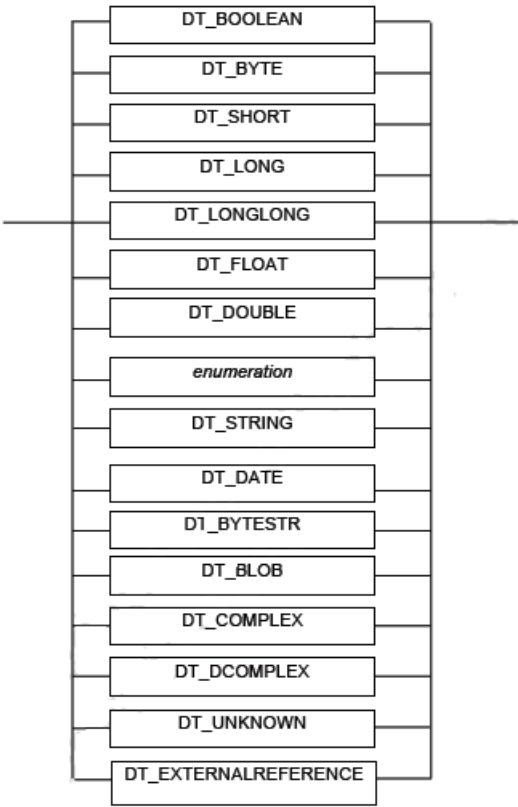


图81 “数据类型”语法图

注：DT_UNKNOWN只允许在AoLocalColumn的基本属性“值”。

9.7.14 枚举的结构

关键字 DT_ENUM 是用来描述一个应用程序模型的枚举类型。枚举含有允许字符串的有限规定。该规定写在关键字 DT_ENUM 的后面，并用方括号括起来。字符串应当遵循标识符规则，并以逗号分

开。“枚举”语法图见图 82。

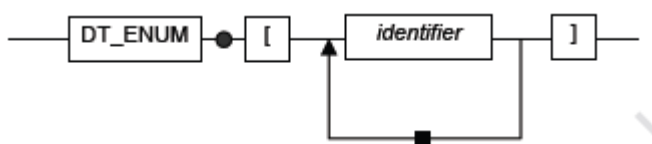


图82 “枚举”语法图

在实例中，包含枚举的实例属性可能有两种的表达方式：

(1) 表示为字符串形式

该字符串必须包含括号之间的 APPLATTR 定义的标识符之一。

示例：

```
APPLELEM ...
APPLATTR exampleAttr, DATATYPE DT_ENUM [First, Second, Third];
...
INSTELEM ...
exampleAttr = "Third"
```

(2) 表示为整数形式

该整数包含一个字符串列表空的索引，由括号之间的 APPLATTR 进行定义。

示例：

```
APPLELEM ...
APPLATTR exampleAttr, DATATYPE DT_ENUM [First, Second, Third];
...
INSTELEM ...
exampleAttr = 2; /*meaning: "Third"*/
```

9.7.15 ATF/CLA 版本标识符的结构 (VERSION)

关键字 ATF_FILE 启动 ATF /CLA 文件，且必须设置在各自文件的开头。此关键字和该文件的生成都要依据 ATF /CLA 标准的版本号。此行之前的所有信息都将被忽略，确认关键字“ATF_FILE”后进行解释。阅读程序必须能够忽略关键字该行之前的这行。不存储关键字之前的行。

在 ATF_FILE 行之后插入重要的评论。这是由于应用程序不会总是在 ATF_FILE 行之前复制行。

通过 E-mail 发送文件时，相应的文件起始部分插入路由信息。在这种情况下，用编辑器删除这些信息的接收者是不方便的。“版本”语法见图 83。

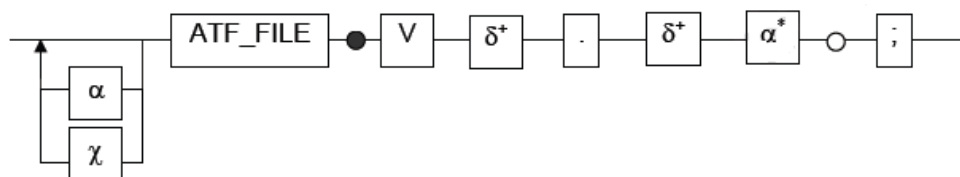


图83 “版本”语法图

示例：这是 ASAM 文件的开始部分，在关键字 ATF_FILE 之前的所有信息都被忽略（即使它们是说明信息）。

```
ATF_FILE V1.4.1;
// Created Fri May 09 16:28:45 2003
// by user GUEST on host ASAM with program APPLICATION1
```

ATF/CLA 文件何时何地创建的信息是一个可选注释，并且对于跟踪问题和达到目的证明是有益的。

9.7.16 文件块的结构 (FILES)

文件块列出所有属于 ATF/CLA 文件的物理文件。这些文件包括 INCLUDE 语句（见下一节）或二进制组件，用来表示大量的像在本本地列的值（请参阅组件部分），这些文件所使用的文件。ATF 文件（包含“ATF_FILE”和“ATF_END”的文件、语句）加上 FILES 块列出的所有文件，FILES 块可以被视为一个逻辑文件。FILES 块是可选的，因为并不是每一个逻辑文件机械性地分布在多个文件中。

在第一个 APPELEM 或 INSTELEM 指令之前必须插入块，该块必须引用有效 ASAM 文件。该文件的顺序是任意的。在 ATF/CLA 文件内具有唯一一个文件块。因此，它不必位于包含文件。

除了在 FILES 块中进行定义属于 ATF/CLA 的逻辑文件，阅读程序必须确保没有文件。属于 ATF/CLA 逻辑文件的所有物理文件一定在 FILES 块 COMPONENT 语句中必须声明。请记住，这样的组件声明的文件不一定是二进制组件。

任何组件数量的定义是允许的。块内组件的顺序是任意的。在下面例子中的缺口增强了可读性，而不是强制性。

然后，组件名称可以用于 INCLUDE 指令以及数据的 COMPONENT。

“文件”语法见图 84。

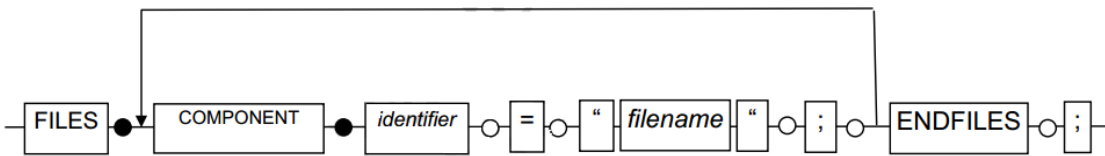


图84 “文件”语法图

FILES 块中允许组件定义。文件名是：

- 简单文件名（作为 ATF/CLA 文件存储在同一个目录中的文件）；
- 绝对路径名（小心处理）；
- 相对路径名（存储于 ATF/CLA 文件的相对目录）。

示例：

```
FILES
// Reference to more physical files
// ... with the internal name "component_1"
```

```

COMPONENT file_1 = "../data/k1.dat";
COMPONENT file_2 = "atf_struct.inc";
<...>
ENDFILES;

```

9.7.17 指令 INCLUDE 的结构 (INCLUDE)

在当前处理的文件中插入含有指令 **INCLUDE** 的文件，见图 85。该指令可以被定位在该文件中，而不是一个或多个 **instelem** 或 **applelem** 结构组件。唯一的条件是在包含文件中所包含的信息必须在相应的位置插入语法上是正确的。

立即将引用文件的内容替换 **INCLUDE** 指令。每次都允许在主要文件的不同位置插入一个文件。

通常情况下插入文件包含别名定义、结构信息、实例或数据。实例要求至少在一个 **ATF/CLA** 文件内并应被定义一次，而数据必须是进行分组的数据。对此类文件的一个例子是一个测量的数量目录，包含了所有的尺寸和大小，需要在一个特殊的环境。

注：请注意，这些文件是ASCII文件，他们的使用与二进制组件文件使用是不同的。

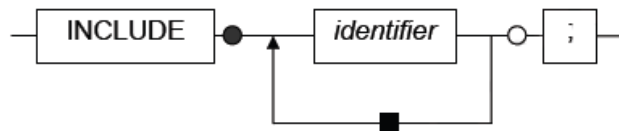


图85 “包括”语法图

示例：

```

FILES

    // Reference to more physical files
    // ... with the internal name "component_1"
    COMPONENT file_1 = "../data/k1.dat";
    COMPONENT file_2 = "atf_struct.inc";
    //...

ENDFILES;

INCLUDE file_1;
INCLUDE file_2, file_1;

```

9.7.18 参考的单位 (PHYS_UNIT)

实例属性的物理单元是由基础元素“单元”所得出的信息定义的。图 86 对“phys_unit”语法进行了说明。在没有引号和括号括起来这种情况下编写单位名称。

示例：

```
Power = 123 [kW]
```

Acceleration = 4.34 [m/s^2]

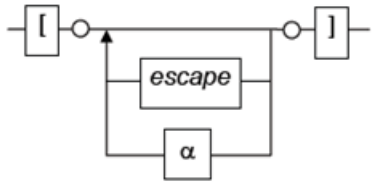


图86 “phys_unit” 语法图

注：请注意，单位字符串不能包含方括号，它们必须回避编码。
在分配给各个单元的属性后面，ATF/CLA 文件内可以设置这样一个定义。

9. 7. 19 数据属性值 (DATAVALUE)

实例属性（参见“实例元素”部分）的数据值具有图 87 规定的格式。这还包括实例的属性值。应用元素（参见“应用元素”）的属性规格确定，何种类型可以使用。

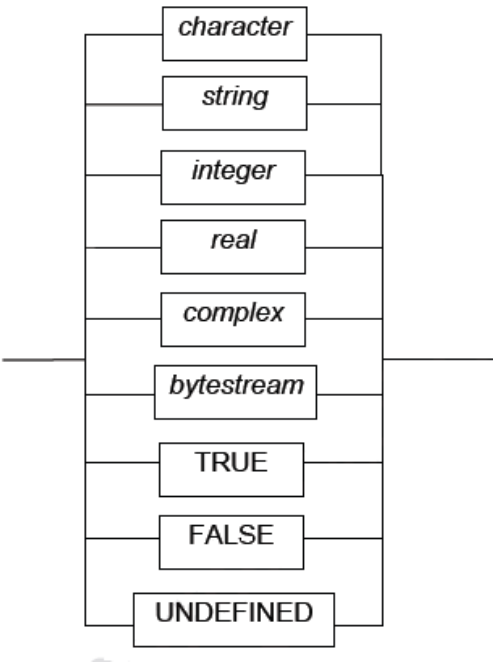


图87 “数据值” 语法图

9. 7. 20 应用元素的结构 (APPLELEM)

APPLELEM 结构允许规定从标准基础元素得到的应用元素。因此，关键字 **BASETYPE** 要强制性的遵循有效的基础元素名称。源自基本属性或特定应用程序属性的属性可以被指定。某些属性可能引用其他应用元素。如果该属性是一个引用，被引用的应用元素则由关键字 **REF_TO** 指定。如果该属性不从基本属性得到的，则必须给出关键字 **REF_TO**。

在基础模型中的实体和属性名称是不区分大小写的，因此可以任何格式写。基本属性名称不遵循标

ASAM 数据模型的应用元素与其他应用元素相关。例如一个测试可以具有几个测量以及一个测试可以具有一个测量。

ATF/CLA 文件的这种关系属性由关键字 **REF_TO** 宣布。实体“测量”包含例如属性“**BASEATTR measurement_quantities**”。

示例:

```
APPLELEM SpecialMeasurement, BASETYPE AoMeasurement

  APPLATTR ID, BASEATTR id;

  APPLATTR Name, BASEATTR name;

  APPLATTR myMeasurementQuantities,
    BASEATTR measurement_quantities,
    REF_TO MeasurementQuantity,
    CARDINALITY 0,MANY;

ENDAPPLELEM;

APPLELEM MeasurementQuantity, BASETYPE AoMeasurementQuantity

  APPLATTR ID, BASEATTR id;

  APPLATTR Name, BASEATTR name;

  //...

ENDAPPLELEM;
```

9.7.21 实例元素的结构 (INSTELEM)

实例元素用于定义实例。**ASAM** 数据模型实例涉及到其他实例。例如测试可以具有多个测量值，另一方面测量可能会告诉它属于哪个测试。

属性分配的顺序是任意的。没有分配值的应用程序属性的默认值是 **UNDEFINED** 。在 **ATF/CLA** 文件中也可以省略其属性，而不是提供一个 **UNDEFINED** 值的属性。

可以定义和分配一个实例的任意数量的新实例属性,必须给出这些实例属性的数类型。

“实例元素”语法见图 91。“data_attribute_values”语法见图 92。“reference_attribute_values”语法见图 93。

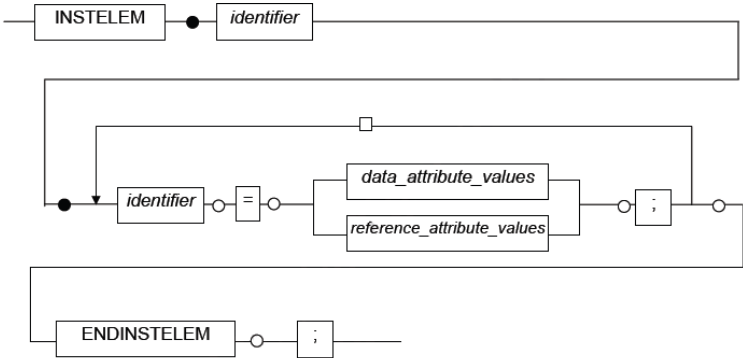


图91 “实例元素” 语法图

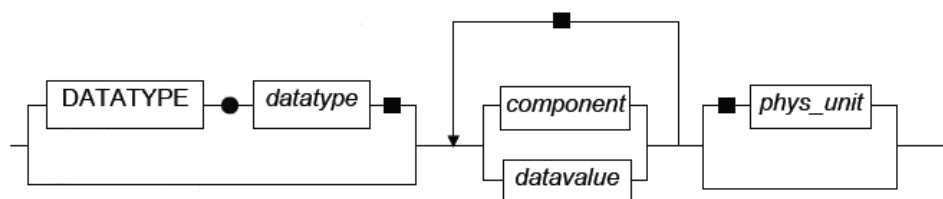


图92 “data_attribute_values” 语法图

图中的圆代表空格，方块代表分隔符；空心意味着可选的，实心意味着强制的。

注 1：定义为 DT_EXTERNALREFERENCE 的属性值用 3 个连续的 datavalue 项表示，作为字符串（双引号）给出，并用逗号彼此分隔。因此，“datavalue”的数量是属性值数量的 3 倍。

注 2：AoLocalColumn 实例的属性“值”以特殊的方式进行处理：AoLocalColumn 的基础元素具有属性“值”特定类型的数据序列。此序列不得作为一个单独的对象出现，因此没有给出属性“值”的参考序列。而不是在此属性中写入序列中包含的 DIRECTLY 值。用带 datatype = DT_... 的 DATATYPEdatatype 表示序列类型（请注意，在 5.7.22 部分示例 1 中给出这种处理方式的示例）。

注 3：DATATYPEdatatype 结构只有在以下两种情况下出现：

- 1) 如果提供的值指定的应用程序属性来自 base 属性'值'（本地序列），该结构是必需的。
- 2) 如果所提供的值指定一个实属性，该结构是必需的（其他这种属性的数据类型未知）。

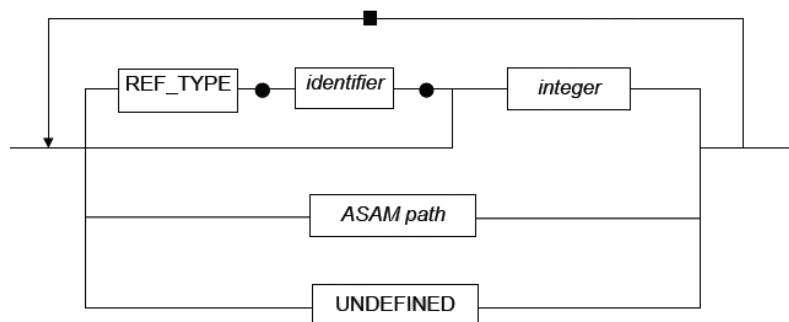


图93 “reference_attribute_values” 语法图

示例：

```
INSTELEM measurement
  //<...>
  ID = 47
  Name = "Example for a measurement";
  //<...>
ENDINSTELEM;
```

采用来自基本属性 id 的、用于识别的实例属性。此 ID 在应用元素实例中是唯一的，并可用于各引

用实例。引用元素的 ID 用逗号分开。

示例:

```
INSTELEM Measurement
    ID = 121;
    Name = "Exhaust gas testing";
    //...
    Measurement_Quantitys = 200, 201, 202;
ENDINSTELEM;

INSTELEM Measurement_Quantity
    ID = 200;
    Name = "Time";
    //...
ENDINSTELEM;
```

5.7.21.1 ATF/CLA 文件的不完整数据

如果 ATF/CLA-文件中不能满足所有的引用，则某些 ID 不可能给出整数值。在这种情况下，它可以通过“ASAM 路径”描述参考位置的字符串替代。

ASAM 路径通过数据集规定只有一个实例的路径。它必须开始于一个 AoEnvironment 实例或 ASAM ODS 规定的任意顶级元素。

一个 ASAM 路径有一个可选的服务名称，如果包含它被称为一个“全 ASAM 路径”，否则，它被称为“本地 ASAM 路径”。服务名称后跟任意数量的实例规格，包括应用元素名称、实例名称和（可选）版本。应用元素名称放在方括号内。可选版本通过';'与实例名称分开。

下列语法图 4 个字符串用标识符表示，见图 94。

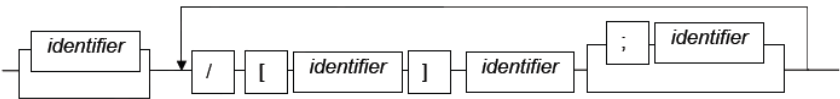


图94 “ASAM 路径” 语法图

描述 ASAM 路径的字符串具有以下结构:

[<Servicename>] { '/' <AppElem> '[' <InstElem> [';' <Version>] }*

- <Servicename> 能够提供 ASAM ODS 应用元素或实例（可选）的文件名或其他定位器;
- <AppElem> <Servicename>的可用应用元素的名称（强制的）;
- <InstElem> <InstElem>的可用实例名称（强制的）;
- <VERSION> <InstElem>实例的现有版本（可选）。

示例: ASAM 路径。以下是一个名为“aeUUTP2”的应用元素的名称为“uutp2a”的实例版本 V2.0 的路径，该实例版本与“MyAsamOdsData.atf”文件中名为“aeUUT1”的应用元素的名称为“uut1”的实例版本 V3.1 有关系（可被发现）。

MyAsamOdsData.atf/[aeUUT1]uut1; 3.1/[aeUUTP2]uutp2a; 2.0

如果服务名称、应用元素名称、实例名称、版本包含任何字符'[', ']', ' ', ' ', '/', '\', 这些字符必须被相应的转义序列替换。

ASAM 路径的转义序列由后面的字符进行转义的字符串“\”定义。因此，表 169 中的转义序列适用于 ASAM。

表169 ASAM 路径中的转义字符

字符	转义字符
[	\[
]	\]
:	\:
/	\/
\	\\

示例：详细的引用说明。下列说明当一个应用元素被子类型化且在其所有子类型中实例的 id 不唯一时如何使用关键字 REF_TYPE。本例中实体 Eq1 和 Eq2 都是应用元素 Equipment（有可能是基础元素 AoTestEquipmentPart 的子类型）的子类型，如果应用模型说明了一个对 Equipment 的引用，则该例中的构造可区分 Eq1 和 Eq2 的实例。

```
INSTELEM ThisEq1
    ID = 1;
    Name = "Engines";
    //...
    Tests = REF_TYPE Eq1 200, 201,
            REF_TYPE Eq2 200;
ENDINSTELEM;
```

```
INSTELEM Eq1
    ID = 200;
    //...
ENDINSTELEM;
```

```
INSTELEM Eq1
    ID = 201;
    //...
ENDINSTELEM;
```

```
INSTELEM Eq2
    ID = 200;
    //...
ENDINSTELEM;
```

9.7.22 组件的结构 (COMPONENT)

ASAM ODS 测量数据由基础元素 `AoLocalColumn` 描述。这些本地序列参考测量数量和子矩阵。而其他元素,如测量数量和子矩阵作为正常的基础元素处理,本地序列通常是指大的外部二进制数据集。

该数据集可以用三种不同的方法表示 ATF:

1) 直接来自基本关系“值”,只是作为一个序列的值,而不是使用任何组件机构;没有进一步的文件涉及的关系。人们应该注意到,虽然“值”是根据基础模型的关系,ATF 文件中它被当作一个属性处理。

从基础属性 `external_component` (如果存在)派生的属性保留为空。

对于显式数据,由基础属性 `sequence_representation` 派生的属性被设为“explicit”。

2) 有基础关系“值”派生的关系中,不是作为一个序列值,而是作为一个 ATF 组件;这描述一个与二进制文件的链接,其中包含的值以及如何可以位于该文件内的值。

由基础属性 `external_component` (如果存在)派生的属性保留为空。

对于显式数据,由基础属性 `sequence_representation` 派生的属性被设置为“explicit”。

3) 在混合模式下,使用由基础关系 `external_components` 派生的关系,由基础关系“值”派生的关系仍然是空的,所有其他属性保持不变(例如派生自基础属性 `sequence_representation` 的属性是 `external_component` 的显示数据。对于原始数据,要么是 `raw_linear_external`, `raw_polynomial_external` 或 `raw_linear_calibrated_external`。

第一种方法是对于小型和中等规模数据的一种简单方法,它的用法涵盖了 9.7.21(实例元素的结构)和 9.7.21 (数据属性的值)部分。

第三种方法支持混合模式的概念,独立于 ATF,并且不使用任何特殊的结构。

第二种方法是通过 ATF 传输中型或大数据比较经济的方法。

主要原因是 ATF/CLA 规范提供了 COMPONENT 结构。这种结构允许在一个 ASCII 文件中包括二进制数据,因此给传输和存储可以节省大量的时间和空间。

每个组件在 FILES 块内都必须预先定义。二进制数据的规范必须加在插入的二进制组件之后,即通常在属性等号之后的一个实例。

“组件”语法图见图 95。

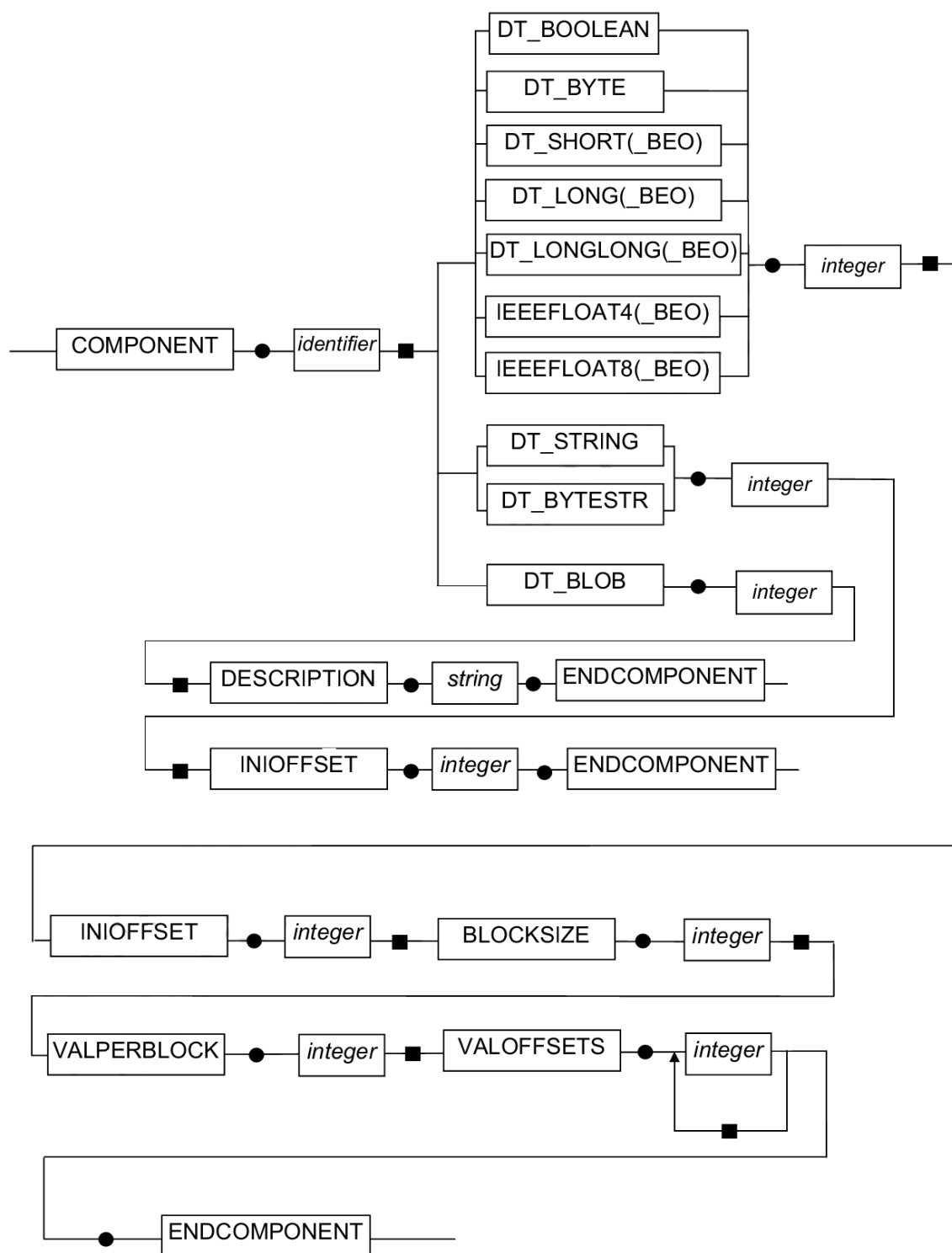


图95 “组件”语法图

整个组件描述要插入属性，而不是 ASCII 表示的值。本部分给出语法图的格式，以及任何附加信息。

(1) 组件识别

关键字 **COMPONENT** 之后必须给出标识名称。此名称必须在 **ATF/CLA** 文件起始处的 **FILES** 块宣布。请注意，**blob** 这个名字是不同的 **blob** 头（放置于 **DESCRIPTION** 关键字之后）。

（2）类型规定

物理数据类型规定数据存储于文件的方式。在这规范简单的整数类型接受了 **IEEE** 浮点格式和几个字节格式：

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

（3）长度规定

类型之后的整数值规定从组件读取的数据量。在 **DT_STRING**、**DT_BYTESTR** 和 **DT_BLOB** 情况下，它表示将要读取的字节数；在其他情况下（整数、浮点数、位和字节）表示将要读取的数值。在 **blob** 情况下，规定与 **blob** 所在文件同等长度，因为一个组件只有一个 **BLOB**。所有其他类型在一个组件文件可能发生序列中的。所有其他类型可能在一个组件文件的序列中出现。

（4）BOOLEAN 表示

在二进制组件文件，每个布尔值来表示一个位（**0 = FALSE**，**1 = 真**）。

这些位被装入一个字节序列。如果位的数量不足以完全填补字节，则该字节的其余部分应保持为未使用状态。字节填充由左到右开始，即从最重要的位开始。在字节序列（**by1**, **by2**, **by3**,...）的第 **n** 个布尔值（**n = 1..Nbi**）位于下列整数格式。假设在一个字节中的位是数字（再次从左至右）**7,6,5,4,3,2,1,0**，第 **n** 个布尔值是 **lbi** 位（from **0,1,...,7**）的 **lby**-th 字节（从 **1** 开始）：

$$lby = (n+7) / 8$$

$$lbi = lby*8 - n$$

使用的字节数（**Nby**）可以通过整数运算从布尔值（**Nbi**）计算得出。

$$Nby = 1 + (Nbi-1) / 8$$

语法图中 **DT_BOOLEAN** 之后的整数应表示的位的数量（即所给布尔值的数目）。

（5）BYTE 表示

此数据类型仅代表字节。因此，DT_BYTE 后面的整数表示序列中字节值的数量，在一个简单值的情况下，它被设置为 1。

(6) SHORT 表示

这 16 位整数数据类型通过 2 个字节表示。如果给出关键字 DT_SHORT，那么最不重要的字节是第一个（低地址），最重要的是最后的字节（高地址）。如果给出关键字 DT_SHORT_BEO，那么最重要的字节是第一个，最不重要的字节是最后一个；顺序也被称为“大字节顺序”。DT_SHORT 之后的整数（相应 DT_SHORT_BEO）表示序列中整数值数量，在简单值的情况下，它被设置为 1。

(7) LONG 表示

这 32 位整数数据类型由 4 个字节表示。如果给出关键字 DT_LONG，那么最不重要的字节是第一个（低地址），最重要的是最后的字节（高地址）。如果给出关键字 DT_LONG_BEO，那么最重要的字节是第一个，最不重要的字节是最后一个；这个顺序也被称为“大字节顺序”。DT_LONG 后面的整数表示序列的整数值数量，在简单值的情况下，它被设置为 1。

(8) LONGLONG 表示

这 64 位整数数据类型由 8 个字节表示。如果给出关键字 DT_LONGLONG，那么最不重要的字节是第一个（低地址），最重要的是最后一个字节（高地址）。如果给出关键字 DT_LONGLONG_BEO，那么最重要的字节是第一位的，最不重要的字节是最后一个；这个顺序也被称为“大字节顺序”。DT_LONGLONG 之后的整数表示序列中整数值数量，在这简单值情况下，它被设置为 1。

(9) FLOAT 表示

此数据类型由 4 个字节表示。如果给出关键字 IEEEFLOAT4，那么最不重要的字节是第一个（低地址），最重要的是最后的字节（高地址）。如果给出关键字 IEEEFLOAT4_BEO，那么最重要的字节是第一位的，最不重要的字节是最后一个，这个顺序也被称为“大字节顺序”。符号的位置、指数和尾数在浮点数结构部分进行描述。IEEEFLOAT4 之后的整数表示序列中浮点值的数目，在简单值的情况下，它被设置为 1。

(10) DOUBLE 表示

此数据类型是由 8 个字节表示。如果给出关键字 IEEEFLOAT8，那么最不重要的字节是第一个（低地址），最重要的是最后的字节（高地址）。如果给出关键字 IEEEFLOAT8_BEO，那么最重要的字节是第一个，最不重要的字节是最后一个；这个顺序也被称为“大字节顺序”。符号的位置、指数和尾数在“浮点数结构”部分进行描述。IEEEFLOAT8 之后的整数表示序列中浮点值的数值，在简单值的情况下，它被设置为 1。

(11) COMPLEX 的表示

复杂数字由关键字 IEEEFLOAT4 和 IEEEFLOAT8（相应 IEEEFLOAT4_BEO 和 IEEEFLOAT8_BEO）控制。值的数量必须增加一倍（即仅偶数是合法的）。对于数据类型之后的单一的复杂值的整数是 2，对于 7 复杂的值的序列整数是 14。复杂序列的顺序是：

- 第一个序列成员的实数部分；
- 第一个序列成员的虚数部分；
- 第二个序列成员的实数部分；
- 第二个序列成员的虚数部分；

——。

(12) STRING 表示

此数据类型由字节表示。每个字符串以一个 NULL 字节终止，这增加了该字符串 1 所需的空空间。在一个字符串序列中该序列的每个构件具有一个终止 NULL 字节，该组件的总长度包括这些 NULL 字节，并在 DT_STRING 之后的整数部分给出。数据类型 DT_STRING 在相同的组件文件中可能不会与其他数据类型混合。

(13) BYTESTR 表示

此数据类型由字节表示。字节流中的第 4 个字节表示这 4 个字节之后的字节数量。在序列的前一个成员的最后字节之后立即给出字节流序列的第二长度指示。请注意，在“大字节顺序”中给出第 4 长度指示字节，即首先最重要的字节。

示例：内容为“”和“”的 2 个字节流应当表示为 00 00 00 07 41 42 43 44 45 46 47 00 00 00 03 58 59 5A
总长度为 18，在 DT_BYTESTR 后写上 18 这个整数。

在相同的组件文件中数据类型 DT_BYTESTR 可能不与其他的数据类型混合。

(14) BLOB 表示

小圆块是一个含有已知长度和头的字节流。而小圆块的 ASCII 格式由两个字符串（第一个包含了头，第二个包含的字节流）表示，在组件处理不同：

——关键字 DESCRIPTION 之后小圆块的头进入字符串；

——DT_BLOB 之后在整数中给出小圆块的长度（与其他长度的迹象是兼容的，因为每次总是只有一个小圆块）；

——小圆块的内容是二进制文件（除了小圆块不包含任何内容）。

在相同的组件文件数据类型 DT_BLOB 可能无法与其他数据类型混合。

(15) 其他参数

文件头规定了含有 INIOFFSET 的字节数。访问文件时，该文件头被跳过。由于小圆块总是覆盖整个文件，在小圆块组件没 INIOFFSET。

文件（文件头后的开始）的其余部分被视为一个或许多块。一个块由值组成。这些值可能属于不同的组件，（在数值数据的情况下）可以具有不同的数据类型。一个块的几个值可能属于同一个组件，在本节结束部分能见到例子。对于所有的数值数据类型，必须规定块大小。在字节中给出 BLOCK SIZE。

在每个块中属于所需组件的值的数量是以 VALPERBLOCK 给定。对于块内每个值的偏移量在字节内给出。所有这些偏移集中于关键字 VALOFFSETS 之后。该程序在以下例子中显示。

示例1：无组件的本地列。下列实例中 ATF/CLA 文件物理上给出了数据。

```
INSTELEM localcolumn
  ID = 2;
  Values = DATATYPE DT_DOUBLE
    ,260.
    ,270.
    ,280.
    ,290.
```

```
, 300.
, 310.
, 320.
, 330.
, 340.
, 350.
, 360.;
ENDINSTELEM;
```

示例2: 一个文件中的一个本地列。该文件包含一个测量量，其测量通道值具有双精度浮点数值。

```
INSTELEM localcolumn
  ID = 3;
  Values = DATATYPE DT_DOUBLE,
  COMPONENT binary_file, IEEEREAL8 11, INIOFFSET 0,
    BLOCKSIZE 8, VALPERBLOCK 1, VALOFFSETS 0
  ENDCOMPONENT;
ENDINSTELEM;
```

示例3: 两个相互紧邻的本地列。该文件包含 2 个测量量：

——一个测量通道值是 10 位双精度浮点数值；

——一个测量通道值是 10 位短整型数值。

第二个 **COMPONENT** 在第一个 **COMPONENT** 之后间隔 80 字节。

```
INSTELEM localcolumn
  ID = 4;
  Values = DATATYPE DT_DOUBLE,
  COMPONENT binary_file, IEEEREAL8 10, INIOFFSET 0,
    BLOCKSIZE 8, VALPERBLOCK 1, VALOFFSETS 0
  ENDCOMPONENTENDINSTELEM;
INSTELEM localcolumn
  ID = 3;
  Values = DATATYPE DT_SHORT,
  COMPONENT binary_file, DT_SHORT 10, INIOFFSET 80,
    BLOCKSIZE 2, VALPERBLOCK 1, VALOFFSETS 0
  ENDCOMPONENT;
ENDINSTELEM;
```

示例4: 一个文件中相互交替的 2 个本地列。该文件包含 2 个测量量，采用相同的取样速率：

——一个生成时间通道具有双精度浮点数值；

——一个测量值通道具有短整型数值。

这些值本轮流写入文件，它们被两个本地列引用。

```
INSTELEM localcolumn
```

```

ID = 6;
Values = DATATYPE DT_DOUBLE,
COMPONENT binary_file, IEEEREAL8 1000, INIOFFSET 0,
      BLOCKSIZE 10, VALPERBLOCK 1, VALOFFSETS 0
ENDCOMPONENT;
ENDINSTELEM;
INSTELEM aolocalcolumn
ID = 7;
Values = DATATYPE DT_SHORT,
COMPONENT binary_file, DT_SHORT 1000, INIOFFSET 0,
      BLOCKSIZE 10, VALPERBLOCK 1, VALOFFSETS 8
ENDCOMPONENT;
ENDINSTELEM;

```

示例5：一个文件中不同采样率的3个本地列。该文件包含三个测量量，具有不同的采样速率：

——一个通道值是采样率为10赫兹的双精度浮点数值；

——一个通道值是采样率为2赫兹的短整型数值；

——一个通道值是采样率为30赫兹的长整型数值。

这些来自测量仪器的值被写入文件，在写入第一个测量数据前，测量仪器发送32字节的参数。因此需要3个本地列。

```

INSTELEM localcolumn
ID = 8;
Values = DATATYPE DT_DOUBLE,
COMPONENT binary_file, IEEEREAL8 10000, INIOFFSET 32,
      BLOCKSIZE 24, VALPERBLOCK 1, VALOFFSETS 0
ENDCOMPONENT;
ENDINSTELEM;
INSTELEM localcolumn
ID = 9;
Values = DATATYPE DT_SHORT,
COMPONENT binary_file, DT_SHORT 20000, INIOFFSET 32,
      BLOCKSIZE 24, VALPERBLOCK 2, VALOFFSETS 8,18
ENDCOMPONENT;
ENDINSTELEM;
INSTELEM localcolumn
ID = 10;
Values = DATATYPE DT_LONG,
COMPONENT binary_file, DT_LONG 30000, INIOFFSET 32,
      BLOCKSIZE 24, VALPERBLOCK 3, VALOFFSETS 10,14,20
ENDCOMPONENT;
ENDINSTELEM;

```

9.7.23 结束文件的结构（ENDFILE）

ATF/CLA 文件必须使用关键字 ATF_END 结束。“ENDFILE” 语法图见图 96。

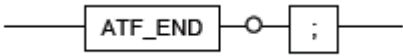


图96 “ENDFILE” 语法图

9.8 ATF/CLA 文件的结构概述（ATF_OVERVIEW）

本部分参照上述元素给出 ATF/CLA 文件的结构概述。语法图只包含主要组件。定位在 ATF/CLA 文件几个位置中的语法元素（它可以被放置在任何位置，其中一个 instelem 或 applelem 允许在图中），未保持该图的可读性。

这是要考虑的，除了 ATF/CLA 文件标识符没有一项下列元素必须是有效的。如果阅读应用程序在一个已经存在的环境中从该文件集成信息，对于 ATF/CLA 文件的有效性可能没有一个自己环境的定义。已经存在的数据结构是应用程序的责任。

正如之前已经提到的，逻辑上的合理分配的信息对已经存在的数据结构是应用程序的责任。ATF/CLA 文件仅被看作是一个标准化的通信平台类似于自然语言。理解这种语言定义的复杂关系需要从所有的通讯伙伴得到相应的知识。

“atf_overview” 语法图见图 97。

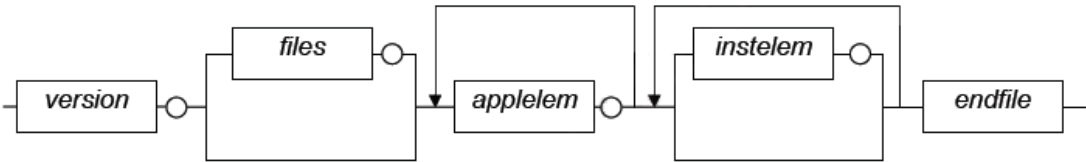


图97 “atf_overview” 语法图

9.9 ATF/CLA 文件的安全信息

9.9.1 ATF/CLA 安全信息的使用

涉及访问保护的相关信息可通过 ASAM ODS ATF/CLA 文件传输。然而，人们应该注意的是这些 ATF/CLA 文件是不加密的、人类可读的，虽然最终消耗时间、访问权限进行了分析和手动修改。

根据 ATF/CLA 文件的目的，可以有多种用例，例如：

——输出存档后重新导入到同一系统中：

安全信息用于补充归档数据，以便能够完全恢复数据，即一起访问的权限。

——与其他系统、公司等数据进行数据交换；导入到不同的系统：

没有安全信息情况下，交流的目的对于写 ATF/CLA 文件是必要的，因为导入可能变得更加合理，如果新的访问权限规定更好地在目标服务器上匹配的情况。

可以写几类安全信息。这些可能包括或排除：

——派生自 AoUser 和 AoUserGroup 的应用元素的实例，（在 ATF/CLA 文件的元数据部分加上这

些应用元素的相应定义)；

- ACL 实体的数据集 (ACLI、ACLA、ACL 模板)；
- 实体 SecurityLevel 的数据集；
- 实体 InitialRightsAt 的数据集。

ATF/CLA 阅读客户应该知道安全信息可能包含在文件中。客户端可以通过各种目标实现，因此安全性信息的处理可能会有所不同。

对于数据库和 ATF/CLA 文件之间相互的数据导入/导出，用户必须经过必要授权来访问数据库中的数据对象；而访问 ATF/CLA 文件，不需要权限。

一般用户访问数据库时需要下列权限：

- 数据导出：阅读；
- 删除导出的数据对象：删除；
- 数据导入：插入和更新。

有关 ASAM ODS 的安全概念详见第 5 章。

9.9.2 ATF/CLA 安全信息的表达

安全信息包括：

- 应用元素的实例派生于：
 - AoUser (必须在元数据部分描述应用元素)；
 - AoUserGroup (必须在元数据部分描述应用元素)。
- 作为下列子类型的实体数据集
 - ACL (ACLA、ACLI、ACL 模板)。
- 以下两个实体的数据集：
 - SecurityLevel；
 - InitialRightsAt。

实体 ACL 子类型没有来自用户。只有三个预定义的子类型是被允许的：

- ACLA (用于规定应用元素和其属性名称的访问权限；‘元素安全’或‘属性安全’的情况下)；
- ACLI (用于规定实例访问权限的情况，在“实例安全”的情况)；
- ACLTemplate (用于说明初始权限：新创建实例的访问权限；仅与“实例安全”相关)。

SecurityLevel、InitialRightsAt 和 ACL 子类不是应用元素；而不只是属于基础模型的实体。因此，他们在 ATF/CLA 文件 (没有 applelem 结构) 的元数据部分中未描述。

请注意 ACL 子类该项目的“用户”是指来自 AoUserGroup 的应用程序实例。因此，该数据集项目值必须使用 REF_TYPE 的结构 (reference_attribute_values) 以规定应用元素，由项目‘用户’点值给出的 ID。

无需特别的语法施加到的安全信息。按所有其他数据相同的规则写 ATF/CLA 文件：

- 来自 AoUsers 和 AoUserGroups 的应用元素必须在 ATF/CLA 文件定义，都与其他应用元素一起写在元数据部分。ACL 的子类在 ATF/CLA 文件元数据部分未声明；
- 与其他实例一起将实例写入数据部分；

——数据集(**SecurityLevel**、**InitialRightsAt**、**ACL** 子类), 均应被视为他们的情况, 必须符合 **instelem** 的语法和语义规范。它们与实例一起被写入数据部分。

本章结尾的示例文件中包含安全信息, 并用注释解释了其含义。

9.10 继承的概念和 ATF/CLA

ASAM ODS 继承的概念可以定义一个应用元素(子类), 从其他应用元素(超类)继承的应用程序属性和关系, 并进一步增加了应用程序属性和扩展应用的关系。如果这样的子类和父类元素和它们的实例以 **ATF/CLA** 文件传输, 则适用下列规则。

应用模型的描述规则:

- 父类和子类两个的应用元素必须在 **ATF** 文件通过 **APPLELEM** 结构规定;
- 父类应用元素的规定定义了其所有的应用程序属性和应用关系;
- 子类应用元素的规定定义那些应用程序属性和特定子类(对应的父类中尚未定义)的应用程序关系。此规则的唯一例外的是来自基本属性'**id**'的应用程序属性。此属性将在子类中被定义, 但它的定义语法不同于通常的、标识符属性的定义;
- 在子类规范中来自基本属性的应用程序属性 '**id**' 通过图 98 的语法进行定义(这是一个上面给出的有效的语句 **APPLATTR** 结构)。



图98 应用程序属性标识符

第一标识符是应用程序属性的名称。在超类和子类元素方面, 应严格对应用程序属性避免提供不同的名称(不同名称的选项只适用于传统的数据!)。

第二个标识符必须是'**id**', 因为这是 **ASAM ODS** 基本属性的名称。

第三标识符必须是从该子类继承的子类应用元素的名称。

- 在继承的情况下访问 **ASAM ODS** 服务器应用程序模型包含一个应用程序来自超类和子类元素的基本属性的“对象类型”的属性, 这是必要的 **API** 来选择超类或子类的实例。

因此, 如果企图把 **ATF/CLA** 文件内容写入 **ASAM ODS** 服务器, **ATF/CLA** 文件应规定超类应用元素的该应用程序属性(这是由子类应用元素继承的)。如果没有这样的应用程序属性, 在向服务器输入信息之前, 给服务器写应用模型的客户端必须生成一个。

请注意, 这个属性的数据类型由 **ASAM ODS** 基础模型定义 **T_LONGLONG** 。

- 如果不计划将 **ATF/CLA** 文件内容写入 **ASAM ODS** 服务器, **ATF/CLA** 文件并不不需要提供一个来自基本属性的“对象类型”的应用程序属性。

实例的描述规则:

- 每个子类应用元素的实例由 **INSTELEM** 结构(如上述定义)给出。它给所有应用程序属性和关系插入值, 不管他们是否从超类的应用元素继承或单独添加到子类元素;
- 如果应用程序模型包含一个来自基本属性“对象类型”的属性, 实例可能包含该属性的一个正整

数（通常是 AID，如果 ATF 文件内容从服务器读取）。另外它可能会被忽略（在此属性的有用值不知道的情况下，ATF 文件的内容无法从服务器读取）。此外，允许提供 ‘UNDEFINED’ 的属性值；

- 如果应用程序模型包含一个来自基本属性“对象类型”的属性，计划将 ATF/CLA 文件内容写入 ASAM ODS 服务器，那么 ATF/CLA 文件应该为每个实例中超类和子类提供一个属性值。如果不存在这样的值，在向服务器输入信息之前，给服务器写应用模型的客户端必须生成一个；
- 应用程序模型包含一个来自基本属性“对象类型”的属性，但它不计划将 ATF/CLA 文件内容写入 ASAM ODS 服务器，ATF/CLA 文件并不需要提供属性值；
- 如果应用程序模型不包含一个来自基本属性“对象类型”的属性，实例不提供这样的属性的值内的 ATF/CLA 文件。如果 ATF/CLA 文件的内容将被写入到服务器的客户端应用程序，该客户端应用程序必须生成一个应用程序的属性来自基本属性'的 OBJECTTYPE “（前写的应用程序模型到服务器），而且必须此属性在每个实例中生成一个合适的值（之前写的实例的服务器）；
- 来自基本属性'对象类型'的属性值是一个 T_LONGLONG 表示范围内的正整数；
- 当从服务器读取数据并将其写入到 ATF/CLA 文件时，必须注意：
超类实例必须从服务器通过 “objecttype” 集合向该超类的应用 id（AID）获取。

示例：ATF/CLA 的继承。假设一个应用模型如下图 99，则该模型及其 EngineSuper 实例（EName= “SuperEng” ，EBore=80.4，EStroke=85.6）和 EngineSub 实例（EName= “SubEng” ，EBore=80.5，EStroke=85.3，Eversion= “2.4” ，Ebore1=81.5，EBore2=79.5）在 ATF/CLA 文件中可表示为：

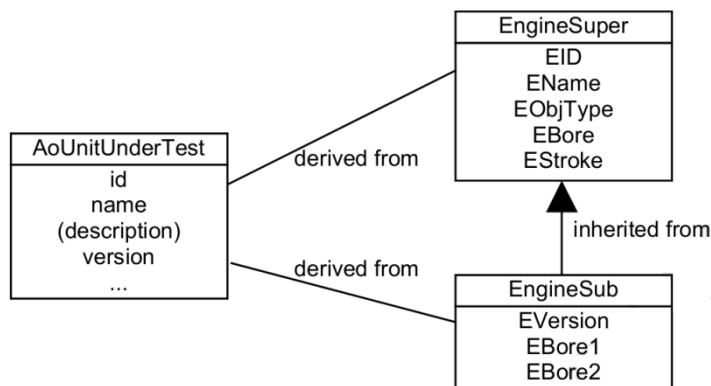


图99 ATF/CLA 的继承

```
ATF_FILE V1.4.1;
APPLELEM EngineSuper, BASETYPE AoUnitUnderTest
    APPLATTR EID, BASEATTR id, DATATYPE DT_LONGLONG;
    APPLATTR EName, BASEATTR name, DATATYPE DT_STRING;
    APPLATTR EObjType, BASEATTR objecttype, DATATYPE DT_LONGLONG;
    APPLATTR EBore, DATATYPE DT_DOUBLE;
    APPLATTR EStroke, DATATYPE DT_DOUBLE;
ENDAPPLELEM;
```



```

APPLELEM EngineSub, BASETYPE AoUnitUnderTest
  APPLATTR EID, BASEATTR id, REF_TO EngineSuper;
  APPLATTR EVersion, BASEATTR version, DATATYPE DT_STRING;
  APPLATTR EBore1, DATATYPE DT_DOUBLE;
  APPLATTR EBore2, DATATYPE DT_DOUBLE;
ENDAPPLELEM;
INSTELEM EngineSuper
  EID = 27;
  EName = "SuperEng";
  EObjType = 4711;
  EBore = 80.4;
  EStroke = 85.6;
ENDINSTELEM;
INSTELEM EngineSub
  EID = 31;
  EName = "SubEng";
  EObjType = 815;
  EBore = 80.5;
  EStroke = 85.3;
  EVersion = "2.4";
  EBore1= 81.5;
  EBore2= 79.5;
ENDINSTELEM;
ATF_END;

```

尽管是非严格要求的，但上述示例文件中包含了“objecttype”的信息。特别需要说明的是当不准备将 ATF/CLA 文件的内容放入 ASAM ODS 服务器中时，可以忽略以下内容：

```

APPLATTR EObjType, BASEATTR objecttype, DATATYPE DT_LONGLONG;
...
  EObjType = 4711;
...
  EObjType = 815;

```

9.11 应用示例

下面的例子给出了一个简化的 ATF/CLA 文件。该模型的主要元素如下图所示。基础模型的详细信息见第 4 章。ATF/CLA 文件示例的应用模型见图 100。

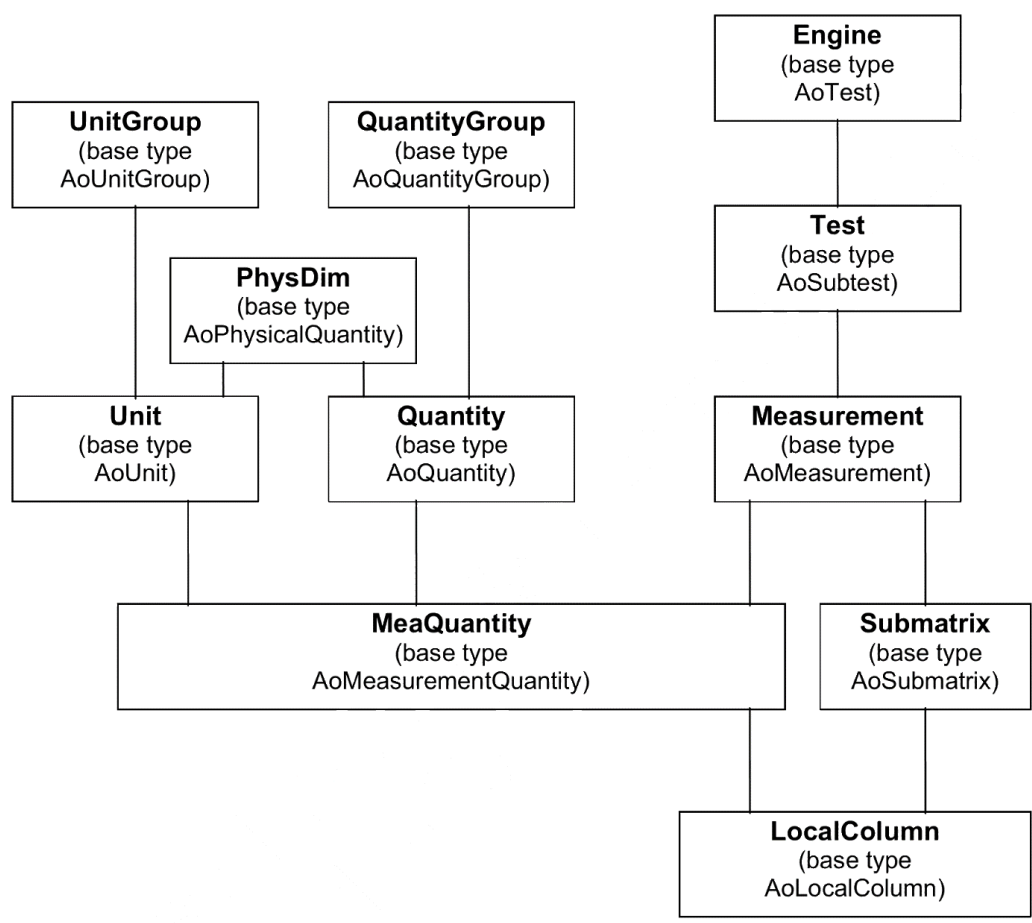


图100 ATF/CLA 文件示例的应用模型

ATF/ACL示例文件如下。

```
ATF_FILE V1.4.1;
/*
//-----
// EXAMPLE 3 --- 22.9.2000
//-----
//*****
//
// Application Structure
//
//*****
*/
/*
//-----
//
// Hierarchic ordering of tests and measurements
// Application elements
//           Engine,   Test,  Measurement, MeaQuantity
```

```
// of base type AoTest, AoSubtest, AoMeasurement, AoMeasurementQuantity
//
// Security information is included
//
//-----
*/
```

APPLELEM Engine, BASETYPE AoTest

```
    APPLATTR EngineName,      BASEATTR name,      DATATYPE DT_STRING;
    APPLATTR EngineId,        BASEATTR id,         DATATYPE DT_LONGLONG;
    APPLATTR EngineVersion,    BASEATTR version,    DATATYPE DT_STRING;
    APPLATTR EngineDescription,BASEATTR description, DATATYPE DT_STRING;
    APPLATTR version_date,     BASEATTR version_date, DATATYPE DT_DATE;
    APPLATTR CreateDate,      DATATYPE DT_DATE;
    APPLATTR EngineType,      DATATYPE DT_STRING;
    APPLATTR bore,            DATATYPE DT_FLOAT;
    APPLATTR cylindernumber,   DATATYPE DT_SHORT;
    APPLATTR stroke,          DATATYPE DT_FLOAT;
    APPLATTR SubTests,        BASEATTR children, REF_TO Test,
                                CARDINALITY
                                0,MANY;
    APPLATTR EngineUser,      REF_TO User;
ENDAPPLELEM;
```

APPLELEM Test, BASETYPE AoSubTest

```
    APPLATTR TestName,        BASEATTR name,      DATATYPE DT_STRING;
    APPLATTR TestId,          BASEATTR id,         DATATYPE DT_LONGLONG;
    APPLATTR TestVersion,     BASEATTR version,    DATATYPE DT_STRING;
    APPLATTR TestDescription, BASEATTR description, DATATYPE DT_STRING;
    APPLATTR version_date,     BASEATTR version_date, DATATYPE DT_DATE;
    APPLATTR CreateDate,      DATATYPE DT_DATE;
    APPLATTR TestType,        DATATYPE DT_STRING;
    APPLATTR TestComment,     DATATYPE DT_STRING;
    APPLATTR exhaust,         DATATYPE DT_STRING;
    APPLATTR air_filter,       DATATYPE DT_STRING;
    APPLATTR Measurements,    BASEATTR children, REF_TO Measurement,
                                CARDINALITY
                                0,MANY;
    APPLATTR MainTest,        BASEATTR parent_test, REF_TO Engine;
    APPLATTR TestUser,        REF_TO User;
ENDAPPLELEM;
```

APPLELEM Measurement, BASETYPE AoMeasurement

```

APPLATTR MeaName,          BASEATTR name,          DATATYPE DT_STRING;
APPLATTR MeaId,            BASEATTR id,            DATATYPE DT_LONGLONG;
APPLATTR MeaVersion,       BASEATTR version,       DATATYPE DT_STRING;
APPLATTR MeaDescription,   BASEATTR description, DATATYPE DT_STRING;
APPLATTR version_date,     BASEATTR version_date, DATATYPE DT_DATE;
APPLATTR MeasurementBegin, BASEATTR measurement_begin, DATATYPE
                                DT_DATE;
APPLATTR MeasurementEnd,   BASEATTR measurement_end, DATATYPE
                                DT_DATE;
APPLATTR CreateDate,              DATATYPE DT_DATE;
APPLATTR MeaType,                 DATATYPE DT_STRING;
APPLATTR MeaComment,              DATATYPE DT_STRING;
APPLATTR ROZ,                     DATATYPE DT_FLOAT;
APPLATTR Test,                    BASEATTR test,          REF_TO Test;
APPLATTR MeaQuantities,           BASEATTR measurement_quantities,
                                REF_TO MeaQuantity, CARDINALITY 0,MANY;
APPLATTR Submatrices,            BASEATTR submatrices, REF_TO Submatrix,
                                CARDINALITY 0,MANY;
ENDAPPLELEM;

```

```
APPLELEM MeaQuantity, BASETYPE AoMeasurementQuantity
```

```

APPLATTR MeaQName,          BASEATTR name,          DATATYPE DT_STRING;
APPLATTR MeaQId,            BASEATTR id,            DATATYPE DT_LONGLONG;
APPLATTR MeaQVersion,       BASEATTR version,       DATATYPE DT_STRING;
APPLATTR MeaQDescription,   BASEATTR description, DATATYPE DT_STRING;
APPLATTR version_date,     BASEATTR version_date, DATATYPE DT_DATE;
APPLATTR Rank,              BASEATTR rank,          DATATYPE DT_LONG;
APPLATTR Dimension,         BASEATTR dimension, DATATYPE DT_LONG,
                                CARDINALITY
                                0,MANY;
APPLATTR DataType,          BASEATTR datatype, DATATYPE DT_ENUM;
APPLATTR TypeSize,          BASEATTR type_size, DATATYPE DT_LONG;
APPLATTR Interpolation,     BASEATTR interpolation, DATATYPE DT_ENUM;
APPLATTR Minimum,           BASEATTR minimum, DATATYPE DT_DOUBLE;
APPLATTR Maximum,           BASEATTR maximum, DATATYPE DT_DOUBLE;
APPLATTR Average,           BASEATTR average, DATATYPE DT_DOUBLE;
APPLATTR SDeviation,        BASEATTR standard_deviation, DATATYPE
                                DT_DOUBLE;
APPLATTR MeaQuantStat,              DATATYPE DT_SHORT;
APPLATTR TimeOffset,                DATATYPE DT_DOUBLE;
APPLATTR SamplingRate,              DATATYPE DT_DOUBLE;
APPLATTR Measurement,               BASEATTR measurement, REF_TO Measurement;
APPLATTR LocalColumns,              BASEATTR local_columns, REF_TO LocalColumn,

```

```

                                CARDINALITY
                                0,MANY;

    APPLATTR Quantity,          BASEATTR quantity, REF_TO Quantity;
    APPLATTR Unit,              BASEATTR unit,        REF_TO Unit;
ENDAPPLELEM;

APPLELEM Submatrix, BASETYPE AoSubmatrix
    APPLATTR name,              BASEATTR name,        DATATYPE DT_STRING;
    APPLATTR id,                BASEATTR id,          DATATYPE DT_LONG;
    APPLATTR version,           BASEATTR version,    DATATYPE DT_STRING;
    APPLATTR description,       BASEATTR description, DATATYPE DT_STRING;
    APPLATTR version_date,      BASEATTR version_date, DATATYPE DT_DATE;
    APPLATTR number_of_rows,    BASEATTR number_of_rows, DATATYPE DT_LONG;
    APPLATTR local_columns,     BASEATTR local_columns, REF_TO LocalColumn,
                                CARDINALITY
                                0,MANY;

    APPLATTR measurement,       BASEATTR measurement, REF_TO Measurement;
ENDAPPLELEM;

APPLELEM LocalColumn, BASETYPE AoLocalColumn
    APPLATTR name,              BASEATTR name,        DATATYPE DT_STRING;
    APPLATTR id,                BASEATTR id,          DATATYPE DT_LONG;
    APPLATTR version,           BASEATTR version,    DATATYPE DT_STRING;
    APPLATTR description,       BASEATTR description, DATATYPE DT_STRING;
    APPLATTR version_date,      BASEATTR version_date, DATATYPE DT_DATE;
    APPLATTR flags,             BASEATTR flags,       DATATYPE DT_SHORT,
                                CARDINALITY
                                0,MANY;

    APPLATTR global_flag,       BASEATTR global_flag, DATATYPE DT_SHORT;
    APPLATTR independent,       BASEATTR independent, DATATYPE DT_BOOLEAN;
    APPLATTR minimum,           BASEATTR minimum,     DATATYPE DT_DOUBLE;
    APPLATTR maximum,           BASEATTR maximum,     DATATYPE DT_DOUBLE;
    APPLATTR values,            BASEATTR values,      DATATYPE DT_UNKNOWN,
                                CARDINALITY
                                0,MANY;

    APPLATTR submatrix,         BASEATTR submatrix, REF_TO Submatrix;
    APPLATTR measurement_quantity, BASEATTR measurement_quantity,
                                REF_TO
                                MeaQuantity;
ENDAPPLELEM;

```

/*

//-----

```
//
// Construction of a catalogue of quantities
// Application elements
//           Quantity (-Group), Unit (-Group), PhysDim,
// of base type AoQuantity (-Group), AoUnit (-Group), AoPhysicalDimension
//
//-----
*/
```

```
APPLELEM Quantity, BASETYPE AoQuantity
```

```
    APPLATTR QName,          BASEATTR name,          DATATYPE DT_STRING;
    APPLATTR QId,            BASEATTR id,             DATATYPE DT_LONGLONG;
    APPLATTR QVersion,       BASEATTR version,        DATATYPE DT_STRING;
    APPLATTR QDescription,    BASEATTR description,    DATATYPE DT_STRING;
    APPLATTR version_date,    BASEATTR version_date,   DATATYPE DT_DATE;
    APPLATTR DefaultRank,     BASEATTR default_rank,   DATATYPE DT_LONG;
    APPLATTR DefaultDimension, BASEATTR default_dimension, DATATYPE
                                DT_LONG, CARDINALITY
```

```
0, MANY;
```

```
    APPLATTR DefaultDatatype, BASEATTR default_datatype, DATATYPE DT_ENUM;
    APPLATTR DefaultType_size, BASEATTR default_type_size, DATATYPE
                                DT_LONG;
    APPLATTR DefaultMqName,   BASEATTR default_mq_name, DATATYPE DT_STRING;
    APPLATTR DefaultUnit,     BASEATTR default_unit,   REF_TO Unit;
    APPLATTR Successors,      BASEATTR successors,    REF_TO Quantity,
                                CARDINALITY
```

```
0, MANY;
```

```
    APPLATTR Predecessor,     BASEATTR predecessor, REF_TO Quantity;
    APPLATTR QuantityGroup,    BASEATTR groups,      REF_TO QuantityGroup,
    CARDINALITY 0, MANY;
```

```
ENDAPPLELEM;
```

```
APPLELEM QuantityGroup, BASETYPE AoQuantityGroup
```

```
    APPLATTR QGroupName,      BASEATTR name,          DATATYPE DT_STRING;
    APPLATTR QGroupId,         BASEATTR id,             DATATYPE DT_LONGLONG;
    APPLATTR QGroupVersion,    BASEATTR version,        DATATYPE DT_STRING;
    APPLATTR QGroupDescription, BASEATTR description,    DATATYPE DT_STRING;
    APPLATTR version_date,     BASEATTR version_date,   DATATYPE DT_DATE;
    APPLATTR QantGroupState,    DATATYPE DT_SHORT;
    APPLATTR ListOfQuantities,  BASEATTR quantities,    REF_TO
```

```
Quantity, CARDINALITY 0, MANY;
```

```
ENDAPPLELEM;
```

APPLELEM Unit, BASETYPE AoUnit

```

    APPLATTR UnitName, BASEATTR name, DATATYPE DT_STRING;
    APPLATTR UnitId, BASEATTR id, DATATYPE DT_LONGLONG;
    APPLATTR UnitVersion, BASEATTR version, DATATYPE DT_STRING;
    APPLATTR UnitDescription, BASEATTR description, DATATYPE DT_STRING;
    APPLATTR version_date, BASEATTR version_date, DATATYPE DT_DATE;
    APPLATTR UnitFactor, BASEATTR factor, DATATYPE DT_FLOAT;
    APPLATTR UnitOffset, BASEATTR offset, DATATYPE DT_FLOAT;
    APPLATTR PhysDim, BASEATTR phys_dimension, REF_TO PhysDim;
    APPLATTR UnitGroups, BASEATTR groups, REF_TO UnitGroup, CARDINALITY
0, MANY;
ENDAPPLELEM;
```

APPLELEM UnitGroup, BASETYPE AoUnitGroup

```

    APPLATTR UGroupName, BASEATTR name, DATATYPE DT_STRING;
    APPLATTR UGroupId, BASEATTR id, DATATYPE DT_LONGLONG;
    APPLATTR UGroupVersion, BASEATTR version, DATATYPE DT_STRING;
    APPLATTR UGroupDescription, BASEATTR description, DATATYPE DT_STRING;
    APPLATTR version_date, BASEATTR version_date, DATATYPE DT_DATE;
    APPLATTR UnitGroupstate, DATATYPE DT_SHORT;
    APPLATTR Units, BASEATTR units, REF_TO Unit, CARDINALITY 0, MANY;
ENDAPPLELEM;
```

APPLELEM PhysDim, BASETYPE AoPhysicalDimension

```

    APPLATTR PhysDimName, BASEATTR name, DATATYPE DT_STRING;
    APPLATTR PhysDimId, BASEATTR id, DATATYPE DT_LONGLONG;
    APPLATTR PhysDimVersion, BASEATTR version, DATATYPE DT_STRING;
    APPLATTR PhysDimDescription, BASEATTR description, DATATYPE DT_STRING;
    APPLATTR version_date, BASEATTR version_date, DATATYPE DT_DATE;
    APPLATTR length, BASEATTR length_exp, DATATYPE DT_SHORT;
    APPLATTR mass, BASEATTR mass_exp, DATATYPE DT_SHORT;
    APPLATTR time, BASEATTR time_exp, DATATYPE DT_SHORT;
    APPLATTR current, BASEATTR current_exp, DATATYPE DT_SHORT;
    APPLATTR temperature, BASEATTR temperature_exp, DATATYPE DT_SHORT;
    APPLATTR molar, BASEATTR molar_amount_exp, DATATYPE DT_SHORT;
    APPLATTR luminous, BASEATTR luminous_intensity_exp, DATATYPE DT_SHORT;
    APPLATTR Units, BASEATTR units, REF_TO Unit, CARDINALITY 0, MANY;
ENDAPPLELEM;

/*
//-----
//
// Application elements User, UserGroup, ACLA, ACLI used for
// security information
```

```

//
//-----
*/
APPLELEM User, BASETYPE AoUser
    APPLATTR UserName, BASEATTR name, DATATYPE DT_STRING;
    APPLATTR UserId, BASEATTR id, DATATYPE DT_LONGLONG;
    APPLATTR UserVersion, BASEATTR version, DATATYPE DT_STRING;
    APPLATTR UserDescription, BASEATTR description, DATATYPE DT_STRING;
    APPLATTR version_date, BASEATTR version_date, DATATYPE DT_DATE;
    APPLATTR password, BASEATTR password, DATATYPE DT_STRING;
    APPLATTR groups, BASEATTR groups, REF_TO UserGroup, CARDINALITY 0, MANY;
    APPLATTR UserDepartment, DATATYPE DT_STRING;
    APPLATTR AliasName, DATATYPE DT_STRING;
ENDAPPLELEM;

APPLELEM UserGroup, BASETYPE AoUserGroup
    APPLATTR GroupName, BASEATTR name, DATATYPE DT_STRING;
    APPLATTR GroupId, BASEATTR id, DATATYPE DT_LONGLONG;
    APPLATTR GroupVersion, BASEATTR version, DATATYPE DT_STRING;
    APPLATTR GroupDescription, BASEATTR description, DATATYPE DT_STRING;
    APPLATTR version_date, BASEATTR version_date, DATATYPE DT_DATE;
    APPLATTR superuser_flag, BASEATTR superuser_flag, DATATYPE DT_BOOLEAN;
    APPLATTR users, BASEATTR users, REF_TO User, CARDINALITY 0, MANY;
ENDAPPLELEM;

/*
//-----
//
// ACLA, ACLI, ACLTemplate are NOT application elements, instead belong
// only to the base element, therefore NOT included here.
//
// SecurityLevel, InitialRightsAt are NOT application elements, instead
// belong only to the base element, therefore NOT included here.
//
//-----
*/
/*
//*****
//
// Instance elements section
//
//*****
*/
/*

```



```

//-----
// Instance elements of the application element PhysDim
//                               from base type AoPhysicalDimension
//-----
*/
INSTELEM PhysDim
    PhysDimName = "empty ";
    PhysDimId = 1;
    PhysDimVersion = "1.0";
    PhysDimDescription = "No dimension";
    version_date = "199702010900";
    length = 0; // Meter m
    mass = 0; // Kilogramme kg
    time = 0; // Second s
    current = 0; // Ampere A
    temperature = 0; // Kelvin K
    molar = 0; // mol mol
    luminous = 0; // Candela cd
    units = 1; /* Refs to instance elements of Unit */
ENDINSTELEM;

INSTELEM PhysDim
    PhysDimName = "temperature";
    PhysDimId = 2;
    PhysDimVersion = "1.0";
    PhysDimDescription = "temperature dimension";
    version_date = "199702010900";
    length = 0; // Meter m
    mass = 0; // Kilogramme kg
    time = 0; // Second s
    current = 0; // Ampere A
    temperature = 1; // Kelvin K
    molar = 0; // mol mol
    luminous = 0; // Candela cd
    units = 2,3; /* Refs to instance elements of Unit */
ENDINSTELEM;

INSTELEM PhysDim
    PhysDimName = "frequency";
    PhysDimId = 3;
    PhysDimVersion = "1.0";
    PhysDimDescription = "frequency dimension";
    version_date = "199702010900";

```

```

length = 0; // Meter m
mass = 0; // Kilogramme kg
time = -1; // Second s
current = 0; // Ampere A
temperature = 0; // Kelvin K
molar = 0; // mol mol
luminous = 0; // Candela cd
units = 4,5; /* Refs to instance elements of Unit */
ENDINSTELEM;

```

```
INSTELEM PhysDim
```

```

PhysDimName = "work";
PhysDimId = 4;
PhysDimVersion = "1.0";
PhysDimDescription = "work dimension";
version_date = "199702010900";
length = 2; // Meter m
mass = 1; // Kilogramme kg
time = -2; // Second s
current = 0; // Ampere A
temperature = 0; // Kelvin K
molar = 0; // mol mol
luminous = 0; // Candela cd
units = 6; /* Refs to instance elements of Unit */
ENDINSTELEM;

```

```
INSTELEM PhysDim
```

```

PhysDimName = "pressure";
PhysDimId = 5;
PhysDimVersion = "1.0";
PhysDimDescription = "pressure dimension";
version_date = "199702010900";
length = -1; // Meter m
mass = 1; // Kilogramme kg
time = -2; // Second s
current = 0; // Ampere A
temperature = 0; // Kelvin K
molar = 0; // mol mol
luminous = 0; // Candela cd
units = 7,8; /* Refs to instance elements of Unit */
ENDINSTELEM;

```

```
/*
```

```
//-----
```

```

// Instance elements of the application element Unit
// from base type AoUnit
//-----
*/
INSTELEM Unit
    UnitName = "-";
    UnitId = 1;
    UnitVersion = "1.0";
    UnitDescription = "Empty Unit";
    version_date = "199702010900";
    UnitFactor = 1.;
    UnitOffset = 0.;
    PhysDim = 1; /* Ref to instance element of PhysDim */
    UnitGroups = UNDEFINED; /* Refs to instance elements of UnitGroup */
ENDINSTELEM;

INSTELEM Unit
    UnitName = "Grd C";
    UnitId = 2;
    UnitVersion = "1.0";
    UnitDescription = "Grad Celcius";
    version_date = "199702010900";
    UnitFactor = 1.0;
    UnitOffset = 273.15;
    PhysDim = 2; /* Ref to instance element of PhysDim */
    UnitGroups = UNDEFINED; /* Refs to instance elements of UnitGroup */
ENDINSTELEM;

INSTELEM Unit
    UnitName = "Grd K";
    UnitId = 3;
    UnitVersion = "1.0";
    UnitDescription = "Grad Kelvin";
    version_date = "199702010900";
    UnitFactor = 1.0;
    UnitOffset = 0.;
    PhysDim = 2; /* Ref to instance element of PhysDim */
    UnitGroups = UNDEFINED; /* Refs to instance elements of UnitGroup */
ENDINSTELEM;

INSTELEM Unit
    UnitName = "1/min";
    UnitId = 4;

```

```

    UnitVersion = "1.0";
    UnitDescription = "Revolution per minunte";
    version_date = "199702010900";
    UnitFactor = 0.016667;
    UnitOffset = 0.;
    PhysDim = 3; /* Ref to instance element of PhysDim */
    UnitGroups = UNDEFINED; /* Refs to instance elements of UnitGroup */
ENDINSTELEM;

```

```

INSTELEM Unit
    UnitName = "1/sec";
    UnitId = 5;
    UnitVersion = "1.0";
    UnitDescription = "Revolution per second";
    version_date = "199702010900";
    UnitFactor = 1.0;
    UnitOffset = 0.;
    PhysDim = 3; /* Ref to instance element of PhysDim */
    UnitGroups = UNDEFINED; /* Refs to instance elements of UnitGroup */
ENDINSTELEM;

```

```

INSTELEM Unit
    UnitName = "Nm";
    UnitId = 6;
    UnitVersion = "1.0";
    UnitDescription = "Torque in Nm";
    version_date = "199702010900";
    UnitFactor = 1.;
    UnitOffset = 0.;
    PhysDim = 4; /* Ref to instance element of PhysDim */
    UnitGroups = UNDEFINED; /* Refs to instance elements of UnitGroup */
ENDINSTELEM;

```

```

INSTELEM Unit
    UnitName = "mbar";
    UnitId = 7;
    UnitVersion = "1.0";
    UnitDescription = "Pressure in mbar";
    version_date = "199702010900";
    UnitFactor = 100.;
    UnitOffset = 0.;
    PhysDim = 5; /* Ref to instance element of PhysDim */
    UnitGroups = UNDEFINED; /* Refs to instance elements of UnitGroup */

```

```
ENDINSTELEM;
```

```
INSTELEM Unit
```

```
    UnitName = "bar";
    UnitId = 8;
    UnitVersion = "1.0";
    UnitDescription = "Pressure in bar";
    version_date = "199702010900";
    UnitFactor = 100000.;
    UnitOffset = 0.;
    PhysDim = 5; /* Ref to instance element of PhysDim */
    UnitGroups = UNDEFINED; /* Refs to instance elements of UnitGroup */
```

```
ENDINSTELEM;
```

```
/*
```

```
//-----
```

```
// Instance elements of the application element Quantity
```

```
// from base type AoQuantity
```

```
//-----
```

```
*/
```

```
INSTELEM Quantity
```

```
    QName = "TEMP";
    QId = 1;
    QVersion = "1.1";
    QDescription = "Temperature Quantity central";
    version_date = "199703100900";
    DefaultRank = 0;
    DefaultDimension = UNDEFINED;
    DefaultDatatype = "DT_FLOAT";
    DefaultType_size = UNDEFINED;
    DefaultMqName = "TEMP";
    DefaultUnit = 2; /* Ref to instance element of Unit */
    Predecessor = UNDEFINED; /* Ref to instance element of Quantity */
    Successors = 2,3; /* Refs to instance elements of Quantity */
    QuantityGroup = UNDEFINED; /* Ref to instance els.of QuantityGroup */
```

```
ENDINSTELEM;
```

```
INSTELEM Quantity
```

```
    QName = "LEFT";
    QId = 2;
    QVersion = "1.1";
    QDescription = "Temperature Quantity left";
    version_date = "199703100900";
    DefaultRank = 0;
```

```

    DefaultDimension = UNDEFINED;
    DefaultDatatype = "DT_FLOAT";
    DefaultType_size = UNDEFINED;
    DefaultMqName = "TEMP_LEFT";
    DefaultUnit = 2; /* Ref to instance element of Unit */
    Predecessor = 1; /* Ref to instance element of Quantity */
    Successors = UNDEFINED; /* Refs to instance elements of Quantity */
    QuantityGroup = UNDEFINED; /* Ref to instance els.of QuantityGroup */
ENDINSTELEM;

```

INSTELEM Quantity

```

    QName = "RIGHT";
    QId = 3;
    QVersion = "1.1";
    QDescription = "Temperature Quantity right";
    version_date = "199703100900";
    DefaultRank = 0;
    DefaultDimension = UNDEFINED;
    DefaultDatatype = "DT_FLOAT";
    DefaultType_size = UNDEFINED;
    DefaultMqName = "TEMP_RIGHT";
    DefaultUnit = 2; /* Ref to instance element of Unit */
    Predecessor = 1; /* Ref to instance element of Quantity */
    Successors = UNDEFINED; /* Refs to instance elements of Quantity */
    QuantityGroup = UNDEFINED; /* Ref to instance els.of QuantityGroup */
ENDINSTELEM;

```

INSTELEM Quantity

```

    QName = "SPTNR";
    QId = 4;
    QVersion = "1.1";
    QDescription = "System point number";
    version_date = "199703100900";
    DefaultRank = 0;
    DefaultDimension = UNDEFINED;
    DefaultDatatype = "DT_LONG";
    DefaultType_size = UNDEFINED;
    DefaultMqName = "SPTNR";
    DefaultUnit = 1; /* Ref to instance element of Unit */
    Predecessor = UNDEFINED; /* Ref to instance element of Quantity */
    Successors = UNDEFINED; /* Refs to instance elements of Quantity */
    QuantityGroup = UNDEFINED; /* Ref to instance els.of QuantityGroup */
ENDINSTELEM;

```

INSTELEM Quantity

```

    QName = "N";
    QId = 5;
    QVersion = "1.1";
    QDescription = "Revolution";
    version_date = "199703100900";
    DefaultRank = 0;
    DefaultDimension = UNDEFINED;
    DefaultDatatype = "DT_FLOAT";
    DefaultType_size = UNDEFINED;
    DefaultMqName = "N";
    DefaultUnit = 4; /* Ref to instance element of Unit */
    Predecessor = UNDEFINED; /* Ref to instance element of Quantity */
    Successors = UNDEFINED; /* Refs to instance elements of Quantity */
    QuantityGroup = UNDEFINED; /* Ref to instance els.of QuantityGroup */

```

ENDINSTELEM;

INSTELEM Quantity

```

    QName = "MD";
    QId = 6;
    QVersion = "1.1";
    QDescription = "Torque";
    version_date = "199703100900";
    DefaultRank = 0;
    DefaultDimension = UNDEFINED;
    DefaultDatatype = "DT_FLOAT";
    DefaultType_size = UNDEFINED;
    DefaultMqName = "MD";
    DefaultUnit = 6; /* Ref to instance element of Unit */
    Predecessor = UNDEFINED; /* Ref to instance element of Quantity */
    Successors = UNDEFINED; /* Refs to instance elements of Quantity */
    QuantityGroup = UNDEFINED; /* Ref to instance els.of QuantityGroup */

```

ENDINSTELEM;

INSTELEM Quantity

```

    QName = "PL";
    QId = 7;
    QVersion = "1.1";
    QDescription = "Pressure left";
    version_date = "199703100900";
    DefaultRank = 0;
    DefaultDimension = UNDEFINED;

```

```

    DefaultDatatype = "DT_FLOAT";
    DefaultType_size = UNDEFINED;
    DefaultMqName = "PL";
    DefaultUnit = 7; /* Ref to instance element of Unit */
    Predecessor = UNDEFINED; /* Ref to instance element of Quantity */
    Successors = UNDEFINED; /* Refs to instance elements of Quantity */
    QuantityGroup = UNDEFINED; /* Ref to instance els.of QuantityGroup */
ENDINSTELEM;
/*
//-----
//
// Instance elements of the application elements Engine and Test
// from base type AoTest and AoSubTest
//
//-----
*/
INSTELEM Engine
    EngineName = "TestEngine 1";
    EngineId = 1;
    EngineVersion = "A 1.0";
    EngineDescription = "The first test engine";
    CreateDate = "19971209141345";
    EngineType = "ABC4711"; /* Hel */
    bore = 92.;
    cylindernumber = 6;
    stroke = 95.3;
    SubTests = 1; /* Ref to instance element of Test
    EngineUser = 21; /* Ref to instance element of User
ENDINSTELEM;

INSTELEM Test
    TestName = "Test-Configuration 1";
    TestId = 1;
    TestVersion = "B20";
    TestDescription = "The first test of engine 1";
    CreateDate = "19971210101422";
    TestType = "Functionality Test";
    TestComment = "Test: torque and at special points temperature";
    exhaust = "System 4711";
    air_filter = "C 12.1";
    Measurements = 1,2; /* Refs to instance elements of Measurement */
    MainTest = 1; /* Ref to instance element of Engine */
    TestUser /* Hel /= 22; /*Ref to instance element of User */

```



```

ENDINSTELEM;
/*
//-----
//
// Instance elements of the application structure part with
// Measurement, MeaQuantity, Submatrix, LocalColumn with value_sequence
// for the first measurement
//
//-----
*/
INSTELEM Measurement
    MeaName = "M001A";
    MeaId = 1;
    MeaVersion = "A1";
    MeaDescription = "1. Measurement ";
    MeasurementBegin = "19971210141345";
    MeasurementEnd = "19971210162135";
    CreateDate = "19971210165331"; /* Hel */
    MeaType = "TLM";
    MeaComment = "No Problem";
    ROZ = UNDEFINED;
    Test = 1; /* Ref to instance element of Test */
    MeaQuantities = 1,2,3; /* Ref to instance element of MeaQuantity */
    Submatrices = 1; /* Ref to instance element of Submatrix */
ENDINSTELEM;

INSTELEM MeaQuantity
    MeaQName = "N";
    MeaQId = 1;
    MeaQVersion = "D1";
    MeaQDescription = "Number of revolutions";
    Rank = 0;
    Dimension = UNDEFINED;
    DataType = "DT_FLOAT";
    TypeSize = UNDEFINED;
    Minimum = 500.;
    Maximum = 3200.;
    Average = 1850.;
    Measurement = 1; /* Ref to instance element of Measurement */
    LocalColumns = 1; /* Refs to instance elements of LocalColumn */
    Quantity = 5; /* Ref to instance element of Quantity */
    Unit = 4; /* Ref to instance element of Unit */
ENDINSTELEM;

```

```

INSTELEM MeaQuantity
    MeaQName = "MD";
    MeaQId = 2;
    MeaQVersion = "D";
    MeaQDescription = "Torque";
    Rank = 0;
    Dimension = UNDEFINED;
    DataType = "DT_FLOAT";
    Minimum = 150.;
    Maximum = 255.;
    Average = 190.;
    SDeviation = 22.2;
    Measurement = 1; /* Ref to instance element of Measurement */
    LocalColumns = 2; /* Refs to instance elements of LocalColumn */
    Quantity = 6; /* Ref to instance element of Quantity */
    Unit = 6; /* Ref to instance element of Unit */
ENDINSTELEM;

```

```

INSTELEM MeaQuantity
    MeaQName = "PL";
    MeaQId = 3;
    MeaQVersion = "D";
    MeaQDescription = "Pressure ";
    Rank = 0;
    Dimension = UNDEFINED;
    DataType = "DT_FLOAT";
    Minimum = 991.2;
    Maximum = 993.2;
    Average = 992.0;
    Measurement = 1; /* Ref to instance element of Measurement */
    LocalColumns = 3; /* Refs to instance elements of LocalColumn */
    Quantity = 7; /* Ref to instance element of Quantity */
    Unit = 7; /* Ref to instance element of Unit */
ENDINSTELEM;

```

```

INSTELEM Submatrix
    name = "M001A";
    id = 1;
    version = "1a";
    description = "1. Measurement for data transfer via ASAM";
    version_date = UNDEFINED;
    measurement = 1;

```

```

    number_of_rows = 15;
    local_columns = 1,2,3; /* Refs to instance elements of LocalColumn */
ENDINSTELEM;

```

```
INSTELEM LocalColumn
```

```

    name = "N";
    id = 1;
    flags = 0;
    global_flag = 0;
    independent = TRUE;
    minimum = 500.;
    maximum = 3200.;
    measurement_quantity = 1; /* Ref to instance element of MeaQuantity */
    submatrix = 1; /* Ref to instance element of Submatrix */
    values = DATATYPE DT_FLOAT
            ,500.,700.,900.,1100.,1300.,1400.,1600.,1800.,2000.
            ,2200.,2400.,2600.,2800.,3000.,3200.;

```

```
ENDINSTELEM;
```

```
INSTELEM LocalColumn
```

```

    name = "MD";
    id = 2;
    flags = 0;
    global_flag = 0;
    independent = FALSE;
    minimum = 150.;
    maximum = 255.;
    measurement_quantity = 2; /* Ref to instance element of MeaQuantity */
    submatrix = 1; /* Ref to instance element of Submatrix */
    values = DATATYPE DT_FLOAT
            ,150.,160.,170.,180.,190.,200.,210.,220.
            ,230.,240.,250.,255.,250.,230.,190.;

```

```
ENDINSTELEM;
```

```
INSTELEM LocalColumn
```

```

    name = "PL";
    id = 3;
    flags = 0;
    global_flag = 0;
    independent = FALSE;
    minimum = 991.2;
    maximum = 993.2;
    measurement_quantity = 3; /* Ref to instance element of MeaQuantity */

```

```

        submatrix = 1; /* Ref to instance element of AoSubmatrix */
        values = DATATYPE DT_FLOAT
                , 991.2, 991.7, 991.5, 991.9, 992.1, 987.6, 992.2, 993.
                , 993.2, 993.2, 992.7, 993., 993., 992.7, 992.7;
ENDINSTELEM;
/*
//-----
//
// Instance elements of the application structure part with
// Measurement, MeaQuantity, Submatrix, LocalColumn with value_sequence
// for the second measurement
//
//-----
*/
INSTELEM Measurement
    MeaName = "M002A";
    MeaId = 2;
    MeaVersion = "A1";
    MeaDescription = "2. Measurement ";
    MeasurementBegin = "19971211090412";
    MeasurementEnd = "19971211121145";
    CreateDate = "19971211121522"; /* Hel */
    MeaType = "VLM";
    MeaComment = "No problem";
    ROZ = UNDEFINED;
    Test = 1; /* Ref to instance element of Test */
    MeaQuantities = 4,5,6,7; /* Ref to instance element of MeaQuantity */
    Submatrices = 2,3; /* Ref to instance elements of Submatrix */
ENDINSTELEM;

INSTELEM MeaQuantity
    MeaQName = "SPTNR";
    MeaQId = 4;
    MeaQVersion = "A";
    MeaQDescription = "System point number ";
    Rank = 0;
    Dimension = UNDEFINED;
    DataType = "DT_LONG";
    Minimum = 1.0;
    Maximum = 15.;
    Average = 7.5;
    Measurement = 2; /* Ref to instance element of Measurement */
    LocalColumns = 4,7; /* Ref to instance elements of LocalColumn */

```

```

    Quantity = 4; /* Ref to instance element of Quantity */
    Unit = 1; /* Ref to instance element of Unit */
ENDINSTELEM;

INSTELEM MeaQuantity
    MeaQName = "N";
    MeaQId = 5;
    MeaQVersion = "D";
    MeaQDescription = "Revolution ";
    Rank = 0;
    DataType = "DT_FLOAT";
    Minimum = 500.;
    Maximum = 3200.;
    Average = 1850.;
    Measurement = 2; /* Ref to instance element of Measurement */
    LocalColumns = 5; /* Refs to instance elements of LocalColumn */
    Quantity = 5; /* Ref to instance element of Quantity */
    Unit = 4; /* Ref to instance element of Unit */
ENDINSTELEM;

INSTELEM MeaQuantity
    MeaQName = "MD";
    MeaQId = 6;
    MeaQVersion = "D";
    MeaQDescription = "Torque ";
    Rank = 0;
    Dimension = UNDEFINED;
    DataType = "DT_FLOAT";
    Minimum = 152.;
    Maximum = 257.;
    Average = 194.;
    Measurement = 2; /* Ref to instance element of Measurement */
    LocalColumns = 6; /* Refs to instance elements of LocalColumn */
    Quantity = 6; /* Ref to instance element of Quantity */
    Unit = 6; /* Ref to instance element of Unit */
ENDINSTELEM;

INSTELEM MeaQuantity
    MeaQName = "TEMP1";
    MeaQId = 7;
    MeaQVersion = "D";
    MeaQDescription = "Temperature ";
    Rank = 0;

```

```

    Dimension = UNDEFINED;
    DataType = "DT_FLOAT";
    Minimum = 150.;
    Maximum = 183.;
    Average = 169.;
    Measurement = 2; /* Ref to instance element of Measurement */
    LocalColumns = 8; /* Refs to instance elements of LocalColumn */
    Quantity = 2; /* Ref to instance element of Quantity */
    Unit = 2; /* Ref to instance element of Unit */
ENDINSTELEM;

INSTELEM Submatrix
    name = "M002A";
    id = 2;
    version = "1a";
    description = "2. Measurement for data transfer via ASAM";
    version_date = UNDEFINED;
    measurement = 2;
    number_of_rows = 15;
    local_columns = 4,5,6; /* Refs to instance elements of LocalColumn */
ENDINSTELEM;

INSTELEM LocalColumn
    name = "SPKTNR";
    id = 4;
    flags = 0;
    global_flag = 0;
    independent = TRUE;
    minimum = 1;
    maximum = 15;
    measurement_quantity = 4; /* Ref to instance element of MeaQuantity */
    submatrix = 2; /* Ref to instance element of Submatrix */
    values = DATATYPE DT_SHORT
        ,1, 2, 3, 4, 5, 6, 7, 8
        ,9,10,11,12,13,14,15;
ENDINSTELEM;

INSTELEM LocalColumn
    name = "N";
    id = 5;
    flags = 0;
    global_flag = 0;
    independent = FALSE;

```

```

    minimum = 500.;
    maximum = 3200.;
    measurement_quantity = 5; /* Ref to instance element of MeaQuantity */
    submatrix = 2; /* Ref to instance element of Submatrix */
    values = DATATYPE DT_FLOAT
            ,500.,700.,900.,1100.,1300.,1400.,1600.,1800.
            ,2000.,2200.,2400.,2600.,2800.,3000.,3200.;
ENDINSTELEM;

INSTELEM LocalColumn
    name = "MD";
    id = 6;
    flags = 0;
    global_flag = 0;
    independent = FALSE;
    minimum = 150.;
    maximum = 255.;
    measurement_quantity = 6; /* Ref to instance element of MeaQuantity */
    submatrix = 2; /* Ref to instance element of Submatrix */
    values = DATATYPE DT_FLOAT
            ,150.,160.,170.,180.,190.,200.,210.,220.
            ,230.,240.,250.,255.,250.,230.,190.;
ENDINSTELEM;

INSTELEM Submatrix
    name = "M002B";
    id = 3;
    version = "1a";
    description = "2. Measurement for data transfer via ASAM";
    version_date = UNDEFINED;
    measurement = 2;
    number_of_rows = 3;
    local_columns = 7,8; /* Refs instance elements of LocalColumn */
ENDINSTELEM;

INSTELEM LocalColumn
    name = "SPKTRN";
    id = 7;
    flags = 0;
    global_flag = 0;
    independent = TRUE;
    minimum = 5;
    maximum = 15;

```

```

        measurement_quantity = 4; /* Ref to instance element of MeaQuantity */
        submatrix = 2; /* Ref to instance element of Submatrix */
        values = DATATYPE DT_SHORT,5,10,15;
ENDINSTELEM;

INSTELEM LocalColumn
    name = "TEMP1";
    id = 8;
    flags = 0;
    global_flag = 0;
    independent = FALSE;
    minimum = 150.;
    maximum = 183.;
    measurement_quantity = 7; /* Ref to instance element of MeaQuantity */
    submatrix = 2; /* Ref to instance element of Submatrix */
    values = DATATYPE DT_FLOAT,150.,174.,183.;
ENDINSTELEM;
/*
//-----//
// instance elements of the application elements User and UserGroup
// and ACLs for security control of application elements and instances.
//
//-----*/
INSTELEM User
    UserName = "Peter Sellers";
    UserId = 21;
    UserVersion = "1A";
    UserDescription = "Super user group 1";
    version_date = "199708101105";
    UserDepartment = "OP/EFG";
    AliasName = "PS";
    password = "^1kjws7hxoish^mjkkdjxoöe--@@@kjg798xh0880djdj90";
    groups = 30,50;
ENDINSTELEM;
INSTELEM User
    UserName = "Todd Martin";
    UserId = 22;
    UserVersion = "1B";
    UserDescription = "Sub user group 1b";
    version_date = "199708101105";
    UserDepartment = "OP/ABC";
    AliasName = "TM";
    password = "8snn236ikl= (/ ($mm22&/&) () NBT/ (nger%/0!cv3kjkkkj";

```



```

    groups = 30,40;
ENDINSTELEM;
INSTELEM User
    UserName = "Otto Biermann";
    UserId = 23;
    UserVersion = "1A";
    UserDescription = "Super user group 1";
    version_date = "199708101105";
    UserDepartment = "OP/EFG";
    AliasName = "OB";
    password = "89xzhun3m,ö)) ?imi/U (/%unpo39048ejdkßdksosüüöü3";
    groups = 40,50;
ENDINSTELEM;
INSTELEM UserGroup
    GroupName = "AQ 1";
    GroupId = 30;
    GroupVersion = "00";
    GroupDescription= "Group 1";
    version_date = "199708101105";
    superuser_flag = FALSE;
    users = 21,22;
ENDINSTELEM;
INSTELEM UserGroup
    GroupName = "AQ 2";
    GroupId = 40;
    GroupVersion = "00";
    GroupDescription= "Group 2";
    version_date = "199708101105";
    superuser_flag = FALSE;
    users = 22,23;
ENDINSTELEM;
INSTELEM UserGroup
    GroupName = "TZU 23";
    GroupId = 50;
    GroupVersion = "00";
    GroupDescription= "Group 3";
    version_date = "199708101105";
    superuser_flag = TRUE;
    users = 21,23;
ENDINSTELEM;
INSTELEM ACLA
    users = REF_TYPE UserGroup 30;
    appl_element_name = "Engine";

```

```

    rights = 7;
    attribute_name = UNDEFINED; // protection of whole application element
ENDINSTELEM;
INSTELEM ACLA
    users = REF_TYPE UserGroup 30;
    appl_element_name = "Quantity";
    rights = 3;
    attribute_name = "description"; // protection of a specific attribute
ENDINSTELEM;
INSTELEM ACLA
    users = REF_TYPE UserGroup 40;
    appl_element_name = "Measurement";
    rights = 10;
    attribute_name = UNDEFINED; // protection of whole application element
ENDINSTELEM;
INSTELEM ACLI
    users = REF_TYPE UserGroup 50;
    appl_element_name = "MeaQuantity";
    rights = 1;
    instance_id = 3; // protection of an instance element
ENDINSTELEM;
INSTELEM ACLI
    users = REF_TYPE UserGroup 40;
    appl_element_name = "MeaQuantity";
    rights = 3;
    instance_id = 5; // protection of an instance element
ENDINSTELEM;
/*
//-----
// Initial rights (ACL templates) and their location
// and security level
//-----*/
INSTELEM ACLTemplate
    users = REF_TYPE UserGroup 30;
    appl_element_name = "Measurement";
    rights = 3;
    instance_id = 2;
    ref_appl_elem_name= "Test";
ENDINSTELEM;
INSTELEM ACLTemplate
    users = REF_TYPE UserGroup 40;
    appl_element_name = "Measurement";
    rights = 1;

```

```

        instance_id = 2;
        ref_appl_elem_name= "Test";
    ENDINSTELEM;
    INSTELEM InitialRightsAt
        appl_element_name = "Measurement";
        attribute_name = "Test";
    ENDINSTELEM;
    INSTELEM SecurityLevel
        appl_element_name = "Measurement";
        level = 7; /* all or specific instances and even attributes */
    ENDINSTELEM;
    INSTELEM SecurityLevel
        appl_element_name = "Engine";
        level = 1; /* only all instances */
    ENDINSTELEM;

    ATF_END;

```

10 传输格式 (ATF/XML)

10.1 ATF/XML 的概述

10.1.1 介绍

ASAM ODS 规范要求文件储存系统允许测量数据独立于数据库存储。

ASAM ODS 版本 5.0 之前, 这样的文件存储系统以经典的 ASAM 传输格式为代表 (指的是 ATF/CLA, 目前发布的版本为 1.4.1)。ASAM ODS 5.0 版本对 ATF/XML 做了介绍, ATF/XML 是基于 XML 语言的 ATF 文件格式。与之前的传输格式 ATF/CLA 一样, ATF/XML 使整个环境中或者不同的计算机和环境之间部分的数据交换成为可能。除了以 XML 语言为基础的细微差别外, ATF/XML 和 ATF/CLA 这两种传输模式功能非常相似, 甚至可以将它们看成是 ASAM 标准中的两种不同“风格”。

ATF/XML 不仅能够传输所需结构和实例信息, 也能够传输各自的数据, 并能以标准化的格式存储所有的信息。

本章描述和解释 ASAM 中的 XML 传输格式 (ATF/XML)。

10.1.2 ATF/XML 和 ATF/CLA 的比较

ATF/CLA 和 ATF/XML 这两种规范都基于 ODS 基础模型。这种基础模型描述基本数据类型和在有效的 ODS 应用模型中使用的更为复杂的结构。所有的 ATF/CLA 和 ATF/XML 文件都使用基础模型中的组件来充分描述应用模型和实例数据, 这样它们就能够为任意的 ODS 兼容系统理解。然而这两种格式在确定模型和实例数据上有细微的差别。

ATF/CLA 和 ATF/XML 最大的区别是, ATF/CLA 使用非常详细的规范来定义文件结构, 而 ATF/XML 主要依赖 W3C XML 规范定义的文件结构。ATF/XML 文件必须是形式合法的有效的 XML 文件, 也必须能由任

意的 XML 验证解析器来检测其结构有效性。的确，正是 ATF/XML 文件的这种属性和 XML 处理工具的扩散推动了 ATF/XML 文件格式的发展。此外，它也可以构造 XML 模式文件，这种 XML 模式文件与验证解析器结合使用，能够对 ATF/XML 文件进行更加全面的验证。

ATF 两种传输格式的另一个显著差异是，ATF/XML 使用 Unicode 字符编码。Unicode 字符集是世界通用的 16 位字符集，用于实施 ASCII 字符集的扩展集。而 ATF/CLA 文件使用的是 255 个不同字符(ASCII 字符及其补充)，由于硬件/软件配置不同，只有子集能确保获得支持。因此，在 ATF/CLA 中，例如德国 umlauts 这样的字符可以使用转义符。

ATF 两种传输格式还有些细微差别：

- ATF/CLA 中，派生自基本属性的属性可以申明与派生基本属性的基础元素不同的数据类型。而 ATF/XML 则要求必须与基础元素保持一致；
- 在 ATF/CLA 中，应用属性和与其它元素的应用关系大体上是以同种方式定义。ATF/XML 则对应用属性，应用关系和实例属性进行说明；
- ATF/CLA 和 ATF/XML 都十分依赖基础模型。然而，ATF/XML 提供了基础架构，基础架构可以用来进行基础验证和更加全面的验证；
- ATF/XML 含有确定 ATF/XML 文件信息的附加文件部分，附加文件确定 ATF/XML 文件的创建方式和时间；
- 在 ATF/CLA 文件中，列表和字符串序列用逗号来分隔。ATF/XML 是因 XML 语言的约束而功能受制，因其数值由空格分隔，这阻止了单个数值块存储一个字符串序列。而内部 XML 元素序列则是不可缺少的；
- ATF/XML 文件能够处理最新扩展的 ODS 基础模型--也就是字符串长度，必选标志，自动生成标志和唯一标志。这些扩展部分并没有加入到 ATF/CLA 规范中；
- 当确定 ATF/CLA 文件中的实例信息时，每个元素的实例次序和属性次序都是任意的。在 ATF/XML 中元素和属性都可以默认为有任意的安排，也可以创造外部模式，让外部模式严格执行元素和属性的顺序；
- ATF/CLA 仅通过指定属性的名称，数据类型和值就可对实例属性进行内嵌声明。ATF/XML 需要对包含所有实例属性元素的实例属性块进行明确声明。每个实例属性都要声明它的名称，数据类型和值。ATF/XML 数据类型的声明机制不同于 ATF/CLA，它允许筛选数据类型（如果有的话）的应用模式被写入；
- ATF/CLA 提供可接受的自定义数据类型，ATF/XML 以潜在的 XML 模式数据类型为组件来构建更加复杂的数据类型；
- ATF/CLA 提供自己的空格规则，ATF/XML 使用 XML 空格规范，空格是来自 0x9（水平制表符），0xA（换行），0xd（回车）和 0x20 的（空格）集合中的一个或多个字符；
- ATF/CLA 提供自己的注释规则，ATF/XML 使用 XML 注释规范；
- ATF/CLA 提供自己的标识符规则，ATF/XML 元素名称和属性标记名称使用 XML 规范。这会 产生一个问题，因为 XML 规范明确地排除标记名称中的字符如空格。因此标记名称的某些字符必须被 ATF/XML 特定的转义序列转换；
- ATF/CLA 提供自己的规则来定义存储测量数据的外部文件和文档中的内容。ATF/XML 使用相

似的规则来定义存储测量数据外部文件,但是将其它文档中的内容纳入XML文件中时使用XML内容规范;

- ATF/CLA 允许用户定义枚举的内嵌声明。如果此枚举仅使用一次,那么这是有效的,但是如果多个元素重复使用枚举,那就是无效的。ATF/XML 允许用户定义枚举定义为单独的元素,这些元素称为 1-n 应用元素;
- ATF/CLA 允许用户定义枚举从零开始的索引整型表示。这允许实例仅仅指定适当的枚举项索引。ATF/XML 不允许这些整型表示。实例必须明确陈述期望的枚举值(字符串);
- 当定义元素之间的关系时,ATF/XML 允许定义相反关系名称。

ATF/CLA 和 ATF/XML 有如此多共同之处并非偶然的。与其认为 ATF/CLA 和 ATF/XML 是两个单独的文件传输格式,不如将它们看成是同种格式的两种不同“特色”,熟悉 ATF/CLA 的人会立即意识到 ATF/XML 的相似之处。下面是在 ATF/XML 和 ATF/CLA 中定义的不同实例的例子。.

示例: ATF/CLA 和 ATF/XML 的比较。相同的实例分别用 ATF/CLA 和 ATF/XML 表达如下。

ATF/XML 表达:

```
<Engine>
  <EngineId>1</EngineId>
  <EngineName>Test Engine 1</EngineName>
  <EngineDescription>The first testengine</EngineDescription>
  <EngineVersion>A 1.0</EngineVersion>
  <CreateDate>19971209141345</CreateDate>
  <EngineType>ABC47111</EngineType>
  <bore>92.</bore>
  <cylindernumber>6</cylindernumber>
  <stroke>95.3</stroke>
  <SubTests>1</SubTests>
  <EngineUser>21</EngineUser>
</Engine>
```

ATF/CLA 表达:

```
INSTELEM Engine
  EngineId = 1;
  EngineName = "TestEngine 1";
  EngineDescription = "The first testengine";
  EngineVersion = "A 1.0";
  VersionDate = UNDEFINED
  CreateDate = "19971209141345";
```

```

EngineType = "ABC4711";

bore = 92.;

cylindernumber = 6;

stroke = 95.3;

SubTests = 1;

EngineUser = 21;

ENDINSTELEM;

```

10.1.3 ATF/XML 中数据类型的使用

ASAM ODS 使用的数据类型的简要介绍见本标准的第 5.5 节。注意,ATF/XML 文件被编码为 UTF8,任意的字符串型数据将采用此种编码方式。然而,其它的 ASAM ODS 组件对字符串型数据都有限制,这种数据类型在 5.5 节中已进行描述,而且,通过 ATF/XML 传输的信息必须符合这种数据类型的定义。

因为 ATF/XML 是和 ASAM ODS 版本 5.0 同步引入的新标准,所以只要合适的 ASAM 数据类型可用,它都会使用此种数据类型。由于 ATF/XML 已经写入新的解析器,因此没有兼容性的问题。这样就可以将 ASAM 数据类型直接映射到已经在诸如 OO-API 或者 RPC-API 使用的 ASAM ODS 数据类型。

这有一些数据类型,它们在 ASAM 数据类型中不存在;在这种情况下,采用 ASAM ODS 数据类型,它们是:

- T_DATE: 一种基于数据类型的 ASCII 码字符串,它指定有无限时间分辨率的任何日期和时间信息。
- T_EXTERNALREFERENCE: 一种包含三种字符串(说明, MIME 类型和位置(例如 URL))的数据类型,当用 ATF/XML 传输时,每一种字符串都被封入标签;它是用来引用 ODS 服务器之外的项目的。

ATF/XML 文件中使用的数据类型将在 ASAM 数据类型规范和 ASAM ODS 规范的第 5.5 节中确定和描述;13.6.1 节给出了用来指定统一数据类型的相关 IDL 文件。

所有使用的数据类型都进一步映射到 XML 标准数据类型,以便给标准的 XML 解析器提供足够信息进行有效性检查。它们是通过使用本机和扩展的 XML 模式数据类型实施的。对于基础模型中定义的数据类型,在 XML 模式中有相关的数据类型。而对于其它的,可以使用由 XSD 提供的基元创建更复杂的数据类型。这些映射根据 ASAM e.V 的要求在 XML 模式"ASAM_HDTypes.xsd"中给出,并且在 10.5.1 节中列出。

尽管许多 ODS 相关数据类型已经被替代了,但是它们仍然涉及到枚举数据类型名称。

10.1.4 基础元素, 属性, 关系和枚举的名称

这些实体的名称在基础模型中指定,基础模型解释它们时要区分大小写。

然而,在 ATF/XML 文件中这种自由度是不存在的。基础元素,基本属性和基本关系的枚举名称必须区分大小写,确切的说,就像在基础模型规范中给出的那样。否则,10.5 节给出的与 XML 模式文件对立的验证将是不可能的。

10.1.5 命名 ATF/XML 文件

建议给 ATF/XML 文件使用'atfx'文件扩展名。

对于包含 ATF/XML 信息的所有文件，允许分配一个特定的应用程序来作为标准的应用程序。

10.2 ATF/XML 文件

本节描述了 ATF/XML 文件的内容。在概述了完整结构和将信息分配到不同物理文件的可能性之后，由 ASAM ODS 指定的文件中的每一个 XML 元素都会被详细描述。

10.2.1 ATF/XML 文件的整体结构

ATF/XML 文件是逻辑 XML 文件，它包含一个或多个形式合法的 XML 文件。ATF/XML 文件可能包含测量数据或包含带有测量数据的外部文件指标。合乎逻辑的 ATF/XML 文件是自我定义的，这意味着应用模型和实例数据一起包含在这个文件中。图 101 给出了 ATF/XML 文件的整体结构。

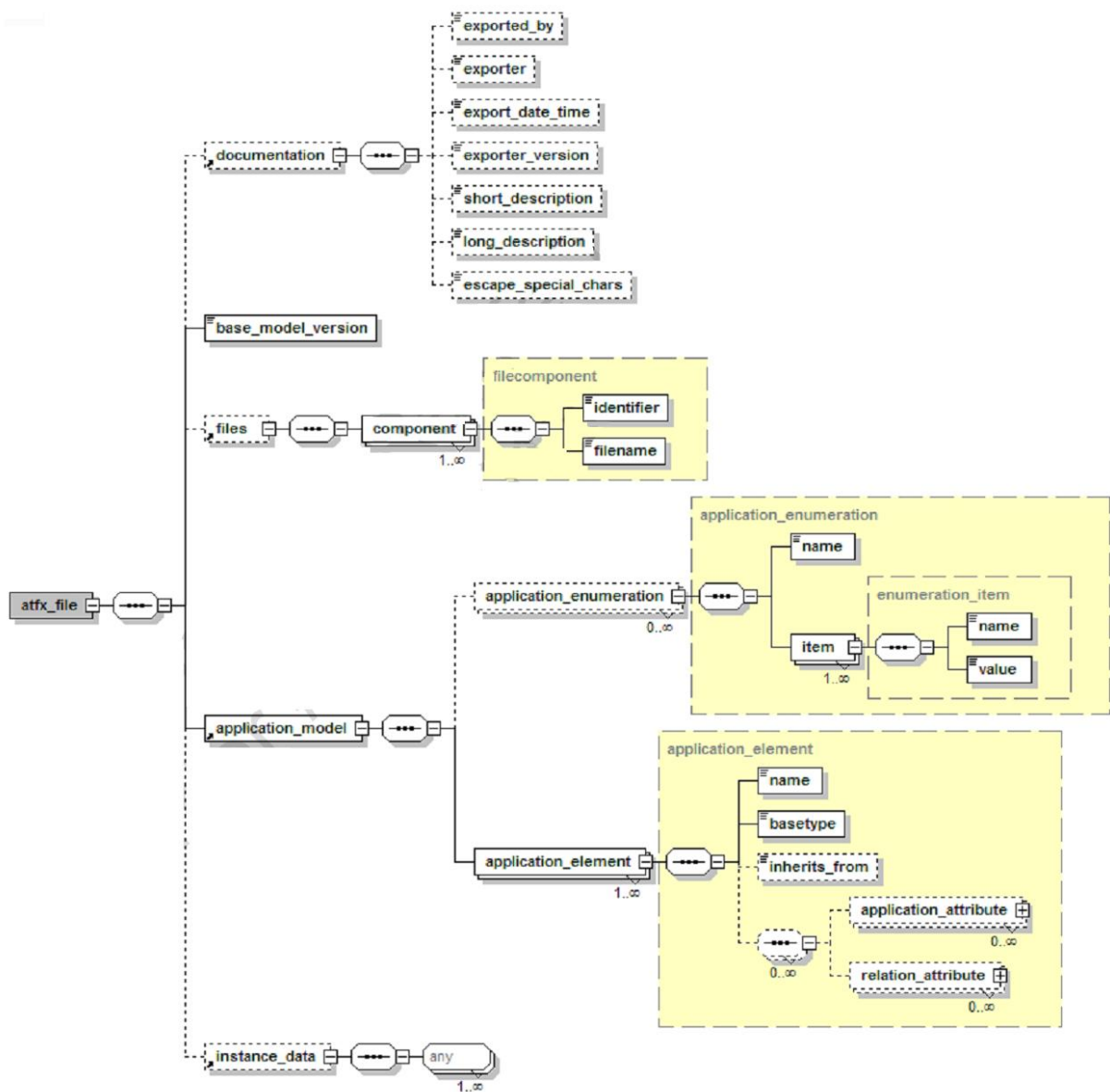


图101 ATF/XML 文件的整体结构

ATF 文件（像 XML 文件）以 XML 声明开头。紧接着的是 XML 元素的 ASAM ODS 特定集合，一系列文件，基础模型信息，外部文件组件的描述，应用模型元数据和实例数据。在它们当中，只有基础模型信息和应用模型元数据是必须的。

ATF/XML 文件可能包含 XML 注释，这些注释包含在任意的 ATF/XML 元素之间。XML 注释以 '<!--' and end with '-->' 开始。它们必须符合 XML 注释规范。

10.2.2 逻辑和物理 ATF/XML 文件

逻辑 ATF 文件由多个物理文件组成。ATF/XML 文件和它们的经典 ATF/CLA 副本提供可以自我包含的机会，也提供被分成多个物理文件的可能性，而这些物理文件又被处理应用程序重新组装成一个逻辑文件。这一能力允许 ATF 设计者将这些文件分割成可重复利用，并且与通常使用的组件一致的组件

式程序库。它也允许大量的测量数据保存在原有格式文件中。

ATF/XML 中多个物理文件的使用和 ATF/CLA 文件的使用有轻微的不同。对于包括物理文件，有两种不同的机制。使用方法取决于包含在物理文件中的信息类型。

如果物理文件包含二进制或者其它测量数据，XML XInclude 就不能使用，而且处理应用程序必须获得本身文件。在这种情况下，这些包含测量数据的文件称为“组件文件”，它们的结构以及在 ATF/XML 文件中如何使用它们将在本章后面进行解释；

如果物理文件包含有效的 XML 片段，那么 XML XInclude 会被使用，XML 解析器会在处理时自动包括文件。

XInclude 是一个最新的 W3V 规范，W3V 规范为建立多个形式合法的 XML 文档之外的大量 XML 文件而制定，并且与验证无关。每一块都是一个完整的 XML 文档或是一个不完整的 XML 文档。这一能力十分适合 ATF/XML 开发者，这些开发者希望开发出包括 XML 文件片段的标准程序库。典型地，应用元数据可以在一个或多个 XML 文件中定义，并在使用时包括这些 XML 文件。

为了提高再利用率和模块性，XInclude 是一种构造新 XML 文档的技术，这些文档来自使用简单包括机制的现存文档。

<http://www.w3.org/2001/XInclude> 是官方的 XInclude 命名空间。

应注意的是，XInclude 是最近添加到 XML 标准中的。它还没有完全被 XML 工具提供者支持。

示例：XINCLUDE。

两个 ATF/XML 文件，分别名为“xitest.xml”和“measurement.xml”，结果文件名为“result.xml”，与从 ATF/XML 文件提取的内容一致（语法分析器将分别组合这两个文件）。

xitest.xml 的内容：

```
<?xml version="1.0" encoding="UTF-8"?>
<atfx_file
  version="atfx_file: V1.2.0"
  xmlns="http://www.asam.net/ODS/5.2.0/Schema"
  xmlns:xi="http://www.w3.org/2001/XInclude"
  ...
  <application_model>
    <application_element>
      <xi:include href="library/measurement.xml"/>
    </application_element>
  </application_model>
  <instance_data>
    ...
  </instance_data>
</atfx_file>
```

measurement.xml 的内容:

```
<?xml version="1.0" encoding="UTF-8"?>
<name>Measurement</name>
<basetype>AoMeasurement</basetype>
...
<application_attribute>
  <name>MeaName</name>
  <base_attribute>name</base_attribute>
  <unique>true</unique>
  <length>50</length>
</application_attribute>
...
<relation_attribute>
  <name>MeaQuantities</name>
  <ref_to>MeasurmentQuantity</ref_to>
  <base_relation>measurement_quantities</base_relation>
  <min_occurs>0</min_occurs>
  <max_occurs>1</max_occurs>
</relation_attribute>
```

result.xml 的内容:

```
<?xml version="1.0" encoding="UTF-8"?>
<atfx_fileversion="atfx_file: V1.2.0"
  xmlns="http://www.asam.net/ODS/5.2.0/Schema"
  xmlns:xi="http://www.w3.org/2001/XInclude"
  ...
  <application_model>
    <application_element>
      <name>Measurement</name>
      <basetype>AoMeasurement</basetype>
      ...
    <application_attribute>
      <name>MeaName</name>
      <base_attribute>name</base_attribute>
      <unique>true</unique>
      <length>50</length>
```

```

</application_attribute>
...
<relation_attribute>
    <name>MeaQuantities</name>
    <ref_to>MeasurmentQuantity</ref_to>
    <base_relation>measurement_quantities</base_relation>
    <min_occurs>0</min_occurs>
    <max_occurs>1</max_occurs>
</relation_attribute>
</application_element>
</application_model>
<instance_data>
...
</instance_data>
</atfx_file>

```

如果没有明确说明，则“ATF/XML”文件通常是指逻辑 ATF/XML 文件。

10.2.3 ATF/XML 文件的 XML 声明

与其它任何 XML 文件一样，ATF/XML 文件以 XML 声明开头：

```
<?xml version="1.0" encoding=UTF-8"?>
```

本行明确了 XML 标准的版本（现在的版本是 1.0），并且指定了文件采用的字符编码。

对于通常用来定义应用模型和实例数据的不同语言而言，XML 文件的编码将是 Unicode 格式“UTF-8”。在使用 ISO/IEC 8859-1 字符集合以外的字符时，应用程序必须小心，因为符合 ASAM ODS 规范的工具可能不能进一步处理它们。

10.2.4 根元素<ATFX_FILE>

XML 文件的根元素是所有其它元素的容器，它的 XML 内容是一组序列元素<documentation>、<base_model_version>、<files>、<application_model>和<instance_data>。每一个元素可能出现一次。它们当中只有<base_model_version>和<application_model>是必选的，其它的是可以省略的。这些元素在本节后面做深入的描述。

根元素有一些 XML 属性，这些属性描述它所附加的 ASAM ODSATF/XML 规范的版本，并且声明文档中使用的命名空间。最重要的是：

1) VERSION 属性

根据每个 ASAM XML 书写风格指南，版本属性指定建立 XML 文件的 ATF/XML 语言版本，该属性的格式是：

```
atfx_file: V<Major Vesion Nr>.<Minor Version Nr>.<Revision Nr>
```

- 主要版本号（数值： [0-9]+）；
- 次要版本号（数值： [0-9]+）；
- 修订版版本号（数值： [0-9]+）。

因此目前 ATF/XML 版本的版本标准是'atfx_file: V1.2.0'。这个 XML 属性是必选的。

2) 命名空间属性 XMLNS 和模式位置

这些属性通过验证 XML 工具来决定 XML 文件中的 XML 元素的命名空间。如果一个元素的命名空间是已知的，并且命名空间的相应 XML 模式定义也给出了，那么适当的 XML 工具不仅能够检查它的形式是否合法也能检查是否有效。这将允许使用各种各样的 XML 工具来判断元素的标记，它的属性以及它的内容是否符合模式定义中给出的元素规范。

ASAM ODS 为 ATF/XML 文件提供模式定义。这将会在 10.3 节作进一步的解释。为了引用这些模式定义，默认命名空间为：

```
xmlns=http: //www.asam.net/ODS/5.2.0/Schema
```

更具体的命名空间可以被指定，例如 ATF/XML 文件中使用的 XInclude 设施：

```
xmlns: xi=http: //www.w3.org/2001/XInclude
```

注：尽管XML允许包括相应XML属性指定模式文件的位置，但是这种情况应该避免。否则，ASAM ODS检查器（ASAM提供）将不能够检查ATF/XML文件是否遵守ASAM ODS规则。

示例：ELEMENT <ATFX_FILE>。

这是一个<atfx_file>元素的开始标记例子。

```
<atfx_fileversion="atfx_file: V1.2.0"
  xmlns="http://www.asam.net/ODS/5.2.0/Schema">
```

10.2.5 元素<DOCUMENTATION>

ATF/XML 文件可能指定它的来源信息以及其它细节。通常通过元素<文档>完成。

它没有 XML 属性，它的 XML 内容是一组元素序列<exported_by>、<exporter>、<export_date_time>、<exporter_version>、<short_description>、<long_description>和<escape_special_chars>。每一个元素可以出现一次。它们都不是必选属性。尽管空的<文档>元素可能会在 ATF/XML 文件中出现，但是在这种情况下，大家通常省略它。

上面包含的元素将在本节后面做详细的描述。

注：此信息既不是来自创建ATF/XML文件的ASAM ODS服务器，也不是ATF/XML文件导入的服务器中。因此它仅存在于 ATF/XML文件中。

10.2.6 元素<EXPORTED_BY>

这个元素没有 XML 属性，它的内容是一串字符串。它指定创建 ATF/XML 文件发起人。没有定义更深入的限制和命名约定。

10.2.7 元素<EXPORTER>

这个元素没有 XML 属性，它的内容是一串字符串。它指定创建 ATF/XML 文件的工具（客户端应用）。如果 ATF/XML 文件是人工创建，那么它取决于创建者要指定什么内容。没有定义更深入的限制和命名约定。

10.2.8 元素<EXPORT_DATE_TIME>

这个元素没有 XML 属性，它的内容是一串字符串。它指定 ATF/XML 文件的创建日期和时间。信息格式必须符合 ASAM ODS T_DATE 格式，为“YYYYMMDDhhmmssllccnnn...”（见 5.5 节，然而省略 0x00 终止符）。

10.2.9 元素<EXPORTER_VERSION>

这个元素没有 XML 属性，它的内容是一串字符串。它指定创建 ATF/XML 文件的工具版本（客户端应用程序）。如果 ATF/XML 文件是人工创建，那么它取决于创建者要指定什么。没有定义更深入的限制和命名约定。

10.2.10 元素<SHORT_DESCRIPTION>

这个元素没有 XML 属性，它的内容是一串字符串。它以简短形式说明 ATF/XML 文件的内容。导入服务器客户端内的典型应用是，当为导入选择文件时，为 ATF/XML 文件集中每个导入文件提供一行信息。没有定义更深入的限制和命名约定。

10.2.11 元素<LONG_DESCRIPTION>

这个元素没有 XML 属性，它的内容是一串字符串。在稍微详细地描述了这些 ATF/XML 文件的内容，并且提供想了解 ATF/XML 文件的所有必要信息，但不包含在这个文件中的某个地方的信息。没有定义更深入的限制和命名约定。

10.2.12 元素<ESCAPE_SPECIAL_CHARS>

这个元素没有 XML 属性，它的内容是由字符串给出的布尔值，它可能有“true”，“false”，“1”，“0”值中的一个，“1”值等于“true”，“0”值等同于“false”。

这个元素控制着特定字符串在 ATF/XML 文件中的表示。应用元素名称或者应用属性/关系名称可能包含一个或多个字符_<>\ / ; + &以及空格字符。XML 不允许标签名称内所有字符，然而另一方面应用元素名称，属性或者关系将会作为标签出现在<instance_data>元素内容中，以上列出的字符

必须是转义的。为了指出这个是出现在 ATF/XML 文件中，元素<escape_special_chars>必须设定为“true”或“1”。

如果这个元素包含“true”或“1”，那么如何转换特殊字符的规则（10.2.40 节中给出）必须适用于 ATF/XML 文件应用模型部分中元素<name>，<ref_to>，和<inverse_name>的所有内容，也必须适用于 ATF/XML 文件<instance_data>元素内的相应标签。

否则，这些元素的内容必须被看作是它本身，即使它包含看起来像转义序列的字符串。

如果这个元素省略了，则假定为“false”值。

10.2.13 元素<BASE_MODEL_VERSION>

ATF/XML 文件必须声明 ODS 应用模型所采用的基础模型版本。这个元素没有 XML 属性，它的内容是一个字符串，该字符串由紧随两位数之后的'asam'组成。这个数字指定了使用的基础模型版本，就是现在的版本 30。

示例：基础模型的版本。

这段代码“asam30”表示 ASAM ODS 基础模型的版本。

```
<base_model_version>asam30</base_model_version>
```

10.2.14 元素<FILES>和<COMPONENT>

ATF/XML 文件可能声明包括除了 ATF/XML 文件本身以外的更多数据文件，它包含各种各样的二进制格式的测量数据。这些文件将被来自元素<instance_data>局部列值部分引用，（见 6.2.39 节和 6.2.42 节）。只有元素<files>中声明的文件可以被引用。

元素<files>没有 XML 属性，它包含类型'filecomponent'的一个或多个<component>元素序列。

<files>元素内的元素<component>包含由一个<identifier>和一个<filename>元素组成的序列，这些将在下面进行深入的描述。

注：有一个名为<component>的元素，它可能在<instance_data>元素中出现，用以指定局部列的值。将在本节后面进行。

示例：外部文件声明。

这段 XML 文件的代码说明一个名为“k1.dat”的文件是个外部文件，该文件位于子目录“data”中，其中子目录“data”与目前 XML 文件所在的目录是平行的，并且它可以被一级以上级别存取。在 XML 文件的<instance_data>元素中，这个文件可以被称为“file1”。

```
<?xml version="1.0" encoding="UTF-8"?>

<atfx_file ...>
...
<!--
*****
* declare any external files
*****
-->

<files>
  <component>
    <identifier>file1</identifier>
    <filename>../data/k1.dat</filename>
  </component>
</files>
```

10.2.15 元素<IDENTIFIER>

这个元素没有 XML 属性，它的内容是一串字符串。它指定了一个名称，这个名称稍后用来引用来

自<instance_data>元素内的外部文件。没有定义更深入的限制和命名约定。

10.2.16 元素<FILENAME>

这个元素没有 XML 属性，它的内容是一串字符串。它指定外部文件的文件名，这些外部文件传输局部列的二进制数据。

ATF/XML 文件中作为参考使用的路径和文件名必须可传输，并且能用于多种计算机架构。因此，路径和文件名定义的通用格式是最重要的，对于目标计算机而言，它必须非常容易解释和转换。

全球互联网使用的 URL 已经可以在不同的工作平台上运行。通过 URL 的命名约定，将来的 ATF 版本可能会将 ATF 文件分布在网络内。

URL 描述目标服务器使用的协议，目标系统所有计算机或者服务器名称，端口，目录路径和文件名。URL 有以下结构：

access_method: //server_name[: port]/directory_path/文件名称

示例：URL。

该例表示远程计算机（万维网服务器的主列表）上一个资源的 URL，利用 HTTP（Web）协议可以定位名为“www.w3.org”的服务器，通过标准端口可访问“/hypertext/DataSources/WWW”目录，该目录下包含一个超文本文档“Geographical.html”。

`http: //www.w3.org/hypertext/DataSources/WWW/Geographical.html`

互联网中每个文件都是以 URL 方式进行唯一寻址。

如果协议引用目录的内容，那么 URL 以斜线“/”结束。当引用协议中的文件时，不会有斜线出现。

上面描述的 URLs 被称为绝对的 URLs，因为它们包含了文件的全部路径名。URLs 的简单格式被称为相对的 URL。这种简化的格式将相同服务器上的其它文件引用为实际文件。采用相对的 URLs，ATF 文件可以直接引用物理元件，而不用引用服务器。

绝对的 URLs 可以引用互联网内的任何资源，包括局部资源。由于后面解释的原因，相对的 URLs 更适合于局部资源。

相对的 URL 暗示将根据 ATF 文件所含 URL 采用相同的存取方法，相同的服务器名称以及相同的目录途径。它描述了和实际 ATF 文件相关的 ATF 文件组件的相对位置。

示例：../matrix_xyz.dat。

相对的 URLs 使 ATF 目录结构移动到目录树中任何位置成为可能。一个相对的 URL 不需要引用实际文档目录，但可能包含和 ATF 文件根目录相关的引用。相对的 URLs 的变体类型以斜线（/）开头。它看起来与绝对的 URLs 相似，但不包含存取方法和服务器名称。

双向正确转换路径名称和插入正确的分隔符是各自目标系统上 ATF 文件阅读和写入程序的责任。

对于目标系统，如果文件名太长（例如 DOS-文件名称 8.3 的长度），那么文件名将会根据以下规则而缩短：文件名的第一个标点之后，最多允许有 3 个字符。这些字符都不能是标点。如果多于 3 个字符或者有另外标点，那么剩余部分就会缩短。不允许出现的字符以波浪字符取代（代字号）。

标点前面的字符数目被计算和删减到 8 个字符。如果文件名中的这个结果不唯一，那么来自 Windows 95 的算法可以被使用（删减到 6 个字符并使用后面 2 个字符进行编号）。

为了使文件传输到 UNIX 系统更简单些，文件名通常使用小写，甚至在不区分大小写的操作系统上（VAX/VMS，MSWindows）也是这样。

只有当 UNIX 计算机的路径和文件名使用大写时，文件名根据需要才使用大写。

10.2.17 元素<APPLICATION_MODEL>

这个元素没有 XML 属性，它包含零个或多个应用枚举定义序列，随后是一个或多个应用元素。

它用来描述应用程序的特定数据模型，包含应用元素以及它们和其它应用元素之间的联系。

此外，它包含枚举，这些枚举需要指定枚举类型属性的枚举项的集合。强烈建议提供由相同 ATF/XML 文件应用模型部分内的实例所使用的所有枚举。然而，那些由 ASAM ODS 指定的枚举（例如 datatype_enum）不能由 ATF/XML 文件提供，因为这可能会导致与存储在服务器中的枚举定义不一致。通常使用的或一般指定的枚举可以通过 XML 的 XInclude 机制包括在 ATF/XML 文件中。

示例：应用模型的声明。

```
...
<!--
*****
* declare application model meta data
*****
-->
<application_model>
  <application_element>
    <!--    *** declare Engine Element    *** -->
    ...
    <name>Engine</name>
    ...
  </application_element>
</application_model>
...
```

注：应用模型作为一个整体必须符合 ASAM ODS 基础模型。这包括应用元素的数目，它们的属性及关系，数据类型等。

10.2.18 元素<APPLICATION_ENUMERATION>

这个元素没有 XML 属性，它包含<name>的序列，随后是一个或多个<item>。

这个元素指定一个枚举。一个枚举定义作为属性值的有限字符串集。在 ATF/CLA 中，这些值在属性定义内被内嵌声明。在 ATF/XML 中，枚举被分别声明，以便它们能被多个属性重新利用。

枚举的名称必须在元素<name>中指定，并且被使用者定义。

每个<item>都指定一个允许的字符串以及鉴别字符的唯一编号。当指定数据类型和 DT_ENUM 相一致的属性值时，这些字符串在一个枚举内也必须唯一，因为它们最终会被用在实例中。建议这些编号以（0）开头，并连续增加。

应当注意，当写入服务器时，ASAM ODS OO-AP 不允许设定枚举项的编号，而是内部自动增加项

目编号；因此导入客户端应当注意正确的调用序列，或者服务器和 ATF/XML 文件可能为相同项字符串提供不同的项目编号。

示例1：示例 1：枚举。

```
...
<application_enumeration>
  <name>MyEnum</name>
  <item>
    <name>first</name>
    <value>0</value>
  </item>
  <item>
    <name>second</name>
    <value>1</value>
  </item>
</application_enumeration>
...
```

枚举是被应用属性使用的，为了声明应用属性值仅限于那些被定义的枚举，枚举的链接必须声明。

示例2：链接一个枚举。

```
...
<application_attribute>
  <name>air_filter</name>
  <datatype>DT_ENUM</datatype>
  <enumeration_type>MyEnum</enumeration_type>
</application_attribute>
...
```

10.2.19 元素<ITEM>

这个元素没有 XML 属性，它包含<name>的序列，随后是一个或多个<item>。

这个元素准确指定一个枚举项，是一对字符串（项目名称）和编号（项目值）。

项目的名称必须在元素<name>中指定，并且必须由使用者定义；然而，由于一些字符在 XML 内容中不允许或者可能会导致在实例部分出现禁用的 XML 标签，因此，一些转义规则已经建立（见本节后面）。

项目值必须在元素<value>中指定。

10.2.20 元素<VALUE>

这个元素没有 XML 属性，它的内容是非负整数（从零开始）。它用来准确鉴别一组项目以外的枚举项。

10.2.21 元素<APPLICATION_ELEMENT>

元素没有 XML 属性，它包含<name>，<basetype>以及可选的<inherits_from>序列，随后是零或更多 <application_attribute> 和 <relation_attribute> 以及 零 或 更多 <application_attribute>，<relation_attribute>元素可能会混淆。

它说明一个应用元素。

应用元素的名称必须在元素<name>中被指定，并且被使用者定义；然而，由于一些字符在 XML 内容中不允许出现或者可能会导致在实例部分出现禁用的 XML 标签，因此，一些转义规则已经建立（见本节后面）。

每个应用元素必须派生自 ASAM ODS 基础元素。这个基础元素在<basetype>元素中被引用。

ASAM ODS 基础模型指定每个基础元素的基本属性和关系。那些以 MANDATORY ATTRIBUTE 为特征的元素必须有和<relation_attribute>副本相关的<application_attribute>，其它可能有一个。可以添加更深一步的属性或关系（这就是所谓的“扩展”）。

注：每个应用元素必须符合 ASAM ODS 基础模型。这包括基本的应用属性和关系的数目，数据类型等等。

10.2.22 元素<NAME>

这个元素没有 XML 属性，它的内容是字符串。它指定应用元素的名称。

应用元素的名称可以被使用者定义，然而，由于一些字符在 XML 内容中不允许出现或者可能会导致在实例部分出现禁用的 XML 标签，因此，一些转义规则已经建立（见本节后面）。

应用元素的名称在整个 ASAM ODS 环境中必须唯一。

10.2.23 元素<BASETYPE>

这个元素没有 XML 属性，它的内容是字符串。它指定 ASAM ODS 基础模型给出的相应基础元素的名称。

ASAM ODS 基础模型定义了允许的基础元素名称集。在应用元素中无论什么被指定为<basetype>，那么它将决定<base_attribute>和<base_relation>内容允许的名称集。

尽管基础模型在服务器 API 上是不区分大小写的，但是就像第四章基础模型定义给出的那样，它严格要求提供 AFT/XML 文件内的基础元素名称。

10.2.24 元素<INHERITS_FROM>

这个元素没有 XML 属性，它的内容是字符串。如果子集应用元素随后被指定，那么它仅出现在<application_element>内。

<inherits_from>提供被子类元素继承的超类应用元素的名称。

请注意，在应用元素名称中有指定字符的转义规则（见以下部分）。

10.2.25 元素<APPLICATION_ATTRIBUTE>

这个元素没有 XML 属性，包含<name>,<base_attribute>,<datatype>，<enumeration_type>,<autogenerate>,<obligatory>,<unique>和<length>的序列。它们中的每一个至多出现一次，某些是可选的。

这个元素指定一个应用属性。

应用属性的名称必须在元素<name>中被指定，并且可以被使用者指定；然而，由于一些字符在 XML 内容中不允许出现或者可能会导致在实例部分出现禁用的 XML 标签，因此，一些转义规则已经建立（见本节后面）。

应用属性的结构见图 102。

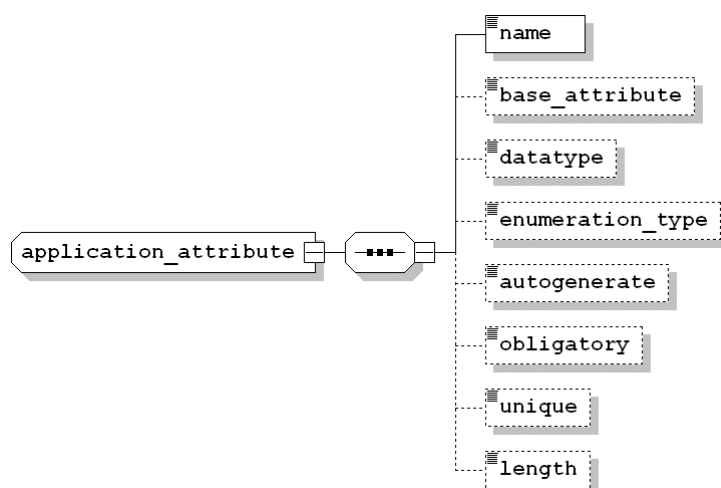


图102 应用属性的结构

根据 ASAM ODS，应用属性可能是基础的或是扩展的应用属性。

1) 如果是基本的应用属性，那么<name>和<base_attribute>元素必须出现。在那种情况下，<base_attribute>元素指定基本属性的名称，像在 ASAM ODS 基础模型中给出的那样。此外，在这种情况下，<datatype>将不会出现（除了在 5.4.3 节中给出的情况，例如使用时间过长而难以维护的应用模型），<obligatory>必须像在 ASAM ODS 基础模型中给出的那样被设置，或者可被缩小（例如，如果基础模型允许属性是可选择的，应用模型可能会需要它）。值得注意的是，对于基本的应用属性，数据类型已经在 ASAM ODS 基础模型中被指定，根据这些数据类型实例必须指定它们的值。例如，对于与 DT_ENUM 相应类型的基本属性，实例会申明枚举项的字符串表示，而不是数量（尽管物理存储可能仅储存数字）。

作为例外，基础元素 AoLocalColumn 有其它的关系“值”，这些值引用一系列值。应用模型将会从基本关系中得到应用关系。然而，应用模型的 ATF/XML 规范将会处理这个应用关系，就好像它是一个应用属性；因此在这种情况下，不是<relation_attribute>而是<application_attribute>被指定，并且<base_attribute>的内容是那种基本关系的名称（“值”）。

2) 对于扩展的应用属性，元素<base_attribute>必须省略，而元素<name>和<datatype>必须出现；<datatype>必须指定有效的 ASAM ODS 数据类型枚举值。

如果数据类型符合 DT_ENUM（基础的和扩展的应用属性），那么元素<enumeration_type>仅仅出现（必须出现）在<application_attribute>的内容中。那时不管它是否是 ASAM ODS 或应用模型指定的枚举，枚举也必须以<application_enumeration>元素包含在 ATF/XML 文件中。

元素<autogenerate>，<obligatory>和<unique>指定一些应用属性的特点。如果它们被省略，那么

“false”就被假定；对于基本应用属性，<obligatory>的内容由 ASAM ODS 基础模型给出。某些基本属性的<unique>内容应用相同。

元素<length>最后被提供，来指定在实例中应用属性可能有的字符串的最大数。

如果数据类型是一串字符串类型（DT_STRING、DT_DATE、DS_STRING、DS_DATE,和（三部分的每一部分） DT_EXTERNALREFERENCE 和 DS_EXTERNALREFERENCE, 和（开头部分） DT_BLOB），所以为这种应用模型创建物理存储的服务器知道这些属性的最大尺寸；如果它被省略，那么 TF/XML 文件内的无限长度必须被假定，实际最大长度只能在读完文件后决定。在所有其它案例中它必须省略。

对于应用属性，ATF/CLA 和 ATF/XML 之间有重大的差异。ATF/XML 文件能够处理 ODS 基础模型的最新扩展--也就是，字符串长度，强制性标记，自动生成标记和唯一标记。扩展还没有被添加到 ATF/CLA 规范中。另一个不同之处是数据类型的声明。在 ATF/CLA 中，派生自基本属性的属性可以声明一个不同于它们派生的基本属性的数据类型。ATF/XML 不允许来自基本属性的任何偏离。

10.2.26 元素<BASE_ATTRIBUTE>

元素没有 XML 属性，它的内容是字符串。它仅出现在基本的应用属性中，像 ASAM ODS 基础模型中给出的那样，它指定相应基本属性的名称。

ASAM ODS 基础模型定义已允许的属性名称集合。它们取决于派生实际应用元素的基础元素。在应用元素中，无论什么被指定为<basetype>，它都决定<base_attribute>内容的允许名称集。

在相同应用元素内，一个基本属性可能至多仅被在一个应用属性使用。

在服务器 API 中，尽管基础模型是不区分大小写的，但是在 ATF/XML 文件中它严格要求以小写字母形式（第 7 章基础模型定义中完成）提供基本属性的名称。

10.2.27 元素<DATATYPE>

这个元素没有 XML 属性，它的内容是字符串。它仅出现在扩展的应用属性中（在 ASAM ODS 基础模型中基本应用属性的数据类型已经指定）。

元素<datatype>指定应用属性实例值的数据类型的枚举项名称。这个元素给出的信息被用作：

- 表示实例属性值的字符序列的解释；
- 当导入 ATF/XML 文件时，对于应用属性，ODS 服务器定义数据类型。

数据类型枚举项名称的列表在 5.5 节中被指定。然而，对于应用属性，以下的名称不得作为应用数学的数据类型说明：DT_UNKNOWN, DT_ID, DS_ID。

10.2.28 元素<ENUMERATION_TYPE>

这个元素没有 XML 属性，它的内容是字符串。它仅出现在应用属性中，应用属性的<datatype>元素包含 DT_ENUM 或 DS_ENUM。在那些案例中，<enumeration_type>包含枚举的名称，其中枚举的项目（它们的字符串表示形式）可能变成应用属性的实例“值”。

如果引用的枚举不包含在相同的 ATF/XML 文件中，那么关于枚举的文件是自我包含的。

如果引用的枚举不包含在相同的 ATF/XML 文件中，那么基于文件自身的引用的完整性不能被检查到，客户端有责任核实服务器是否包含已给出名称的枚举。

10.2.29 元素<AUTOGENERATE>

这个元素没有 XML 属性，它的内容是“true”或“1”（两个都意味着应用属性的 AUTOGENERATE 标签将会被设置）或是“false”或“1”（两个都意味着应用属性的 AUTOGENERATE 的标签将会被复位）。

当设置应用属性的 AUTOGENERATE 标签时，应当小心所有的客户端/服务器，这些所有的客户端/服务器处理相同的应用模型，按照同样的程序在实例中产生值。通常这样的程序在服务器和客户端中手动设置。

10.2.30 元素<OBLIGATORY>

这个元素没有 XML 属性，它的内容是“true”或“1”（两个都意味着应用属性的 OBLIGATORY 标签将会被设置）或是“false”或“1”（两个都意味着应用属性的 OBLIGATORY 标签将会被复位）。

如果为应用属性设置 OBLIGATORY 标签，那么相应应用元素的每一个实例必须有一个属性值，如果它复位，实例可以但不需要有这些应用属性值。

对于应用属性，基础模型决定了这个元素的内容。

如果相应的基本属性是可选的，那么 OBLIGATORY 标签必须复位（`<obligatory>>false</obligatory>`），另外 OBLIGATORY 标签必须被设定为（`<obligatory>>true</obligatory>`）。

10.2.31 元素<UNIQUE>

这个元素没有 XML 属性，它的内容是非负整数。

这个元素没有 XML 属性，它的内容是“true”或“1”（两个都意味着应用属性的 UNIQUE 标签将会被设置）或是“false”或“1”（两个都意味着应用属性的 UNIQUE 标签将会被复位）。

如果为应用属性设置 UNIQUE 标签，那么相应应用元素的每一个实例必须有那个属性的值，这个属性不同于那个应用元素的所有其它实例，也不同于继承的应用元素的所有实例的值。如果标签是复位的，那么对于这些应用属性不同的实例可能会有相同的值。

10.2.32 元素<LENGTH>

这个元素没有 XML 属性，它的内容是非负整数。它指定这些应用属性值的最大长度。

10.2.33 元素<RELATION_ATTRIBUTE>

这个元素没有 XML 属性，它包含了<name>，<ref_to>，和可选择性的<base_relation>，<min_occurs>，<max_occurs>，和<inverse_name>的序列。它们中的每一个都至多出现一次，某些是可选的。

这个元素指定了应用关系，是应用元素之间建立的关系。

父子元素之间的关系是通用的，但是 ASAM ODS 确定了更多的关系类型。关系属性结构见图 103。

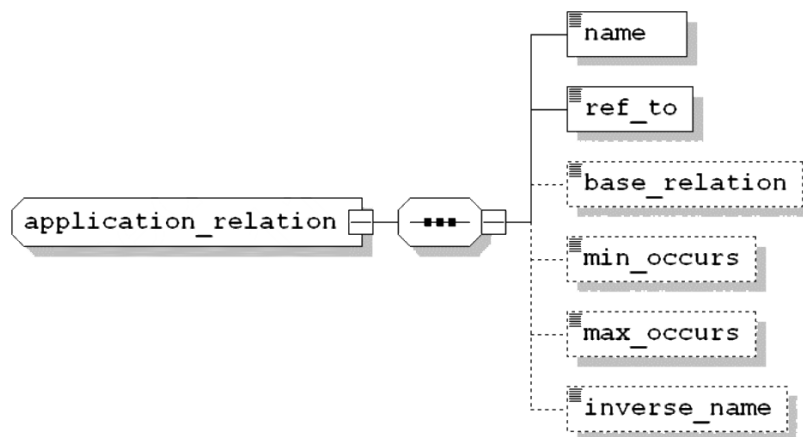


图103 关系属性的结构

应用关系的名称在元素<name>中必须确定，也可以被使用者确定，然而某些字符并不允许在 XML 内容中出现，也可能在实例部分出现禁用的 XML 标签，为此，一些转义规则已经建立（见下面内容）。

<ref_to>指定应用元素的名称，应用元素是这个关系的目标。元素<name>和<ref_to>总是在 <relation_attribute>.内出现。

根据 ASAM ODS，应用关系可能是一个基本的或是扩展的应用关系。

1) 在基本的应用关系中，<base_relation>元素必须出现，元素<min_occurs>和<max_occurs>可以被忽略，因为它们已经由 ASAM ODS 基础模型指定，然而，应用模型可能缩小基数（见 5.4.6 节）。在这情况下，<base_relation>元素像 ASAM ODS 基础模型给出的那样指定基本关系的名称。

作为例外，基础元素 AoLocalColumn 有其它的关系值，这些值引用一系列值。应用模型将会从基本关系中获得应用关系。然而，应用模型的 ATF/XML 规范将会处理这个应用关系作为应用属性。因此，不是<relation_attribute>而是<application_attribute>将会在案例中被指定。

2) 在扩展的应用关系中，元素<base_relation>必须被忽略。<min_occurs>和<max_occurs>指定可以被关系引用的实例的最大和最小数目。

元素<inverse_name>指定相反关系的名称，相反关系是被应用元素定义的，这个元素被<ref_to>引用，也引用这个应用元素。

并不像 ATF/CLA，ATF/XML 将应用元素和应用关系分开，为声明关系提供标准的模式。

示例1：扩展应用关系的声明。

```
...
<relation_attribute>
  <name>TestUser</name>
  <ref_to>User</ref_to>
  <min_occurs>0</min_occurs>
  <max_occurs>1</max_occurs>
  <inverse_name>UserTests</inverse_name>
</relation_attribute>
...
```

示例2：应用元素的声明。

```

...
<!-- *** declare Test Element *** -->
<application_element>
  <name>Test</name>
  <basetype>AoSubTest</basetype>
  <application_attribute>
    <name>TestId</name>
    <base_attribute>id</base_attribute>
    <autogenerate>true</autogenerate>
    <obligatory>true</obligatory>
    <unique>true</unique>
    <length>50</length>
  </application_attribute>
  <application_attribute>
    <name>TestName</name>
    <base_attribute>name</base_attribute>
    <obligatory>true</obligatory>
  </application_attribute>
  <application_attribute>
    <name>TestDescription</name>
    <base_attribute>description</base_attribute>
    <length>100</length>
  </application_attribute>
  <application_attribute>
    <name>TestVersion</name>
    <base_attribute>version</base_attribute>
    <obligatory>true</obligatory>
  </application_attribute>
  <application_attribute>
    <name>VersionDate</name>
    <base_attribute>version_date</base_attribute>
    <obligatory>false</obligatory>
  </application_attribute>
  <application_attribute>
    <name>CreateDate</name>

```

```

        <datatype>DT_DATE</datatype>
        <obligatory>>false</obligatory>
    </application_attribute>
    <application_attribute>
        <name>TestType</name>
        <datatype>DT_STRING</datatype>
        <obligatory>>false</obligatory>
    </application_attribute>
    <application_attribute>
        <name>TestComment</name>
        <datatype>DT_STRING</datatype>
        <obligatory>>false</obligatory>
    </application_attribute>
    <application_attribute>
        <name>exhaust</name>
        <datatype>DT_STRING</datatype>
        <obligatory>>false</obligatory>
    </application_attribute>
    <application_attribute>
        <name>air_filter</name>
        <datatype>DT_STRING</datatype>
        <obligatory>>false</obligatory>
    </application_attribute>
    <!--*** related elements *** -->
    <relation_attribute>
        <name>MainTest</name>
        <ref_to>Engine</ref_to>
        <base_relation>parent_test</base_relation>
        <min_occurs>0</min_occurs>
        <max_occurs>1</max_occurs>
        <inverse_name>Engine</inverse_name>
    </relation_attribute>
    <relation_attribute>
        <name>Measurements</name>
        <ref_to>Measurement</ref_to>
        <base_relation>children</base_relation>
        <min_occurs>0</min_occurs>

```



```

        <max_occurs>Many</max_occurs>
        <inverse_name>Test</inverse_name>
    </relation_attribute>
    <relation_attribute>
        <name>TestUser</name>
        <ref_to>User</ref_to>
        <min_occurs>0</min_occurs>
        <max_occurs>1</max_occurs>
        <inverse_name>Tests</inverse_name>
    </relation_attribute>
</application_element>
...

```

10.2.34 元素<REF_TO>

此元素没有 XML 属性，并且它的内容是一个字符串。它指定应用关系名称，也就是关系的目标。

ATF/XML 文件必须包含一个完整的应用模式，以便把所有用到的元素纳入同一个文件中。因此在这个相同的 ATF/XML 文件中必须存在一个应用元素，它的名称与<ref_to>内容所给名称一致。

10.2.35 元素<BASE_RELATION>

此元素没有 XML 属性，并且它的内容是一个字符串。它只在基本的应用程序关系中出现，并指定相应 ASAM/ODS 基础模型中基本关系的名称。

ASAM/ODS 基础模型定义所允许的基本关系的名称。它们依赖于基础元素，而实际应用元素是由基础元素派生的。在应用元素中任何一个被指定为<basetype>的元素决定<base_relation>的内容许的名称集合。

一个基本关系最多只有在相同应用元素内的一个应用关系中使用。

尽管基础模型在 API 服务器中不区分大小写，但是在 ATF/XML 文件中它被严格的要求提供用小写字母表示的基本关系的名称（正如在第 4 章中基本模式的定义中所说）。

10.2.36 元素<MIN_OCCURS>

这个元素没有 XML 属性，并且它的内容是一个非负整数。它指定这种关系的最小值，相关实例应用元素实例可能有这种关系。

尽管在原则上<min_occurs>可以包含任何 32 位无符号整数，但它的主要用途通常是确定（1）或非（0）的这种关系是否必须存在于应用元素的任何实例中。

10.2.37 元素<MAX_OCCURS>

此元素没有 XML 属性，并且它的内容是一个非负整数或字符串“many”。它指定这种关系的最大值，相关实例应用元素实例可能有这种关系。

在原则上，元素<max_occurs>可能包含任何 32 位无符号整数，它给出为应用元素的每个实例而

建立关系的确定的最大数。如果这个关系的数目完全没有限制，而是设置一个很大的数字，那么字符串“many”会被使用。

10.2.38 元素<INVERSE_NAME>

此元素没有 XML 属性，并且它的内容是一串字符串。它指定被<ref_to>引用的应用元素中定义的应用关系的名称。该应用关系和包含<inverse_name>的实际关系相反。

ATF/XML 文件必须包含一个完整的应用模型，以便把所有引用的应用元素包含在相同文件中。因此相同的 ATF/XML 被<ref_to>引用的应用元素必须存在，并且必须有一个关系，这个关系的名称与<ref_to>内容所给的名称相同。

10.2.39 元素<INSTANCE_DATA>

此元素有没有 XML 的属性和它的内容是形式合法的实例序列。至少有一个实例必须包含在这个元素中，否则<instance_data>必须省略。

每个实例都被封装在一个标签中，其名称是实例应用元素的名称。如果多个实例被相同应用元素创建，那么在<instance_data>中相同标签出现不止一次。虽然 ASAM/ODS 没有同时严格要求，但是在将其它应用元素添加到文件之前把相同应用元素的所有实例一个接着做一个地放在一起是一种很好的做法；这大大增加了解析的性能。

一个实例没有 XML 属性，并且包含属性和关系值的序列，由零个或多个特定实例属性紧随其后，称为“实例属性”。

每个属性值本身被封装在一个标签中，这个标签没有 XML 属性且它的名称是相应应用属性的名称，这点同样适用于关系值：它们的标签没有 XML 属性，它们的标签名称是相应应用关系的名称。在基本的或扩展的应用属性/关系之间不存在结构性分离，在应用属性和应用关系之间也没有结构性分离。虽然 ASAM/ODS 现在没有严格要求，但是在相同序列中提供应用属性和应用关系（在应用元素中的应用程序模型部分已经将其定义）是最佳做法，其中相同序列已经在那个应用元素的应用模型部分定义。

一个实例包含至少两个属性值：一个是它的标识符（这个实例的应用属性于基本属性“ID”一致），这个标识符只是一个数字，并且在相同应用元素的所有实例中是唯一的；另一个是它的名称（这个实例的应用属性于基本属性“name”一致）。

应用属性的值存储在一个组件文件中或者必须存储在一个格式的标签中，这个格式适合于为那个应用元素给出的数据类型。

在第一种情况中（这只被一个局部列中派生自基本属性“flags”的属性以及一个局部列中派生自基本关系‘values’的关系所允许），属性标签包含该元素<component>，这在第 6.2.42 节已经描述过，它们的值可以在组件文件中找到。

在第二种情况中，属性的值（这也可能是个别情况下 DS_..类型的属性值序列）必须放在标签内。虽然对大多数数值类型来说这是显而易见的，但应该注意的是：

- 布尔值可能是逻辑真值“true”或“1”和逻辑假值“false”或“0”。这与布尔值的 XML 表示相一致；
- 字节流是一个字节序列。每个字节由 0 到 255 范围内的十进制数表示，字节被空格字符分隔；
- 复数值是由浮点值对给定的：首先是实部，然后是虚部，二者由一个空格隔开；

- 枚举值由字符串值给出，而不是由其整数项编号给出；
- 外部引用在子元素<external_reference>中给出，这个子元素没有 XML 属性，并且包含元素<description>，<mimetype>，和<location>序列。这三个元素都没有 XML 属性，并且都包含描述性字符串，相关的模拟类型字符串和相关的位置字符串；
- 一个 blob 值是由<text>和<bytefield>两个元素给出的，后者含有两个子元素<length>和<sequence>。<text>保存 blob 的标题文字，长度包含<sequence>中所含字节数目，<sequence>包含自身字节序列（序列之间有一个空格符隔开）。字节是 0-255 间的十进制数字。

如果一个属性值对具体实例是未知的或者没有定义，那么它的标签必须省略；此行为允许使用例如字符串类型属性来区分一个空字符串（如<attName></attName>或<attname/>）和未定义的属性值（在这种情况下，标签不会出现）。

单值属性必须立即把值放在属性标签中。对多值属性来说，例如列表，数据类型决定值是如何被声明的。这是 XML 处理列表和数据集的方式决定的。XML 声明，值的列表的各个值是用空格分开的。这非常适用于除字符串以外的所有数据类型：例如一个被空格分开的字符列表中不能区分出一个含有空格的字符串。因此，包含多个字符串值的属性被声明时应该使用一个子元素序列，每个属性包含一个字符串。在 ODS 帮助模式（6.5.3 节）中，说明了基本应用元素使用的子元素的标签名称；所有扩展的应用属性是<S>。这适用于与 DS_STRING 或 DS_ENUM 相关的数据类型属性。与 DS_EXTERNALREFERENCE 相关的数据类型属性包含不止一个<external_reference>元素。

示例：有多个字符串值的属性。

这是一个有多个字符串值的属性的例子，第一个块给出了应用模型相应部分，第二个块指定一个值。

```

...
<application_model>
  <application_element>
    ...
    <application_attribute>
      <name>Mfg_Plants</name>
      <datatype>DS_STRING</datatype>
      <obligatory>false</obligatory>
    </application_attribute>
    ...
  </application_element>
  ...
</application_model>
...
...
<Mfg_Plants>
  <s>Ypsilanti</s>
  <s>Stockholm</s>

```

```

    <s>Turin</s>

    <s>San Francisco</s>

  </Mfg_Plants>

  ...

```

一个应用关系的值包含一个或多个整数数字或者 ASAM 路径，它们彼此间由空格隔开。每个数字引用另一个应用元素的实例，这种引用是通过指定那个引用的实例的唯一标识符值来实现的；这个实例必须存在于 ATF/ XML 文件内。每个 ASAM 路径引用另一个应用元素的实例，这种引用是通过指定那个实例的完整 ASAM 路径得以实现的；这个实例必须存在于任何一个 ATF/XML 文件导入的 ASAM/ODS 服务器内。

这种做法有一个例外：基础元素 **AoLocalColumn** 已位于“values”的关系中，这些值引用值序列。应用模型将从这种基本关系中衍生出一个应用关系。然而应用模型 ATF/XML 规范会把这种关系当做它的一个应用关系属性对待。

因此，在实例中这种关系的值将不是整数的一个列表或 ASAM 路径（就像被关系要求的一样）而是：

- 要么是一个包含本身测量值序列的数据类型标签；
- 要么是描述一个链接到二进制文件（ATF/XML 组件文件）的<component>元素，这个二进制文件包含值以及这些值在那个文件中的位置。

10.2.42 节中将对第二个实例做进一步描述。

在第一个例子中，对较小或中等大小的数据来说是一个简单的方法，测量值完全包含在 ATF/XML 文件中。在这个 ATF/XML 文件中数据类型表示可能不同于 ASAM/ODS 服务器所要求的（是由派生自相应测量量的基本应用属性“datatype”的应用属性值指定的）表示。因此测量数据值包含在说明 ATF/XML 文件（见已知问题部分）当中数据类型的数据类型标签中。

下面的数据类型标签可能会用到，它们没有 XML 属性。

```

<A_UTF8STRING>, <A_INT16>, <A_FLOAT32>, <A_BOOLEAN>, <A_INT8>, <A_INT32>,
<A_FLOAT64>, <A_INT64>, <A_TIMESTRING>, <A_BYTEFIELD>, <A_BLOB>,
<A_COMPLEX32>, <A_COMPLEX64>, <A_EXTERNALREFERENCE>

```

示例：ATF/XML 文件中的测量数据值。

这是一个有关测量数据的实例，这些测量数据以浮点数的方式被直接放在 ATF/XML 文件中。假定“aeValues”是派生自基本关系“values”中应用属性的名称。

相应的应用模型规范没有在此表示出来。

```

...

<aeValues>

  <A_FLOAT32>

    .56 20 21.4 43.2 94.6 1.3E2 2.56E2 7.6E2 4.102E3

  </A_FLOAT32>

</aeValues>

...

```

注意，圆点'.'是十进制分隔符。

各个值用空格分开。字节流是字节序列，每个字节由 0-255 的十进制数字表示；字节间被空格字符隔开。一些数据类型需要特别处理：

字符串类型值放在元素<A_UTF8STRING>中。原则上允许将字符串指定为任何的 UTF-8 字符序列。然而需要注意的是，ASAM ODS V5.2.0 规范中其它部分不支持完全的 UTF 字节设置。因此，该 ATF / XML 规范是对一个 UTF 兼容的数据存储的第一步。除非其它部分适合 UTF 字节设置，否则应谨慎使用 ISO 8859-1 字符集以外的字符。

如果测量时间被表示成字符串类型（<A_UTF8STRING>）并且给出不止一个，那么每一个值会被相似地看成是多字符串值的属性（见上）；它们被指定为子元素<s>序列，每个序列包含一个单字符串元素。这样允许字符串值内出现空格（否则无法与后面的字符串值之间的分隔符区分）。这也意味着，枚举值由它们的字符串值，而不是整数项编号给出。

A_BOOLEAN 的内容可能由有效的布尔 XML 值组成，即由 XML 规范决定的'1', '0', 'true', 'false'。

A_BYTEFIELD 含有两个 XML 元素：<length>和<sequence>，这两个元素都没有 XML 属性。<length>含有用整数表示的<sequence>中的字节数。<sequence>包含字节流中的单个字节，每个字节用 0-255 间的整数表示。

示例：ATF/XML 中的字节流

这个例子显示如何在 ATF / XML 中表示字节流'0000 000312 34 DE'。前四个字节为字节流的长度，并且放在为<length>中。最后三个字节是字节流值，并且被放到<sequence>中。假设，“aeValues”是派生自基本关系“values”的应用属性的名称。

```
...
<aeValues>
  <A_BYTEFIELD>
    <length>3</length>
    <sequence>18 52 222</sequence>
  </A_BYTEFIELD>
</aeValues>
...
```

A_BLOB 包含两个 XML 元素<text>和<bytefield>，且都没有 XML 属性。<text>包含 blob 的文本；<bytefield>包含 blob 的大小以及它的二进制部分；它的组成像<A_BYTEFIELD>一样含有一个<length>和一个<sequence>子元素。

示例：ATF/XML 文件中的 BLOB

这个例子显示了含有文本'MyBlob'的 blob 以及四个 blob 字节（是由十六进制字符表示的分别为 12 2A 34 DE）在 ATF/XML 是怎样表示的，其中'MyBlob'有四个字节。假定“aeValues”是派生自基本关系“values”的应用属性的名称。

```
...
<aeValues>
  <A_BLOB>
```

```

    <text>MyBlob</text>

    <bytefield>

        <length>4</length>

        <sequence>18 42 52 222</sequence>

    </bytefield>

</A_BLOB>

</aeValues>

...

```

如果测量数据值被表示成复杂类型（<A_COMPLEX32>或者<A_COMPLEX64>），那么偶数值必须在标签中给出，每一个复数被两个连续的浮点数字指定；实部先给出，随后是虚部。

如果测量数值被表示为外部引用，那么数据类型标签<A_EXTERNALREFERENCE>的内容是一个或多个没有 XML 属性的元素<external_reference>，但是每个元素都含有<description>，<mimetype>，和<location>组成的序列。这三个元素都没有 XML 属性，但含有描述性字符串，相关的 mimeType 型字符串和位置字符串。

最终我们给出一个完全确定的实例如下。

示例：实例数据

这是一个应用模型的一部分（在第一个块中），随后是一个相应的实例数据（在第二个块中）的简单例子。

```

...

<application_model>

...

<application_element>

    <name>Engine</name>

    <basetype>AoUnitUnderTest</basetype>

    <application_attribute>

        <name>EngineId</name>

        <base_attribute>id</base_attribute>

        <autogenerate>true</autogenerate>

        <obligatory>true</obligatory>

        <unique>true</unique>

    </application_attribute>

    <application_attribute>

        <name>EngineName</name>

        <base_attribute>name</base_attribute>

        <obligatory>true</obligatory>

        <unique>true</unique>

        <length>50</length>

    </application_attribute>

```

```

<application_attribute>
  <name>EngineDescription</name>
  <base_attribute>description</base_attribute>
  <length>100</length>
</application_attribute>
<application_attribute>
  <name>CreateDate</name>
  <datatype>DT_DATE</datatype>
  <obligatory>false</obligatory>
</application_attribute>
<application_attribute>
  <name>bore</name>
  <datatype>DT_FLOAT</datatype>
  <obligatory>false</obligatory>
</application_attribute>
<application_attribute>
  <name>cylindernumber</name>
  <datatype>DT_SHORT</datatype>
  <obligatory>false</obligatory>
</application_attribute>
<relation_attribute>
  <name>EngineUser</name>
  <ref_to>User</ref_to>
  <min_occurs>0</min_occurs>
  <max_occurs>1</max_occurs>
</relation_attribute>
</application_element>
</application_model>

<instance_data>
  <Engine>
    <EngineId>1</EngineId>
    <EngineName>Test Engine 1</EngineName>
    <EngineDescription>The first test engine</EngineDescription>
    <CreateDate>199702010900</CreateDate>
    <EngineType>ABC47111</EngineType>
    <bore>92</bore>
  </Engine>
</instance_data>

```

```
<cylindernumber>6</cylindernumber>

<EngineUser>21 22 23 24 25 87 62</EngineUser>

</Engine>

</instance_data>

...
```

10.2.40 名称中特殊字符的使用

在XML1.0规范中，XML元素和XML属性的名称可能包含字母、数字和组合而成的无空格字符组。嵌入空格是不允许的。结果，XML元素的名称如“measurement description”是不允许的，这种名称的应用属性（它最终会导致在<instance_data>中出现一个XML元素）通常是需要避免的。

非法语法：

```
<Engine>
  <Engine Id>1</Engine Id>
  <Engine Name>Test Engine 1</Engine Name>
</Engine>
```

然而ASAM/ODS在应用模型的命名约定中不得使用这些限定条件（事实上在应用模型中可能已经存在大量的命名约定，应用模型的元素/属性/关系名称含有空格或其它特殊字符）。因此ASAM/ODS已经在ATF/XML中定义了它专用的转义顺序：

下划线字符‘_’（在ASCII中用十六进制字符表示为0x5F）应作为转换字符。它后面紧跟着一个待转换字符的两位数的十六进制代码。这里甚至允许名称以这些转换字符开始（尽管不准许使用如引导空格，因为空格难以被使用者看到。）

只有有限的字符应该转换：_ < > \ / ; + & 和空格。它们以以下转换序列表示。

ATF/XML中的转义字符见表112。

表170 ATF/XML中的转义字符表

字符	转义序列
_	_5F
<	_3C
>	_3E
\	_5C
/	_2F
;	_3B
+	_2B
&	_26
空格	_20

只有在ATF/XML文件的应用模型部分的元素<name>,<ref_to>及<inverse_name>以及在ATF/XM

L 文件中的<instance_data>元素的相应标签中这些字符才可转换。在 XML 属性或其它元素的内容中它们会保持原样；如果 XML 本身需要在这种元素内容中转换它们，那么转换特定字符的 XML 方法会被使用。

如果 ATF/XML 包含有 ASAM ODS 特殊转换的应用元素/属性/关系，那么元素<escape_special_chars>必须含有"true" 或 "1"以便读取这个 ATF/XML 文件的任何应用程序知道需要采用这个转换程序。同时，如果被写 ATF/XML 中的数据含有特殊字符，必须按照上述原则转换，并且元素<escape_special_chars>必须设为"true" 或 "1"。

如果一个 ATF/XML 文件不包含需要这种 ASAM ODS 特殊转换的应用元素/属性/关系，那么元素<escape_special_chars>应该包含"false"或 "0"以提高解析性能。

如果元素<escape_special_chars>包含"false" 或 "0"，那么读取 ATF/XML 文件的应用程序不必考虑这些转换字符，即使这个文件包含看似像转义序列的字符串。

上述实例的合规范语法：

```
<Engine>
  <Engine_20Id>1</Engine_20Id>
  <Engine_20Name>Test Engine 1</Engine_20Name>
</Engine>
```

10.2.41 实例属性说明

实例属性是实例特有的信息，在应用实例模型中是无法预见。因此，从应用模型中不能得到它们的名称和数据类型，并且必须在特定实例中给出。有时了解与实例属性值有关的受测设备也是很有用的。

实例属性被声明为一个实例的一部分，这是通过把 XML 元素<instance_attributes>放到实例的内容中，其后紧跟着所有的属性值和关系值来完成的。如果没有声明实例属性，元素<instance_attributes>必须省略。

元素<instance_attributes>没有 XML 属性，并且它的内容是一个或多个代表个体实例属性的元素组成的序列。依赖于用于指定实例属性值的数据类型，元素可能是：

- <inst_attr_asciistring>：如果实例属性值是一个字符串，那么这个元素就会被用到；
- <inst_attr_float32>：如果实例属性值能用单精度浮点值代表，那么这个元素就会被使用（注意：字符点'.'被用作十进制分隔符）；
- <inst_attr_float64>：如果实例属性值能用双精度浮点数值代表，那么这个元素就会被使用（注意：字符点'.'被用作十进制分隔符）；
- <inst_attr_int8>：如果实例属性值能用 1 个字节带符号的整数值代表，那么这个元素就会被使用；
- <inst_attr_int16>：如果实例属性值能用两个字节带符号的整数值代表，那么这个元素就会被使用；
- <inst_attr_int32>：如果实例属性值能用四个字节带符号的整数值代表，那么这个元素就会被使用；
- <inst_attr_int64>：如果实例属性值能用八个字节带符号的整数值代表，那么这个元素就会被使用；

使用；

——<inst_attr_time>：如果实例属性值是一个时间/日期的信息并且是用按照 T_DATE 数据类型编码的字符串表示，那么这个元素就会被使用。

这些元素中的每一个元素都有 XML 属性 ‘name’。XML 属性的值指定实例属性的自定义名称。

这些元素中的每一个元素可能有 XML 属性 ‘unit’。如果给出这个 XML 属性值，那么它的值指定一个标识符（必须是 ATF / XML 文件中的一个实例的标识符）或引用一个实例的 ASAM 路径（如果它不包含在 ATF / XML 文件中，这个实例必须存在于任何一个输出 ATF / XML 文件的 ASAM/ODS 服务器中）。在这两种情况下，引用的实例必须是派生自 AoUnit 应用元素的一个实例。

示例：含有实例属性的实例。

下面给出的实例是由应用模型（在这未给出）的相关部分指定的，这个应用模型是关于除了实例属性‘altkey’, ‘passkey’, 和 ‘age’（存在于元素<instance_attributes>中的这些元素）的所有属性。它们中的其中一个元素含有一个对单元的引用；因此标识符为 17 的单元必须存在于这个 XML 文件的其它地方。

注意：按照 6.2.40 节，属性‘version_date’已经在这个例子中被转换。

```
...
<User>
  <UserId>22</UserId>
  <UserName>Todd Martin</UserName>
  <UserDescription>Sub user group 1b</UserDescription>
  <UserVersion>1B</UserVersion>
  <version_5Fdate>199708101105</version_5Fdate>
  <password>j</password>
  <UserDepartment>OP/ABC</UserDepartment>
  <AliasName>TM</AliasName>
  <groups>30 40</groups>
  <instance_attributes>
    <inst_attr_int32 name="altkey">123</inst_attr_int32>
    <inst_attr_asciistring name="passkey">hello</inst_attr_asciistring>
    <inst_attr_int8 name="age" unit="17">37</inst_attr_int8>
  </instance_attributes>
</User>
...
```

10.2.42 使用外部文件组件

在 ASAM/ODS 中，测量数据被基础元素 AoLocalColumn 描述。这些局部列被测量量和子矩阵引用。而另外的像 AoMeasurementQuantity 和 AoSubmatrix 元素会被像正常基础元素一样处理，局部列通常引用大的外部二进制数据集。

这些数据集可以使用三种不同的方法代表 ATF / XML：

——直接在派生自基本关系'**values**'的关系中，只是作为一个值序列，不使用任何组件创建；没有进一步的文件引用。应该注意，根据基础模型，尽管“值”是一个关系，但在 ATF/XML 文件中它被例外的作为属性值对待。

派生自基本属性'**external_component**'（如果存在）的属性值在实例中会被忽略。

对于明确的数据，派生自基本属性'**sequence_representation**'的属性被设置为'**explicit**'。

——在派生自基本关系'**values**'的关系中，不是作为一个值的序列，而是作为一个 ATF/XML 组件；这描述了一个到二进制文件的链接，这个二进制文件含有值以及值怎样在那个文件中被定位的。在实例中派生自基本属性'**external_component**'（如果存在）的属性值会被忽略。对于明确的数据派生自基本属性中'**sequence_representation**'的属性被设置为'**explicit**'。

——在混合模式中，使用派生自基本关系'**external_components**'的关系。在实例中派生自基本属性'**values**'的属性会被忽略，并且所有其它属性不变（例如派生自基本属性'**sequence_representation**'的属性，对于数据'**external_component**'是明确的，对于未知的数据它或者是 '**raw_linear_external**', '**raw_polynomial_external**', 或者是 '**raw_linear_calibrated_external**'）。

第一种方法对于小的和中等的数据来说是个简单的方法，它的使用在 6.2.39 节已经介绍了。

第三种方法支持混合模式概念并从 ATF/XML 中独立出来，它不使用任何特殊的结构。

相似于 ATF/CLA 中的使用，对于通过 ATF/XML 传输中等大小的或大型的数据，第二种方法方法是一种经济的方法。主要因为这个原因，COMPONENT 结构在 ATF/CLA 和 ATF/XML 中被提供。这个结构允许大量的二进制数据以相当紧凑的方式存储在 ATF/XML 外，因此在传输和存储上可以节省大量的时间和空间。

这个结构允许大量的二进制数据以相当紧凑的方式存储在 ATF/XML 外，因此在传输和存储上可以节省大量的时间和空间。

每一个外部文件组件必须在<files>模块内被定义（见 10.2.14）。一旦被定义，文件的引用就应当被插入，而不是值本身的序列。文件本身被<identifier>元素的内容引用，另一些元素指定这些值如何在文件内被发现。

仅有派生自基本属性'**flags**'的应用属性加上派生自局部列的基本关系'**values**'的关系可能将它们的值放在外部文件组件中。这种情况下，标签内的内容而不是它们本身是元素<component>。

<instance_data>内组件的结构见图 104。

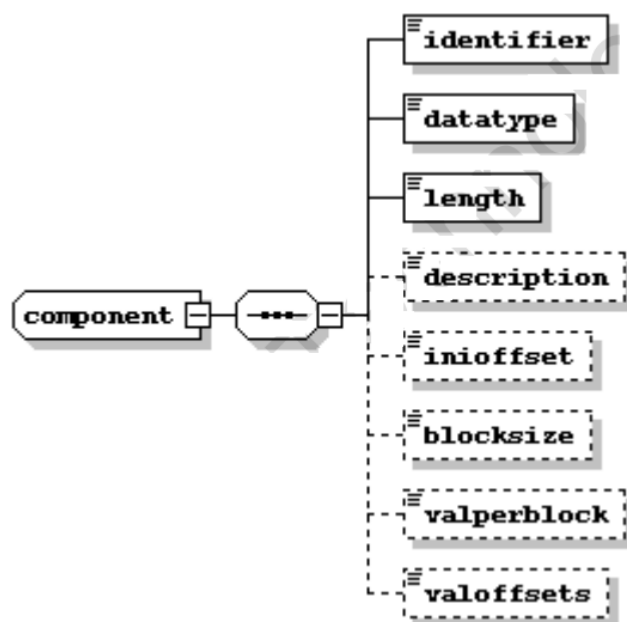


图104 <instance_data>内组件的结构

元素<component>（类型'component'）描述了这些值是如何从使用的外部文件中得到的，就像它们已经被本地存储一样。它没有 XML 属性，并且包含一系列元素<identifier>, <datatype>,<length>和可选择性的<description>, <inioffset>, <blocksize>, <valperblock>和<valoffsets>。

第一个元素是<identifier>。它没有 XML 属性，<identifier>的内容必须与组件其中之一<identifier>的内容相符，这些组件被列在 ATF/XML 文件开始的元素<files>中。它是用来验证文件的名称的，这个文件包含了局部列的测量数据。

元素<datatype>指定数据存储的方式。它没有 XML 属性，包含了已用数据类型的关键词。在 ATF/XML， ASAM ODS 的 typespec_enum 的枚举项如下：

- 1) dt_boolean;
- 2) dt_byte;
- 3) dt_short;
- 4) dt_long;
- 5) dt_longlong;
- 6) ieeefloat4;
- 7) ieeefloat8;
- 8) dt_short_beo;
- 9) dt_long_beo;
- 10) dt_longlong_beo;
- 11) ieeefloat4_beo;
- 12) ieeefloat8_beo;
- 13) dt_string;

- 14) dt_bytestr;
- 15) dt_blob;
- 16) dt_bytestr_beo;
- 17) dt_sbyte;
- 18) dt_ushort;
- 19) dt_ushort_beo;
- 20) dt_ulong。

这些案例中每个二进制文件中数据的存储方式将在这节做进一步描述。

注意，这列关键词的列表与 ATF/XML 的第一版本不同。

也应注意，<datatype>的内容并不是数据类型的名称，而是 typespec_enum 枚举项的名称。

此外应注意的是，ASAM ODS 支持组件文件有符号的数据类型 type dt_sbyte，没有符号的数据类型 dt_ushort, dt_ushort_beo, dt_ulong, 和 dt_ulong_beo。因此，这种数据类型的值无论在什么时候通过 API 方法写入到 ODS 服务器上，它首先转换为由 API 支持的合适的数据类型（通常是下一个更大尺寸）。当通过 API 读取值时，它们将总会在 API 数据类型中的其中一个被提供。

元素<length>没有 XML 属性，包含一个整数值，这个整数值指定来自该文件的数据要被读取。

对于字符串和流数据类型（dt_string, dt_bytestr, dt_blob, 和 dt_bytestr_beo），它表示要读取的字节数目。字符串的终止符 NULL-字符计入长度信息，像位字段和位节流内四个字节长度信息一样。

对于复杂数据类型，它表示了要读取的浮点值的数目（它是复数数目的两倍，因此必须是一个整数）。

对于其它的（整数，浮点值，布尔值，字节），它表示了要读取的值数目。在对于 dt_blob，长度规范与 blob 所在文件的长度是相等的，因为仅有一个 blob 存于组件文件中。所有其它类型可能出现在组件文件的序列中。

元素<description>除了出现在 dt_blob 中，不得再出现。它没有 XML 属性，包含了 blob 标题字符串。

元素<inioffset>没有 XML 属性，包含一个指定文件标题字节数目的整数值。在读取文件时，文件标题会被跳过。

文件的其它部分被看作是一个或更多块。

由于一个 BLOB 总是包含整份文件，所以元素<inioffset>在 dt_blob 中是省略的。

元素<blocksize>没有 XML 属性，包含一个指定文件中每个模块大小的整数值。标题之后，文件的主要部分应当包括一个或多个模块。一个模块可以由属于不同组件的值组成而且可以有不同数据类型（如果是数值数据）。模块内几个值可以属于相同元素，见本节末尾的例子。对于所有数值型数据类型，元素<blocksize>必须被指定，对于其它数据类型元素<blocksize>必须省略。

元素<valperblock>没有 XML 属性，包含一个指定属于目标组件模块中值编号的整数值。对于每一个整数值，每个模块开始部分的位移必须以字节形式指定。所有这些位移都在元素<valoffsets>中给出。

元素<valoffsets>没有 XML 属性，包含了一系列由空格分开的一个或多个位移。元素中位移值的数目必须是一或者等于<valperblock>的值。如果位移值数目为一，则假定模块中本地列中的数值以给出的位移中的第一个开始，不留空格的一个接一个排列。如果它们不是一个接一个，则必须提供模块中的每一个数值的位移。见以下例子。

示例：组件声明。

这是一个关于组件声明的简单例子，它指定组件文件包含 11 个双重值；这些值在文件开头部分一个接一个呈现。

```
....
<component>
  <identifier>binary_file_1</identifier>
  <datatype>ieeefloat8</datatype>
  <length>11</length>
  <description>double precision floating-point numbers </description>
  <inioffset>0</inioffset>
  <blocksize>8</blocksize>
  <valperblock>1</valperblock>
  <valoffsets>0</valoffsets>
</component>
...
```

下节描述组件文件中不同类型值的编码。

(1) 关键词 DT_BOOLEAN

在二进制文件中每个布尔对象都应该用一个比特表示（0=false, 1=true）。这些比特会被填充到字节序列中。如果比特的数目不足以完全填补一个字节，那么该字节的其余部分应保持闲置。字节是从左到右填充的，也就是以最高有效位开始。在字节序列（by₁, by₂, by₃, ...）中，nth 布尔对象（n = 1 ... Nbi）可以通过使用下面的整数公式进行定位。假设一个字节中的数位位置是按编号（也是从左到右）7,6,5,4,3,2,1,0 排列，那么 nth 布尔对象位于数位位置 lbi（从 0, 1, ..., 7）的 lbyth 个字节（从 1 开始）：

$$lby = (n+7) / 8$$

$$lbi = lby*8 - n$$

使用的字节数目（Nby）可以通过整数算法由布尔对象数计算得到：

$$Nby = 1 + (Nbi-1) / 8$$

元素<length>内的值指定被编码的布尔对象数目，组件文件中的字节数目可以用以上方法计算出来。

(2) 关键词 DT_BYTE

这个没符号的数据类型简单地以字节表示。因此，<length>元素内的值表明序列中的字节值数目，如果是单个数值，其字节值设置为 1。

(3) 关键词 DT_SHORT 和 DT_SHORT_BE0

这个带符号的 16 位的整型数据类型以两个字节表示。如果关键词 dt_short 在<datatype>元素中给出，那么最低有效字节在前（较低地址），最高有效字节在后（较高地址）。如果关键词 dt_short_be0 被给出，那么最高有效字节在前，最低有效字节在后；这个顺序又叫做“大字节存储次序”。<length>元素内的值表明序列中的整数值数目，如果是单个数值，其值设置为 1。

(4) 关键词 DT_LONG 和 DT_LONG_BEO

这个有符号的 32 位的整型数据类型以 4 个字节表示。如果关键词 `dt_long` 在 `<datatype>` 元素中给出,那么最低有效字节在前(较低地址),最高有效字节在后(较高地址)。如果关键词 `dt_long_beo` 被给出,那么最高有效字节在前,最低有效字节在后(大字节存储次序)。`<length>` 元素内的值表明序列中的整数值数目,如果是单个数值,其值设置为 1。

(5) 关键词 DT_LONGLONG 和 DT_LONGLONG_BEO

这个有符号的 64 位的整型数据类型以 8 个字节表示。如果关键词 `dt_longlong` 在 `<datatype>` 元素中给出,那么最低有效字节在前(较低地址),最高有效字节在后(较高地址)。如果关键词 `dt_long_beo` 被给出,那么最高有效字节在前,最低有效字节在后(大字节存储次序)。`<length>` 元素内的值表明序列中的整数值数目,如果是单个数值,其值设置为 1。

(6) 关键词 IEEEFLOAT4 和 IEEEFLOAT4_BEO

这个数据类型以 4 个字节表示。如果关键词 `ieeefloat4` 在 `<datatype>` 元素中给出,那么最低有效字节在前(较低地址),最高有效字节在后(较高地址)。如果关键词 `ieeefloat4_beo` 被给出,那么最高有效字节在前,最低有效字节在后(大字节存储次序)。符号的位置,指数,尾数在 IEEE 754 中有描述。`<length>` 元素内的值表明序列中的浮点值的数目,如果是单个数值,其值设置为 1。

(7) 关键词 IEEEFLOAT8 和 IEEEFLOAT8_BEO

这个数据类型以 8 个字节表示。如果关键词 `ieeefloat8` 已给出,那么最低有效字节在前(较低地址),最高有效字节在后(较高地址)。如果关键词 `ieeefloat8_beo` 被给出,那么最高有效字节在前,最低有效字节在后(大字节存储次序)。符号的位置,指数,尾数在 IEEE 754 中有描述。`<length>` 元素内的值表明序列中的浮点值的数目,如果是单个数值,其值设置为 1。

(8) 关键词 DT_STRING

这个数据类型以一系列字节表示。它作为枚举值由它的字符串代表给出,而不是它的整数项值给出,因此它按照字符串处理。每个字符串被 NULL 字节终止,这个 NULL 字节为那个字符串增加 1 个字节空间。在字符串序列中,每个字符都有一个终止 NULL 字节,这些字节都包含在组件的总长度中,这个组件在 `<length>` 元素的值中给出。在同一组件文件中,与 `dt_string` 相对应的数据不能与其它数据类型相混合。

(9) 关键词 DT_BYTESTR 和 T_BYTESTR_BEO

这个数据类型以字节表示。在字节域/字节流中,前面的 4 个字节表示紧随这四个字节之后的字节数目。在一字节域序列中,下一个长度的显示紧随先前序列的最后一个字节。请注意,4 个长度指示字节总是在“大字节存储次序”中给出,也就是最高有效字节在前。

示例:如果它们在大字节存储次序中是 00 00 0007 41 42 43 44 45 46 47 00 00 00 03 58 59 5A,那么内容为“ABCDEFGH” and “XYZ”的 2 个字节域/字节域以十六进制形式表示。

总长度是 18 字节，因此<length>元素的值是 18。

与 dt_bytestr 和 dt_bytestr_beo 相对应的数据在同一组件文件中不能与其它的数据类型相混合。

（10）关键词 DT_BLOB

blob 是一个有已知长度和标题的字节流，然而，在 ASCII 格式中，blob 由两个字符串表示（第一个包含标题，而另外一个包含字节流），这在组件中处理不同：

- blob 的标题应当进入<description>元素的内容中；
- blob 的长度由<length>元素的内容给出；
- blob 本身位于二进制文件中。

与 dt_blob 相对应的数据在同一组件文件中不能与其它的数据类型相混合。只有一个 blob 可以包含在组件文件中。

（11）关键词 DT_SBYTE

这个有符号的数据类型简单地以单个字节表示。因而<length>元素内的值表明序列中字节值的数目，如果是单个数值，其字节值设置为 1。

（12）关键词 DT_USHORT 和 DT_USHORT_BE0

无符号的 16 位的整型数据类型以 2 个字节表示。如果关键词 dt_ushort 在<datatype>元素中给出，那么最低有效字节在前（较低的地址），最高有效字节在后（较高地址）。如果关键词 dt_ushort_beo 被给出，那么最高有效字节在前，最低有效字节在后；这个命令也叫做“大字节存储次序”。<length>元素内的值表明序列中整数值的数目，如果是单个数值，其值设置为 1。

（13）关键词 DT_ULONG 和 DT_ULONG_BE0

无符号的 32 位的整数数据类型以 4 个字节表示。如果关键词 dt_ulong 在<datatype>元素中给出，那么最低有效字节在前（较低地址），最高有效字节在后（较高地址）。如果关键词 dt_ulong_beo 被给出，那么最高有效字节在前，最低有效字节在后（大字节存储次序）。<length>元素内的值表明序列中整数值的数目，如果是单个数值，其值设置为 1。

（14）复数的表示法

无论局部列的值是否复杂，这些值都会在相应测量量的基本属性“datatype”的应用属性中被指定。组件文件描述中不会给复数以特定的关键词，但是这些复数被<datatype>元素指定的关键词 ieeefloat4 和 ieeefloat8（相关的 ieeefloat4_beo 和 ieeefloat8_beo）控制。每个复数都以两个连续的浮点数表示。在<length>中指定的值总数必须是复数（也就是，在<length>元素中合法的值）数目的两倍。对于单个复数，<length>元素内的值是 2，对于一个由 7 个复数组成的序列，值是 14。复杂序列的顺序是：

- 第一序列成员的实部；
- 第一序列成员的虚部；
- 第二序列成员的实部；

——第二序列成员的虚部;

—— 。

以下例子说明了如何通过 **ATF/XML** 存储测量数据。.

示例1: 没有<COMPONENT>的本地列。

这个例子说明了 **ATF/XML** 文件中带有测量数据的本地列的一个实例。尽管根据基础模型“值”是一个关系，但是它被例外的看成是一个属性，并且进一步指定数据类型，这种数据类型通过特定标签<A_FLOAT32>将在 **ATF/XML** 文件的值编码。

应用模型并未在此说明。派生自基础元素 **AoLocalColumn** 的应用元素的名称是'**LocalColumn**'，基本的应用程序/关系的名称被选择出来以与相应的基本属性/关系保持一致。

```
<LocalColumn>
  <id>2</id>
  <name>MD</name>
  <global_flag>0</global_flag>
  <independent>>false</independent>
  <minimum>150.</minimum>
  <maximum>255.</maximum>
  <sequence_representation>explicit</sequence_representation>
  <values>
    <A_FLOAT32>
      150. 160. 170. 180. 190. 200. 210. 220. 230. 240. 250. 255. 250. 230.
    </A_FLOAT32>
  </values>
  <submatrix>1</submatrix>
  <measurement_quantity>2</measurement_quantity>
</LocalColumn>
```

示例2: 一个文件的本地列。

这个例子申明单独组件文件中带有测量数据的本地列的一个实例。尽管根据基础模型“值”是一个关系，但是它在这被例外处理，并且在<component>元素中它包含了如何在外部文件中发现本地列值的描述。

应用模型并未在此说明。派生自基础元素 **AoLocalColumn** 的应用元素的名称是'**LocalColumn**'；基本应用属性/关系的名称被选择出来与相应的基本属性/关系保持一致。

本地列应用使用由 **binary_file_1** 鉴别的文件，它先前必须被文件部分定义（在此也未能显示）。这个文件包含了一个测量量，该测量量有一个具有 **11** 个双精度浮点数的测量值通道，在文件的一开始，这些浮点数一个接一个放在文件中。

```
...
<LocalColumn>
  <id>2</id>
  <name>MD</name>
```

```

    <global_flag>0</global_flag>
    <independent>false</independent>
    <minimum>150.</minimum>
    <maximum>255.</maximum>
    <sequence_representation>explicit</sequence_representation>
    <values>
      <component>
        <identifier>binary_file_1</identifier>
        <datatype>ieeefloat8</datatype>
        <length>11</length>
        <inioffset>0</inioffset>
        <blocksize>8</blocksize>
        <valperblock>1</valperblock>

        <valoffsets>0</valoffsets>
      </component>
    </values>
    <submatrix>1</submatrix>
    <measurement_quantity>2</measurement_quantity>
  </LocalColumn>
  ...

```

示例3：一个文件中的两个本地列：一个接一个。

这个例子说明了单独组件文件中带有测量数据的本地列的两个实例。尽管根据基础模型“值”是一个关系，但是它在这被例外处理，并且在<component>元素中它包含了如何在外部文件中发现本地列值的描述。

应用模型并未在此说明。派生自基础元素 AoLocalColumn 的应用元素的名称是'LocalColumn'；基本应用属性/关系的名称被选择出来与相应的基本属性/关系保持一致。

本地列应用由 **binary_file_2** 鉴别的文件，它先前必须在文件部分定义。这个文件包含了两个测量量：

——一个具有 10 个双精度浮点值的测量通道；

——一个具有 10 个短整型值的测量通道。

第一个本地列在文件的一开始就发现了它的十个值。因此在文件中它不需要跳过初始字节。

```

  ...
  <LocalColumn>
    <id>2</id>
    <name>MD</name>
    <global_flag>0</global_flag>
    <independent>false</independent>

```

```

<minimum>150.</minimum>
<maximum>255.</maximum>
<sequence_representation>explicit</sequence_representation>
<values>
  <component>
    <identifier>binary_file_2</identifier>
    <datatype>ieeefloat8</datatype>
    <length>10</length>
    <inioffset>0</inioffset>
    <blocksize>8</blocksize>
    <valperblock>1</valperblock>
    <valoffsets>0</valoffsets>
  </component>
</values>
<submatrix>1</submatrix>
<measurement_quantity>2</measurement_quantity>
</LocalColumn>
...

```

文件中的第二个本地列在第一个之后。因为在同一文件中它紧随第一个本地列，所以它的初始位移值设定为 80，这表示第一列的长度是 80 字节。第二列被定义为：

```

...
<LocalColumn>
  <id>3</id>
  <name>QS</name>
  <global_flag>0</global_flag>
  <independent>false</independent>
  <minimum>125.</minimum>
  <maximum>2455.</maximum>
  <sequence_representation>explicit</sequence_representation>
  <values>
    <component>
      <identifier>binary_file_2</identifier>
      <datatype>dt_short</datatype>
      <length>10</length>
      <inioffset>80</inioffset>
      <blocksize>2</blocksize>
    </component>
  </values>
</LocalColumn>

```

```

        <valperblock>1</valperblock>

        <valoffsets>0</valoffsets>

    </component>

</values>

    <submatrix>1</submatrix>

    <measurement_quantity>2</measurement_quantity>

</LocalColumn>

...

```

示例4：一个文件中的两个本地列：一个接一个。

这个例子说明了单独组件文件中带有测量数据的本地列的两个实例。尽管根据基础模型“值”是一个关系，但是它在这被例外处理，并且在<component>元素中它包含了如何在外部文件中发现本地列值的描述。

应用模型并未在此说明。派生自基础元素 **AoLocalColumn** 的应用元素的名称是'**LocalColumn**'；基本应用属性/关系的名称被选择出来与相应的基本属性/关系保持一致。

本地列使用由 **binary_file_2** 鉴别的文件，它先前必须在文件部分定义。文件包含两个以相同采样率进行测量的测量量：

- 一个已经生成的时间通道（**1000** 个双精度浮点值）；
- 一个已测量的通道（**1000** 个短整型值）。

这些值被交替的写入文件。它们被两个本地列引用。

第一个局部列在文件的一开始部分就发现了它的第一个值。因此在文件中它不需要跳过初始字节。

它在文件中每隔十个字节就会发现后面的值；因此十的程序块尺寸和零的 **valo** 位移是恰当的。

```

...

<LocalColumn>

    <id>6</id>

    <name>MQ</name>

    <global_flag>0</global_flag>

    <independent>false</independent>

    <minimum>150.</minimum>

    <maximum>255.</maximum>

    <sequence_representation>explicit</sequence_representation>

    <values>

        <component>

            <identifier>binary_file_2</identifier>

            <datatype>ieeefloat8</datatype>

            <length>1000</length>

            <inioffset>0</inioffset>

            <blocksize>10</blocksize>

```

```

        <valperblock>1</valperblock>
        <valoffsets>0</valoffsets>
    </component>
</values>
<submatrix>1</submatrix>
<measurement_quantity>2</measurement_quantity>
</LocalColumn>
...

```

第二个局部列发现它的每个值都是分离的 10 个字节，第一字节位置是 8。因此，因此在文件中它不需要跳过初始字节，十的程序块尺寸和零的 **valo** 位移是恰当的。另一种定义也是可能的。

```

...
<LocalColumn>
    <id>3</id>
    <name>QS</name>
    <global_flag>0</global_flag>
    <independent>false</independent>
    <minimum>125.</minimum>
    <maximum>2455.</maximum>
    <sequence_representation>explicit</sequence_representation>
    <values>
        <component>
            <identifier>binary_file_2</identifier>
            <datatype>dt_short</datatype>
            <length>1000</length>
            <inioffset>0</inioffset>
            <blocksize>10</blocksize>
            <valperblock>1</valperblock>
            <valoffsets>8</valoffsets>
        </component>
    </values>
    <submatrix>1</submatrix>
    <measurement_quantity>2</measurement_quantity>
</LocalColumn>
...

```

示例5：同一文件中有不同采样率的三个本地列。

这个例子说明了带有单独组件文件中包含的测量数据的本地列的三个实例。尽管根据基础模型“值”是一个关系，

但是它在这被例外处理，并且在<component>元素中它包含了如何在外部文件中发现本地列值的描述。

应用模型并未在此说明。派生自基础元素 AoLocalColumn 的应用元素的名称是'LocalColumn'；基本应用属性/关系的名称应于相应的基本属性/关系一致。

本地列使用被 binary_file_5 鉴别的文件，它先前必须在文件部分定义。文件包含三个以不同采样率进行测量的测量量：

- 一个具有 10 Hertz 双精度浮点数的已测量的值通道；共有 1000 个值；
- 一个具有 20 Hertz 短整型的已测量的值通道；共有 2000 个值；
- 一个具有 30 Hertz 长整型的已测量的值通道；共有 3000 个值。

因为这些值来自测量仪器，所以它们被写入文件。在第一个测量点之前，设备发送了一个 32 字节参数标题。因此文件内通道序列是 1-2-3-3-2-3-1-2-3-3-2-3-...，它是周期性连续的。相应的字节长度是 8-2-4-4-2-4-8-2-4-4-2-4-...，也是周期性连续的，在 32 字节文件标题之后。

第一个本地列在文件的开头部分之后发现了它的第一个 32 字节值。

因此它跳过文件的 32 个初始字节。它在文件中每隔 24 个字节就会发现后面的值；因此 24 的程序块尺寸和零的 valoffsets 是恰当的。

```
...
<LocalColumn>
  <id>8</id>
  <name>MQ</name>
  <global_flag>0</global_flag>
  <independent>false</independent>
  <minimum>150.</minimum>
  <maximum>255.</maximum>
  <sequence_representation>explicit</sequence_representation>
  <values>
    <component>
      <identifier>binary_file_5</identifier>
      <datatype>ieeefloat8</datatype>
      <length>10000</length>
      <inioffset>32</inioffset>
      <blocksize>24</blocksize>
      <valperblock>1</valperblock>
      <valoffsets>0</valoffsets>
    </component>
  </values>
  <submatrix>1</submatrix>
  <measurement_quantity>2</measurement_quantity>
</LocalColumn>
```

...

第二个本地列在第一个本地列值之后发现它的第一，第三，第五个值，中间值总是 10 个字节。它发现文件中每隔 24 个字节会有一对这样的值，因此，24 的程序尺寸模块和 8 与 18 的 **valoffsets** 是恰当的。在文件中它也必须跳过 32 个初始字节。

...

```

<LocalColumn>
  <id>9</id>
  <name>QS</name>
  <global_flag>0</global_flag>
  <independent>>false</independent>
  <minimum>125.</minimum>
  <maximum>2455.</maximum>
  <sequence_representation>explicit</sequence_representation>
  <values>
    <component>
      <identifier>binary_file_5</identifier>
      <datatype>dt_short</datatype>
      <length>20000</length>
      <inioffset>32</inioffset>
      <blocksize>24</blocksize>
      <valperblock>2</valperblock>
      <valoffsets>8 18</valoffsets>
    </component>
  </values>
  <submatrix>2</submatrix>
  <measurement_quantity>3</measurement_quantity>
</LocalColumn>

```

...

第三个本地列在第二个本地列的第一个值之后立即发现它的第一和第二个值相互之间很接近，而它的第三个值在第二个本地列的第二个值之后。这每隔 24 个字节重复一次。因此，24 的程序尺寸块和 10,14 与 20 的 **valoffsets** 是恰当的，在文件中它也必须跳过 32 个初始字节。

...

```

<LocalColumn>
  <id>10</id>
  <name>QM</name>
  <global_flag>0</global_flag>
  <independent>>false</independent>

```

```

    <minimum>125.</minimum>

    <maximum>2455.</maximum>

    <sequence_representation>explicit</sequence_representation>

    <values>
        <component>
            <identifier>binary_file_5</identifier>
            <datatype>dt_long</datatype>
            <length>30000</length>
            <inoffset>32</inoffset>
            <blocksize>24</blocksize>
            <valperblock>3</valperblock>
            <valoffsets>10 14 20</valoffsets>
        </component>
    </values>

    <submatrix>3</submatrix>

    <measurement_quantity>4</measurement_quantity>
</LocalColumn>

...

```

10.2.43 存储安全性信息

与访问保护相关的信息可能被 ASAM ODS ATF/XML 文件传输。然而，应注意，这些 ATF/XML 文件不是加密的，而是可读的，尽管可能是耗时的，但是访问权限可以是人工分析和修改。

根据 ATF/XML 文件的目的，可以鉴别几个使用案例，例如：

a) 存档输出，之后又导入到相同系统中：为了能够完整地恢复数据，安全性信息（也就是与访问权限一起），用来伴随存档数据；

b) 数据输出，与其它系统，公司等数据进行数据交换；导入到不同的系统中：

- 1) 为了数据交换，需要写无安全信息的 ATF/XML 文件，因为如果新的更适合目标服务器的访问权限被指定，那么导入便会变得更加充分；
- 2) 几个安全性信息可以被写入。这些可能包括或者排除。

c) 派生自 AoUser 和 AoUserGroup 的应用元素（加上 ATF/XML 文件元数据部分中的应用元素的相应定义）的实例；

d) ACL 实体（ACLI, ACLA, ACL 模版）的数据集；

e) SecurityLevel 实体的数据集；

f) InitialRightsAt 实体的数据集。

ATF/XML 阅读客户端应当注意，安全性信息可能包含在文件中。客户端可以实现多种目标，因此安全性信息的处理可能会不同。

对于 ATF/XML 文件中或者是数据库中的数据的输入输出，使用者需要有必要的权利来存取数据对

象；对于访问 ATF/XML 文件，不需要权限。

总之，使用者需要以下访问权限来访问数据库：

——输出：阅读；

——删除输出的数据对象：删除；

——导入：插入和更新。

ASAM ODS 安全性概念的更多信息在第 5 章给出。

10.2.43.1 ATF/XML 的安全性信息写入

安全性信息包含：

a) 应用元素实例派生自

1) AoUser （这种应用元素必须在<application_model>元素中描述）；

2) AoUserGroup （这种应用元素必须在<application_model>元素中描述）。

b) 子类型的实体数据集属于

ACL （这些是： ACLA, ACLI, ACL模版）。

c) 以下两个实体的数据集：

1) 安全水准；

2) InitialRightsAt。

使用者不能派生实体ACL的子类型。仅有三个预先定义的子类型是被允许的：

d) ACLA （如果一个应用元素属性名称被指定，为了“元素安全性”或”属性安全性，用来指定应用元素的访问权限）；

e) ACLI （为了“实例安全性”，用来指定实例的访问权限）。

f) ACL 模版（用来指定初始权利：最新创建实例的访问权限；仅与“实例安全性”相关）。

SecurityLevel, InitialRightsAt 和 ACL 子类型不是应用元素，这些实体仅仅属于基础模型。因此它们并不在 ATF/XML 文件中的<application_model>元素中描述。

请注意，ACL 子类型中的“使用者”项是指派生自 AoUserGroup 的应用元素实例。因此，数据集项中的项值必须通过提供用户组的 ID 号来指定与用户组实例的关系（记住，仅有一个应用元素派生自 AoUserGroup；因此用户组实例的 ID 来唯一确定一个用户组是很充分的）。

没有特别的语法适用于安全性信息。按照所有其它的数据一样规则，安全性信息被写入到 ATF/XML 文件中。

g) 派生自 AoUser 和 AoUserGroup 的应用元素必须在 ATF/XML 文件中被定义，并且与其它应用元素一起被写入到应用模型部分（<application_model>元素）。ACL 子类型在 ATF/XML 文件的应用模型部分并没有被声明；

h) 这些实例与其它实例一起被写入到实例数据部分（<instance_data>元素）；

i) （SecurityLevel, InitialRightsAt, 和 ACL 子类型）的数据集就像实例一样被处理，并且必须符合<instance_data>元素的语法和语义规范。它们与实例一起被写入到数据部分。

下面的例子说明了如何在 ATF/XM 文件中存储安全性信息以及内容是如何解释的。

示例：用户和用户组声明。

下文给出了用户和用户组规范。Peter Sellers 属于第 30 和 50 组，Todd Martin 属于第 30 和 40 组，Otto Biermann 属于第 40 和 50 组。

应用模型在此并未申明。假定“User”是派生自 AoUser 的应用元素名称，'UserGroup'是派生自 AoUserGroup 的应用元素名称。基本的应用属性/关系的名称已经选择出来与相应的基本属性/关系保持一致。

```

...
<User>
  <id>21</id>
  <name>Peter Sellers</name>
  <description>Super user group 1</description>
  <version>1A</version>
  <version_date>199708101105</version_date>
  <password>^1kjws7hxoish^mjkkdjxo"e--@@@kjg798xh0880djdj90</password>
  <groups>30 50</groups>
</User>
<User>
  <id>22</id>
  <name>Todd Martin</name>
  <description>Sub user group 1b</description>
  <version>1B</version>
  <version_date>199708101105</version_date>
  <password>j</password>
  <groups>30 40</groups>
</User>
<User>
  <id>23</id>
  <name>Otto Bierman</name>
  <description>Sub user group 1</description>
  <version>1A</version>
  <version_date>199708101105</version_date>
  <password>89xzhuon3m,")) ?imi/U (%unpo39048ejdkádkxsos••o•3</password>
  <groups>40 50</groups>
</User>
<UserGroup>
  <id>30</id>
  <name>AQ 1</name>
  <description>Group1</description>
  <version>00</version>

```

```

    <version_date>199708101105</version_date>
    <superuser_flag>true</superuser_flag>
    <users>21 22</users>
  </UserGroup>
  <UserGroup>
    <id>40</id>
    <name>AQ 2</name>
    <description>Group 2</description>
    <version>00</version>
    <version_date>199708101105</version_date>
    <superuser_>false</superuser_flag>
    <users>22 23</users>
  </UserGroup>
  <UserGroup>
    <id>50</id>
    <name>TZU 23</name>
    <description>Group 3</description>
    <version>00</version>
    <version_date>199708101105</version_date>
    <superuser_flag>false</superuser_flag>
    <users>21 23</users>
  </UserGroup>
  ...

```

元素安全性和属性安全性由下面的 ATF/XML 编码例子指定，它们产生以下权限：

- Peter Sellers 和 Todd Martin 在名称为 AppEI01 的应用元素上，有权限'读取','更新',和'插入'（元素安全性）；
- Peter Sellers 和 Todd Martin 在名称为 AppEI02 的应用元素的名为 'description' 的应用属性上，有权限'读取','更新'（属性安全性）；
- Todd Martin and Otto Biermann 在名称为 AppEI03 的应用元素上，有权限'读取','删除'（元素安全性）。

```

  ...
  <ACLA>
    <users>30</users>
    <appl_element_name>AppEI01</appl_element_name>
    <rights>7</rights>
    <attribute_name/>
  </ACLA>
  <ACLA>
    <users>30</users>

```

```

    <appl_element_name>AppEl02</appl_element_name>

    <rights>3</rights>

    <attribute_name>description</attribute_name>
</ACLA>
<ACLA>

    <users>40</users>

    <appl_element_name>AppEl03</appl_element_name>

    <rights>9</rights>

    <attribute_name/>

</ACLA>

...

```

实例安全性被接下来的 ATF/XML 编码例子指定，它们产生以下权限：

- Peter Sellers 和 Otto Biermann 在名称为 AppEl03 应用元素的 ID23 实例上，有权限'读取'和 '删除'；
- Todd Martin 和 Otto Biermann 在名称为 AppEl02 应用元素的 ID17 实例上，有权限'读取'和'插入'。

10.2.44 继承概念和 ATF/XML

ASAM ODS 继承概念允许定义应用元素（子类），这个子类从另一个应用元素（超类）那继承应用属性和关系，并且还添加更深入的应用属性和扩展的应用关系。如果子类和超类元素以及它们的实例用 ATF/XML 文件传输，下面的规则要遵循。

应用模型规范的规则：

- a) 超类和子类应用元素必须在 ATF 文件内被<application_element>元素指定；
- b) 超类应用元素的规范定义它所有的应用属性和应用关系；
- c) 子类应用元素的规范只定义它的应用属性和应用关系，这些应用属性和应用关系是特定于子类的（因此在相应的超类中没有定义）；
- d) 必须在每个子类应用元素内指定，因超类应用元素从子类应用元素那继承。这由 XML 元素 <inherits_from>完成，<inherits_from>被放在指定子类的<application_element>内容中；
- e) 在继承过程中，对 ASAM ODS 服务器的访问要求应用模型包含一个应用属性，这个应用属性派生自超类和子类元素的基本属性'objecttype'；这是接口在选择超类或子集实例时所需要的。因此，如果要将 ATF/XML 文件内容写入 ASAM ODS 服务器，ATF/XML 文件应当在超类应用元素（这个元素后来被子类应用元素继承）内指定这个应用属性。如果没有这样的应用属性，则写这个应用模型的客户端在将这些信息导入到服务器之前必须生成它。请注意，这个属性的数据类型被 ASAM ODS 基础模型定义为 T_LONGLONG。
- f) 如果不打算把 ATF/XML 文件内容写入 ASAM ODS 服务器，ATF/XML 文件不需要提供派生自基本属性'objecttype'的应用属性。

实例规范的规则是：

- a) 因为超类应用元素不知道其它应用元素是否从它那继承，因此，它的实例不申明任何的特定行

为；

- b) 子类应用元素的每个实例被<instance_data>元素内一个 XML 元素准确给出,XML 元素的名称与子类应用元素的名称是相同的。它为所有应用属性和关系嵌入值,不管它们是否从超类应用元素那继承,或者个别地加入到子类元素中；
- c) 如果应用模型包含派生自基本属性'objecttype'的属性,那么对于那个属性(通常这个是 AID,如果 ATF 文件已经由服务器读取),实例可能包含一个正整数。可选择地,在实例中它可能被完全忽略(通常出现在 ATF 文件内容不被服务器读取时,因为在这种情况下,对这个属性有用的值是不知道的)；
- d) 如果应用模型包含一个派生自基本属性'objecttype'的属性,并且打算把 ATF/XML 文件内容写入 ASAM ODS 服务器,那么 ATF/XML 文件应该在超类和子集每个实例中为那个属性提供一个值。如果没有这样的值,那么把这个数据写入服务器的客户端必须在把这些信息写入服务器之前生成它；
- e) 如果应用模型包含一个派生自基本属性'objecttype'的属性,但是不打算把 ATF/XML 文件内容写入 ASAM ODS 服务器,那么 ATF/XML 文件不需要为此属性提供值；
- f) 如果应用模型没有包含一个派生自基本属性'objecttype'的属性,那么实例不会为 ATF/XML 文件的这个属性提供值。如果这个 ATF/XML 文件的内容要被客户端程序写入服务器,那么这个客户端程序必须生成一个派生自基本属性'objecttype'的应用属性(在它将这个应用模型写入服务器之前)。

此外,也必须为每个实例的属性生成一个适当的值(在它将这个实例写入服务器之前)。

g) 派生自基本属性'objecttype'的属性值在 T_LONGLONG 表示范围内是一个正整数；

h) 必须注意,在读取来自服务器的数据以及将它们写入 ATF/XML 文件时:超类的实例必须服从服务器的要求,将 ObjectType 设置为超类的应用 ID (AID)。

示例: ATF/XML 上的继承见图 105。

设有以下应用模型:

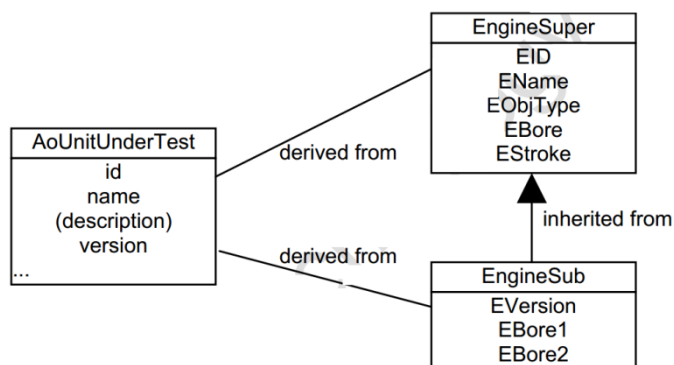


图105 ATF/XML 上的继承

应用模型的实例 EngineSuper (EName="SuperEng", EBores=80.4, EStroke=85.6), 实例 EngineSub (EName="SubEng", EBores=80.5, EStroke=85.3, EVersion="2.4", EBore1=81.5, EBore2=79.5) 可在 ATF/CLA 文件中如下表达。

相应的 ATF/XML 文件:

```
...
<application_model>
<application_element>
  <name>EngineSuper</name>
  <basetype>AoUnitUnderTest</basetype>
  <application_attribute>
    <name>EID</name>
    <base_attribute>id</base_attribute>
    <autogenerate>true</autogenerate>
    <obligatory>true</obligatory>
    <unique>true</unique>
  </application_attribute>
  <application_attribute>
    <name>EName</name>
    <base_attribute>name</base_attribute>
    <obligatory>true</obligatory>
  </application_attribute>
  <application_attribute>
    <name>EObjType</name>
    <base_attribute>objecttype</base_attribute>
  </application_attribute>
  <application_attribute>
    <name>EBore</name>
    <datatype>DT_DOUBLE</datatype>
  </application_attribute>
  <application_attribute>
    <name>EStroke</name>
    <datatype>DT_DOUBLE</datatype>
  </application_attribute>
</application_element>
<application_element>
  <name>EngineSub</name>
  <basetype>AoUnitUnderTest</basetype>
  <inherits_from>EngineSuper</inherits_from>
  <application_attribute>
    <name>EVersion</name>
```

```

        <base_attribute>version</base_attribute>
    </application_attribute>
    <application_attribute>
        <name>EBore1</name>
        <datatype>DT_DOUBLE</datatype>
    </application_attribute>
    <application_attribute>
        <name>EBore2</name>
        <datatype>DT_DOUBLE</datatype>
    </application_attribute>
</application_element>
</application_model>
<instance_data>
    <EngineSuper>
        <EID>27</EID>
        <ENAME>SuperEng</ENAME>
        <EObjType>4711</EObjType>
        <EBore>80.4</EBore>
        <EStroke>85.6</EStroke>
    </EngineSuper>
    <EngineSub>
        <EID>31</EID>
        <ENAME>SubEng</ENAME>
        <EObjType>815</EObjType>
        <EBore>80.5</EBore>
        <EStroke>85.3</EStroke>
        <EVersion>2.4</EVersion>
        <EBore1>81.5</EBore1>
        <EBore2>79.5</EBore2>
    </EngineSub>
</instance_data>

```

显示的示例文件包含'objecttype'的信息，尽管这不是严格要求的。特别是，如果 ATF/XML 文件的内容不需要导入 ASAM ODS 服务器，那么会省略这些行。

```

<application_attribute>
    <name>EObjType</name>
    <base_attribute>objecttype</base_attribute>
</application_attribute>

```

```

...
    <EObjType>4711</ENAME>
...
    <EObjType>815</ENAME>

```

10.3 ODS 基本的 XML 模式

10.3.1 验证 ATF/XML 文件

使用 ATF/XML 的好处之一是 XML 内置的验证能力。

- a) 任何一个 XML 解析器都能验证 ATF/XML 文件是否是“形式合法的”。如果元素具备有效的开始和结束标签，也被正确的嵌入，那么这个 ATF/XML 文件“形式合法”。
- b) 许多 XML 解析器通过由 XML 模式指定的内容模型验证相对的 ATF / XML 文件内容；如果这个验证是成功的，那么 XML 文件是“有效的”。

然而这些是 XML 的普通验证能力，它们不能用来保证 ATF/XML 文件 100%无错误，因为：

- a) ASAM ODS 首先为应用元素指定规则，并且给实例留有很大的自由度；
- b) XML 模式并不能够简单地指定 ASAM ODS 的所有规则。

ASAM ODS ATF/XML 规范包括了 ODS 基本的 XML 模式，这个模式可以用来验证 ATF/XML 文件的应用模型部分。这个模式保证在应用模型部分定义的模型在某种程度上是正确的模型。每个应用元素以及它们的属性和关系将会是有效的 XML 片段，这些片段符合 ODS 关于应用元素的基本 XML 模式定义。

然而，应用模型不仅要符合 ODS 基本的 XML 模式规则，也须符合 ASAM ODS 规范的所有规则。ODS 基本的 XML 模式并不指定 ASAM ODS 的所有规则，这就意味着即使文件已经验证是不符合 ODS 基本的 XML 模式的，它不一定能指定出符合 ASAM ODS 规则有效的应用模型。

ATF/XML 文件的实例数据部分不能由 ODS 基本的 XML 模式验证，因为实例数据的结构是特定的应用模型和潜在的 ODS 基础模型的结合。然而，创建一个特定应用模式是可能的，这个模式根据给出的应用模型或多或少的执行元素和属性的顺序和内容。这些将在 10.4 节作出进一步的描述。

10.3.2 ODS 基本 XML 模式的主要特征

ODS 基本 XML 模式在某种程度上允许 ATF/XML 文件的应用模型部分的验证。为此，它指定了所有的可能包含在 ATF/XML 文件中但不在<instance_data>元素中的 XML-元素。在 ODS 基本 XML 模式中被指定的<instance_data>元素可能包含没有任何限制的更深元素。因此<instance_data>内容是可接纳的，但是不能使用 ODS 基本 XML 模式验证。

ODS 基本 XML 模式分层次地建立在标准化的数据类型基础上。从 ASAM ODS 版本 5.1 开始，ODS 基本的 XML 模式文件已经明确的分离开来并具有潜在的层次。

- a) 那个层次的最低水平是 XML 模式数据类型。它们没有被 ASAM ODS 进一步指定，而是被相应的 W3C 工作组指定；
- b) 下一个水平，建立在 XML 模式数据类型上，被 ASAM 一致的数据类型给出。为了让不同 ASAM 水平的互操作性简易方便，ASAM 有一组常用的统一数据类型，也设置了一个相应的规范。它

们中的一些被 ASAMODS 应用。ASAM 一致的数据类型模式在'ASAM_HDTypes.xsd'中指定（见 6.5.1）。因为其它 ASAM 标准目前不需要用于 ASAM 统一数据类型的 XML 模式，因此这个模式仅指定被 ODS 使用的一致数据类型，所以这只是一个局部规范。此时，它是在 ASAM ODS 控制之下，但是将来可以被扩展来包含所有的 ASAM 一致数据类型，它的控制可以授权给 ASAM 内的某些内部标准组。ASAM 一致数据类型模式的进一步的陈述见 6.5.1；

- c) 第三个水平，建立在 XML 模式数据类型和 ASAM 统一数据类型上，最终由 ODS 基本的 XML 模式自身给出。它描述了 ODS 特定的类型以及 ATF/XML 文件中使用的元素，它在'schema.xsd'中被指定（见 10.5.2）。

图 106 显示的是 ASAM 模式文件和 ATF/XML 文件之间的关系。

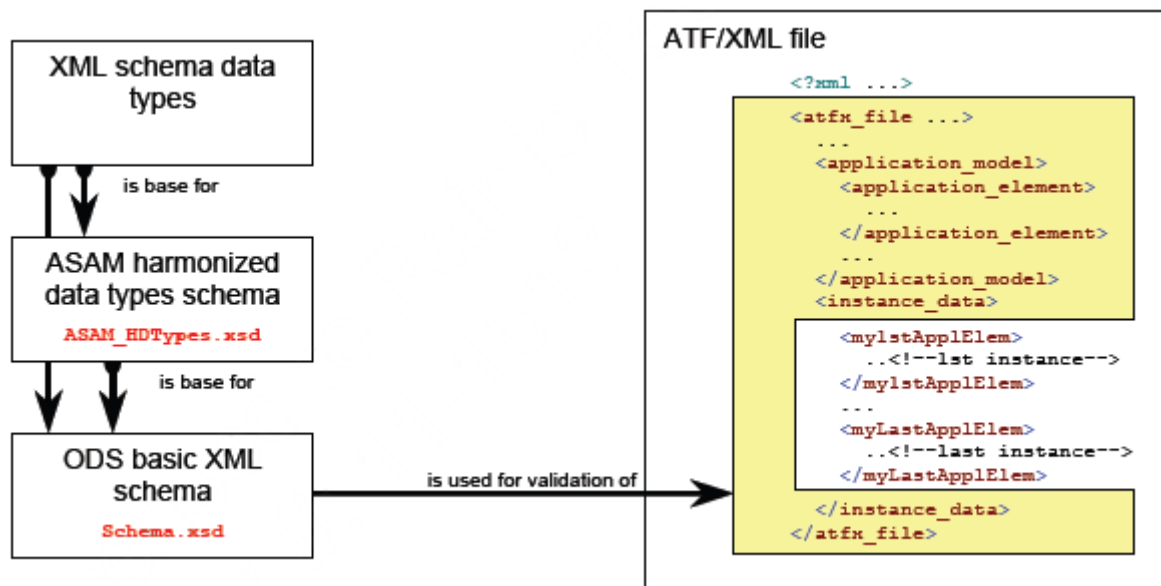


图106 ASAM 模式文件和 ATF/XML 文件

允许 XML 验证工具验证与 ODS 基本的 XML 模式相对的 ATF/XML 文件唯一要做的是，获得从 ASAM 网站下载的模式文件 ASAM_HDTypes.xsd 和 ASAM_HDTypes.xsd，把它们放在 ATF/XML 文件中同一目录下，然后，

- 或者用验证工具将 Schema.xsd 指定为 XML 模式来验证（通常是可能的）；
- 或者包括模式以及它在 ATF/XML 文件的根元素<atfx_file>中的位置。

这通过添加以下属性来完成：

```
xmlns: xsi=http://www.w3.org/2001/XMLSchema-instance
```

```
xsi:schemaLocation="http://www.asam.net/ODS/5.2.0/Schema ./Schema.xsd".
```

第二个选择是不提倡的，因为由于明确的命名空间位置，ASAM ODS 检查工具可能面对作为结果的 ATF/XML 文件的问题。

ASAM ODS 已经意识到在应用模型的类型安全规范方面，目前的 ATF/XML 规范的能力是有限的。应用模型必须服从海量的规则，XML 验证工具可能仅检查一小部分。这是由于应用模型在 ATF/XML

中被指定的方法，这个方法一开始就注重与 ATF/CLA 和 ATF/XML 功能保持相似，进而导致了非限制的 XML 模式。这种有限的检查性的几个例子是：

- a) 这个模式允许指定更多的基础元素的应用元素，其数量比 ODS 基础模型允许还多；
- b) 这个模式允许添加派生自基础元素 AoX 的应用元素。一些基本的应用属性和基本的应用关系并不是基础元素 AoX 的基本属性/关系；
- c) 这个模式允许跳过基本的应用属性和基本的应用关系，即使基础模型需要它们 (MANDATORY_ATTRIBUTE)。

因此，在 ATF/XML 规范的未来版本中，一个用于指定应用模型的全新方法将被 ASAM ODS 采纳。

10.4 特定应用模式注意事项

10.4.1 起因

因为在应用程序和应用程序之间的应用模式可能并且会是不同的，所以通用的 ODS 基本 XML 模式对于验证 ATF/XML 文件实例数据部分的语法是不充足的。如果实例数据的验证部分需要使用 XML 方法，那么特定的应用 XML 模式必须被创建，用来验证整个文件。下图申明了涉及到的不同 XML 模式文件之间的依赖关系。尽管 XML 模式数据类型没有明确地显示，但是它们在特定应用 XML 模式中可能仍然作为元素的基础来使用；如果其它的模式定义不提供恰当的类型，那么应当按此操作。

图 107 图示了应用 XML 模型验证文件。

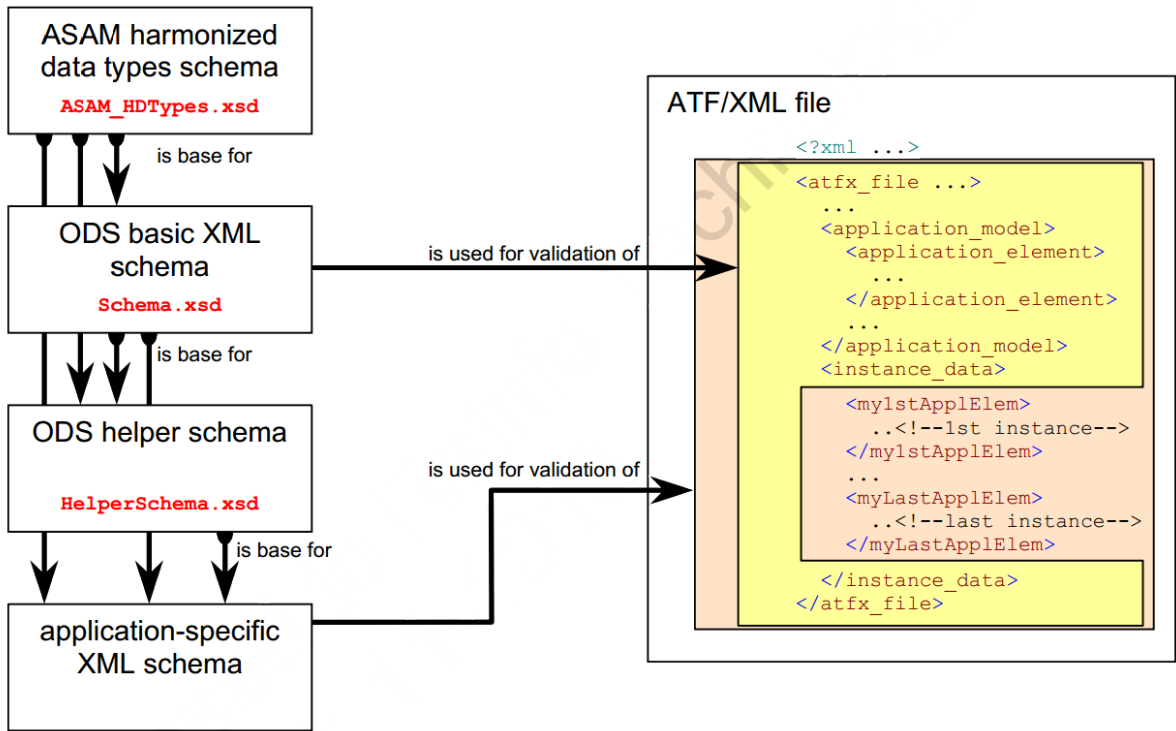


图107 使用 XML 模式来验证文件

像在 10.3 节解释的那样，只要标准化的 ASAM 数据类型被使用，ASAM 一致的数据类型模式就被用在 ODS 基本的 XML 模式中。ODS 基本的 XML 模式在某种程度上对于验证一个完整的 ATF/XML 文件是充足的；然而，由于<instance_data>元素被定义的方法，它能容忍<instance_data>内容，而非验证；它的定义是（'这里的 xsd'和章节的其它部分成为一个前缀，这个前缀是与模式语言名称空格有关系"http: //www.w3.org/2001/XMLSchema"）：

```
<xsd: elementname="instance_data">
  <xsd: complexType>
    <xsd: sequence>
      <xsd: anyprocessContents="skip"maxOccurs="unbounded"/>
    </xsd: sequence>
  </xsd: complexType>
</xsd: element>
```

这个定义同样允许<instance_data>内任何形式合法的内容，在此定义中 XML 元素及它们的属性被简单地忽略。

对于特定的应用模型，它是值得用 XML 验证工具来验证<instance_data>的内容的。在这种情况下，XML 元素<instance_data>必须被应用模型特定的 XML 模式重新定义。

实施上述内容并且不漏掉 ATF/XML 剩余文件的 ODS 基本模式的验证性能的方法是复制 ODS 基本的 XML 模式文件并且通过特定的应用定义来取代<instance_data>元素的定义。

定义一个应用程序或者伴随的特定模式层次有更多的复杂方法，在这些方法中（修改的）ODS 基本的 XML 模式和一些深层的模式规范一起被包含在下面的例子中。这允许模式定义的标准化并有利其重新使用。

示例：分层组织的模式集合。

```
<?xml version="1.0?>

<xsd: schematargetNamespace=http: //www.asam.net/ODS/5.2.0/Schema
  xmlns="http: //www.asam.net/ODS/5.2.0/Schema"
  xmlns: xsd="http: //www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified">

  <xsd: include schemaLocation="ODSBasicPart.xsd"/>
  <xsd: include schemaLocation="CorporateModel.xsd"/>
  <xsd: include schemaLocation="DepartmentModel.xsd"/>

</xsd: schema>
```

注意（像 10.3.2 节描述的那样），在验证期间建议存储本地的模式文件并安装验证工具，以便于它根据这些局部模式文件验证相 ATF/XML 文件。

以下描述了当为<instance_data>内容提供特定应用模式定义的时候，一些应当被考虑的方面。在此，它们解释了单独的可用的 ODS 辅助模式对于这些定义如何会有帮助。应考虑的部分是：

- a) 'hdt'是与 ASAM 一致数据类型模式命名空间"http://www.asam.net/AIS/2005/HDTypes"相关联的前缀。
- b) 'ods'是与 ODS 基本 XML 模式命名空间"http://www.asam.net/ODS/5.2.0/Schema"相关联的前缀
- c) 'odh'是与 ODS 辅助模式命名空间"http://www.asam.net/ODS/5.2.0/HelperSchema"相关联的前缀

一旦特定应用模式被开发，它就可以单独地用来验证 ATF/XML 文件。然而，这一步在技术上来说是不必要的，它可以提供另一个水平的保证，即这个文件必须满足商定的模型。例如，如果 OEM 仅仅想要确保供应商提供在某种程度上句法正确的 ATF/XML 文件，那么 OEM 只要求供应商仅使用 ODS 基本 XML 模式文件来验证这些文件即可。然而，如果 OEM 想要有一个更深入的验证，那么它应当要求供应商使用更全面的，特定应用模式来验证。

10.4.2 应用元素的模式定义

无论应用元素的实例在<instance_data>元素中何时出现，它都将会嵌入到一个标签中，这个标签的名称是应用元素的名称。从 XML 角度看，应用元素的名称组成了一个新的 XML-元素。因此，为了更加严格的 XML 验证，需要为这个 XML-元素提供一个模式定义。

尽管可能出现各种各样的特定应用模式方法，但是也应该提供一个典型的使用案例作为例子。

示例：应用元素和相应的模式定义。

应用元素'Engine'是派生自基础元素 AoUnitUnderTest 的，它有一些基本的应用属性/关系（EngineId, EngineName, EngineDescription, EngineVersion, EngineParts），还有一些深层的扩展的应用属性（ProtoDate, EType, Bore, Stroke, NumCyl, EUser）。

下面给出一个实例。

在 ATF/XML 中给出的每个应用元素实例都遵循同样的规则，因此一个 XML 模式可以用来验证这些实例。下面给出一个合理的模式定义以及深入的讨论：

- 模式准确地定义了属性和关系的数目以及名称。这用来检测类型错误和被遗忘的元素；
- 一些属性是由基础模型定义的。对于它们 ODS 辅助模式提供了可以被"odh: .."引用的 XML 模式数据类型。它用来检测基本应用属性的数据类型错误；
- 扩展的应用属性可以从 ASAM 一致的数据类型（来自命名空间"hdt: .."）或者 ODS 特定的数据类型（来自命名空间"ods: .."）中获取一个数据类型；
- 应用关系由数字给出，表示相关实例的 ID 值；因此它们可能被"odh: S_INT64"或者（至多允许一个相关的实例）"hdt: A_INT64"模式化。

...

```
<Engine>
  <EngineId>17</EngineId>
  <EngineName>TestE17</EngineName>
  <EngineDescription>Test engine no. 17</EngineDescription>
```

```

    <EngineVersion>V1.0</EngineVersion>
    <ProtoDate>20051017</ProtoDate>
    <EType>4CylMethanol</EType>
    <Bore>87.4</Bore>
    <Stroke>81.6</Stroke>
    <NumCyl>4</NumCyl>
    <EngineParts>3 5 11</EngineParts>
    <EUser>27</EUser>
  </Engine>
  ...
  ...
  <xsd: complexType name="Engine">
    <xsd: complexContent>
      <xsd: extension base="odh: AoUnitUnderTest">
        <xsd: sequence>
          <xsd: element name="EngineId" type="odh: id"/>
          <xsd: element name="EngineName" type="odh: name"/>

          <xsd: element name="EngineDescription" type="odh: description"
            minOccurs="0"/>

          <xsd: element name="EngineVersion" type="odh: version"
            minOccurs="0"/>

          <xsd: element name="ProtoDate" type="ods: T_DATE" minOccurs="0"/>
          <xsd: element name="EType" type="hdt: A_UTF8STRING" minOccurs="0"/>
          <xsd: element name="Bore" type="hdt: A_FLOAT32" minOccurs="0"/>
          <xsd: element name="Stroke" type="hdt: A_FLOAT32" minOccurs="0"/>
          <xsd: element name="NumCyl" type="hdt: A_INT16" minOccurs="0"/>
          <xsd: element name="EngineParts" type="odh: S_INT64" minOccurs="0"/>
          <xsd: element name="EUser" type="hdt: A_INT64" minOccurs="0"/>
        </xsd: sequence>
      </xsd: extension>
    </xsd: complexContent>
  </xsd: complexType>
  ...

```

可以看到，ODS 辅助模式提供了预先定义的类型集合在可应用时使用。

10.4.3 实例属性的模式定义

10.2.41 节描述了如何在一个实例中存储实例属性。ODS 辅助模式提供了元素<instance_attributes>的 XML 类型定义。如果一个特定的应用模式打算为应用元素的实例允许其实例属性，那么它可以把有 XML 类型的<instance_attributes>元素放入模式文件中，像下面的例子所示。

示例：包括实例属性的模式定义。

来自上面例子的应用元素'Engine'应当被模式化以便它的实例可能包含实例属性。如下文所示。

在 ATF/XML 文件中提供了实例数据例子，ATF/XML 文件是与给出的模式片段相对应的。

```
...
<xsd: complexType name="Engine">
  <xsd: complexContent>
    <xsd: extension base="odh: AoUnitUnderTest">
      <xsd: sequence>
        <xsd: element name="EngineId" type="odh: id"/>
        <xsd: element name="EngineName" type="odh: name"/>
        <xsd: element name="EngineDescription" type="odh: description"
          minOccurs="0"/>
        <xsd: element name="EngineVersion" type="odh: version"
          minOccurs="0"/>
        <xsd: element name="ProtoDate" type="ods: T_DATE" minOccurs="0"/>
        <xsd: element name="EType" type="hdt: A_UTF8STRING" minOccurs="0"/>
        <xsd: element name="Bore" type="hdt: A_FLOAT32" minOccurs="0"/>
        <xsd: element name="Stroke" type="hdt: A_FLOAT32" minOccurs="0"/>
        <xsd: element name="NumCyl" type="hdt: A_INT16" minOccurs="0"/>
        <xsd: element name="EngineParts" type="odh: S_INT64" minOccurs="0"/>
        <xsd: element name="EUser" type="hdt: A_INT64" minOccurs="0"/>
        <xsd: element name="instance_attributes" type="odh:
          instance_attributes" minOccurs="0" maxOccurs="1"/>
      </xsd: sequence>
    </xsd: extension>
  </xsd: complexContent>
</xsd: complexType>
...
...
<Engine>
  <EngineId>17</EngineId>
  <EngineName>TestE17</EngineName>
  <EngineDescription>Test engine no. 17</EngineDescription>
```

```

    <EngineVersion>V1.0</EngineVersion>

    <ProtoDate>20051017</ProtoDate>

    <EType>4CylMethanol</EType>

    <Bore>87.4</Bore>

    <Stroke>81.6</Stroke>

    <NumCyl>4</NumCyl>

    <EngineParts>3 5 11</EngineParts>

    <EUser>27</EUser>

    <instance_attributes>
        <inst_attr_asciistring name="error">Engine broke during test!
        </inst_attr_asciistring>
        <inst_attr_time name="errorTime">200512241153</inst_attr_time>
    </instance_attributes>
</Engine>

...

```

为<instance_attributes>定义一个特定的应用 XML 类型是可能的，而不是使用类型'odh:instance_attributes'。这个类型能够限制经选择的可能数据类型的子集的实例属性，这个可能的数据类型与特定应用元素的实例属性相对应。

创建一个不允许任何的实例属性被使用的应用模式也是可能的。

10.4.4 特定应用模式的进一步支持

ODS 辅助模式提供一些简单的序列类型定义，它们可以用来指定扩展的应用属性的类型。

ODS 辅助模式为文件组件规范提供了一个 XML 类型定义。这可允许本地列将它们的值放入组件文件或者禁止这种外部存储。

尽管 XML xPointer 机制允许 xPointer-aware 工具核实元素引用的 ids 是否有效，但是 ASAMODS 明确决定不使用这个机制，而是引用在应用属性实例中给出的标识符值，这个应用属性与基本属性“id”相关。因此，不会为了引用目的而在特定应用模式中预见特定的方法。然而，这也暗示，ATF/XML 文件的引用完整性不能被标准的 XML 工具核实。这并不重要，因为包含在 ATF/XML 文件中的引用实例可能会被 ASAM 路径引用，xPointer 机制不能核实 ASAM 路径。因此，引用完整性的检查需要特定的工具。

注意：ODS 辅助模式作为一种工具被提供，来方便特定应用模式的创建，来协调特定应用模式中的类型定义，最终减少 ATF/XML 文件中的错误。将来会根据这些经验来进一步开发它；因此，可能某些类型需要被添加，而其它的会被移除或修改。明确要求使用者对模式的有效性和本标准第二页上的邮件地址所需的更改给出反馈。

10.5 相关的模式文件

10.5.1 ASAM 一致的数据类型模式文件

指定 ASAM 一致数据类型的 XML 模式在下面给出。这也以纯文本文件形式使用，也可以从 ASAM 网站上下载得到。

ASAM 一致数据类型模式的备注：

- a) A_ASCIISTRING (A_ASCII) 目前被限制到 XML 数据类型 “string” 中。任何不与 XML 数据类型 'string' 相对应的值在验证过程中都会产生 “错误”；
- b) A_BOOLEAN 目前被限制到 XML 数据类型 “boolean” 中。这使得布尔类型元素的内容只有 '0', '1', 'false' 和 “true”；
- c) 有一些辅助类型只有在这个模式文件内有意义。

如果这一部分内容和分离的可用的模式文件之间有不同，分离的模式文件是这个引用。

```
<?xml version="1.0"?>
<xsd:schema targetNamespace="http://www.asam.net/AIS/2005/HDTypes"
  xmlns="http://www.asam.net/AIS/2005/HDTypes"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified">
  <xsd:annotation>
    <xsd:documentation>
      ASAM Harmonized Datatypes
      W3C Schema Version 1.0
      Copyright 2009 - Association for Standardization of Automation and
        Measuring Systems.
        All rights reserved.

      Revision History:
      August 2009 - removed the datatype enumeration as it is no longer
        kept in the harmonized data types as ASAM internal standard.
        Moved it to the ODS schema file.
      February 2006 - Rainer Bartz rbartz@online.de
        Added schema definition for A_UTF8STRING.
      August 2005 - Rainer Bartz rbartz@online.de
        Created the schema, based on the ASAM ODS schema specification
        (by extracting the general specs of ASAM Datatypes and thus
        allowing them to be reused)
      NOTE: This is currently just a subset of the ASAM harmonized
        datatypes, containing those types that are used by ASAM ODS.
        It should be seen as a starting point for future extension
        as more types are going to be used by ASAM ODS or the other
        ASAM working groups.
      Added type definitions for the following types:
```


A_ASCII

Removed type definitions for the following types, as they are currently not needed for ASAM ODS:

A_BCD, A_COUNTRY, A_LANGUAGE

Changed definitions of the following types:

A_ASCIISTRING, A_COMPLEX32, A_COMPLEX64, A_INT16, A_INT32,
A_UINT8, A_UINT16, A_UINT32, A_UINT64, A_ENUM, A_ASCIIIFIELD,
A_BITFIELD, A_BYTEFIELD

Added some helper types (which are currently not part of the ASAM harmonized datatypes) :

S_FLOAT, S_DOUBLE, S_UINT8, A_ASCIISTRING_4294967295,
S_UINT8_255, S_UINT8_4294967295

</xsd:documentation>

</xsd:annotation>

<!--

* declare simple ASAM harmonized datatypes (in alphabetical order)

* Remarks to specific mappings:

* A_ASCIISTRING: ASAM allows the hex representation of the individual
* characters to be 0x1..0xFF

* From them, an xsd:string does not allow the hex values 0x0..0x8,
* 0xB, 0xC, 0xE..01F

* This restriction has been accepted (otherwise A_ASCIISTRING may
* not base on xsd:string)

* A_BOOLEAN: ASAM allows the hex representation of 0x1..0xFF for
* 'true', 0x0 for 'false'.

* xsd:boolean only allows '1' or 'true' for true and '0' or
* 'false' for false

* This restriction has been accepted (otherwise A_BOOLEAN may not
* base on xsd:boolean)

* A_COMPLEX32, A_COMPLEX64: ASAM expects the most significant bytes
* (MSBs) to contain the real part

* No restriction could be formulated by this schema; it is
* expected that from the two consecutive

* floating point numbers (separated by a blank 0x20) the first
* one is the real part, the second is the imaginary part.

```

-->
<xsd:simpleTypeName="A_ASCII">
  <xsd:restrictionbase="xsd:string">
    <!--NOTE: to decide whether we restrict this to 1..255-->
    <!--xsd:patternvalue="(\p{IsBasicLatin}) *"/-->
    <!--xsd:patternvalue="(\p{IsLatinSupplement}) *"/-->
    <!--Must be "\p{IsLatin-1Supplement}) *" according to XML spec;
    but not accepted by some parsers -->
    <!--alt:value="(\p{IsBasicLatin}|\p{IsLatin-1Supplement}) *"
    -->

    <xsd:lengthvalue="1"/>
  </xsd:restriction>
</xsd:simpleType>
<xsd:simpleTypeName="A_ASCIISTRING">
  <xsd:restrictionbase="xsd:string">
    <!--NOTE: to decide whether we restrict this to 1..255-->
    <!--xsd:patternvalue="(\p{IsBasicLatin}) *"/-->
    <!--xsd:patternvalue="(\p{IsLatinSupplement}) *"/-->
    <!--Must be "\p{IsLatin-1Supplement}) *" according to XML spec;
    but not accepted by some parsers -->
    <!--alt: value="(\p{IsBasicLatin} | \p{IsLatin-1Supplement})
    * "
    -->

  </xsd:restriction>
</xsd:simpleType>
<xsd:simpleTypeName="A_UTF8STRING">
  <xsd:restrictionbase="xsd:string">
    <!--NOTE: No restriction. ASAM ODS however does not support all
    characters of UTF8 at the moment -->

  </xsd:restriction>
</xsd:simpleType>
<xsd:simpleTypeName="A_BOOLEAN">
  <xsd:restrictionbase="xsd:boolean"/>
</xsd:simpleType>
<xsd:simpleTypeName="A_COMPLEX32">
  <xsd:restrictionbase="S_FLOAT">
    <xsd:lengthvalue="2"/>

```

```

        </xsd:restriction>
</xsd:simpleType>
<xsd:simpleTypeName="A_COMPLEX64">
    <xsd:restrictionbase="S_DOUBLE">
        <xsd:lengthvalue="2"/>
    </xsd:restriction>
</xsd:simpleType>
<xsd:simpleTypeName="A_FLOAT32">
    <xsd:restrictionbase="xsd:float"/>
</xsd:simpleType>
<xsd:simpleTypeName="A_FLOAT64">
    <xsd:restrictionbase="xsd:double"/>
</xsd:simpleType>
<xsd:simpleTypeName="A_INT8">
    <xsd:restrictionbase="xsd:byte"/>
</xsd:simpleType>
<xsd:simpleTypeName="A_INT16">
    <xsd:restrictionbase="xsd:short"/>
</xsd:simpleType>
<xsd:simpleTypeName="A_INT32">
    <xsd:restrictionbase="xsd:int"/>
</xsd:simpleType>
<xsd:simpleTypeName="A_INT64">
    <xsd:restrictionbase="xsd:long"/>
</xsd:simpleType>
<xsd:simpleTypeName="A_ENUM">
    <xsd:restrictionbase="xsd:int"/>
</xsd:simpleType>
<!--
*****
* Note: Unsigned integer not yet used in ASAM ODS
*****
-->
<xsd:simpleTypeName="A_UINT8">
    <xsd:restrictionbase="xsd:short">
        <xsd:minInclusivevalue="0"/>
        <xsd:maxInclusivevalue="255"/>
    </xsd:restriction>
</xsd:simpleType>

```

```

    </xsd:restriction>
</xsd:simpleType>
<xsd:simpleTypeName="A_UINT16">
    <xsd:restrictionbase="xsd:int">
        <xsd:minInclusivevalue="0"/>
        <xsd:maxInclusivevalue="65535"/>
    </xsd:restriction>
</xsd:simpleType>
<xsd:simpleTypeName="A_UINT32">
    <xsd:restrictionbase="xsd:long">
        <xsd:minInclusivevalue="0"/>
        <xsd:maxInclusivevalue="4294967295"/>
    </xsd:restriction>
</xsd:simpleType>
<xsd:simpleTypeName="A_UINT64">
    <xsd:restrictionbase="xsd:integer">
        <xsd:minInclusivevalue="0"/>
        <xsd:maxInclusivevalue="18446744073709551615"/>
    </xsd:restriction>
</xsd:simpleType>
<!--*****
* declare complex ASAM harmonized datatypes (in alphabetical order)
* A_BITFIELD and A_BYTEFIELD are currently specified as a length
* information followed by a restricted list of bytes (each separated
* by a blank) . If a hexadecimal stream representation is preferred,
* these type definitions need to be revised
*****
-->
<xsd:complexTypename="A_ASCIIIFIELD">
    <xsd:sequence>
        <xsd:elementname="length"type="A_UINT32"/>
        <xsd:elementname="sequence"type="A_ASCIIISTRING_4294967295"/>
    </xsd:sequence>
</xsd:complexType>
<xsd:complexTypename="A_BITFIELD">
    <xsd:sequence>
        <xsd:elementname="length"type="A_INT16"/>

```

```

        <xsd:elementname="sequence" type="S_UINT8_255"/>
    </xsd:sequence>
</xsd:complexType>
<xsd:complexTypename="A_BYTEFIELD">
    <xsd:sequence>
        <xsd:elementname="length" type="A_UINT32"/>
        <xsd:elementname="sequence" type="S_UINT8_4294967295"/>
    </xsd:sequence>
</xsd:complexType>
<!--*****
* declare helper types (in alphabetical order)
*****
-->
<xsd:simpleType name="S_FLOAT">
    <xsd:listitemType="A_FLOAT32"/>
</xsd:simpleType>
<xsd:simpleType name="S_DOUBLE">
    <xsd:listitemType="A_FLOAT64"/>
</xsd:simpleType>
<xsd:simpleType name="S_UINT8">
    <xsd:listitemType="A_UINT8"/>
</xsd:simpleType>
<xsd:simpleType name="A_ASCIISTRING_4294967295">
    <xsd:restriction base="A_ASCIISTRING">
        <xsd:maxLength value="4294967295"/>
    </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="S_UINT8_255">
    <xsd:restriction base="S_UINT8">
        <xsd:maxLength value="255"/>
    </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="S_UINT8_4294967295">
    <xsd:restriction base="S_UINT8">
        <xsd:maxLength value="4294967295"/>
    </xsd:restriction>
</xsd:simpleType>

```

```
</xsd:schema>
```

10.5.2 ODS 基本的 XML 模式文件

ODS 基本的 XML 模式文件的内容如下. 其也可采用纯文本文件形式, 也可以从 ASAM 网站上下载得到。

如果本节的内容与独立的可用模式文件之间有不同, 则独立模式文件是其引用。

```
<?xml version="1.0"?>
<xsd:schematargetNamespace="http://www.asam.net/ODS/5.2.0/Schema"
xmlns="http://www.asam.net/ODS/5.2.0/Schema"
xmlns:hdt="http://www.asam.net/AIS/2005/HDTypes"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
<!--The ASAM harmonized datatypes schema must be placed in the
working directory at the moment -->
<xsd:importnamespace="http://www.asam.net/AIS/2005/HDTypes"
schemaLocation="ASAM_HDTypes.xsd"/>
<xsd:annotation>
<xsd:documentation>
ASAM-ODS Base Model Architecture
W3C Schema Version 1.0
Copyright 2009 - Association for Standardization of Automation
and Measuring Systems.
All rights reserved.
Revision History:
August 2009 - Rainer Bartz rbartz@online.de
Added the datatype enumeration as it was removed from
the HDTypes schema.
November 2006 - Rainer Bartz rbartz@online.de
Created inherits_from and added it to application_element.
Updated version to 1.2.
February 2006 - Rainer Bartz rbartz@online.de
Replaced A_ASCIISTRING by A_UTF8STRING as ODS will herewith prepare
supporting UTF-8 character set.
December 2005 - Rainer Bartz rbartz@online.de
Changed application_element to allow any sequence of application
attributes and relations with the drawback of no longer requiring at
```

least 2 attributes (id and name) .

Changed sequence within application model; now enumerations must appear before application elements.

November 2005 - Rainer Bartz rbartz@online.de

Removed ACL* attributes/relations from the list of base attributes and base elements as these are never used in application models.

October 2005 - Rainer Bartz rbartz@online.de

Separated the helper part from the ODS basic schema into a new file

Introduced exporter instead of creator in element documentation

Removed boolean_enum and replaced it by hdt:A_BOOLEAN

September 2005 - Rainer Bartz rbartz@online.de

Removed min_occurs and max_occurs from element application_attribute

Removed type DS_ID

Removed all sequence instance attributes and those of complex type

Changed component_length from a list to a single element

Introduced type reference to be either a 64 bit number (ID) or the ASAM path.

Added a unit attribute to all instance attribute elements

Moved 'values' from the list of relations to the list of attributes

August 2005 - Rainer Bartz rbartz@online.de

Extracted type definitions for ASAM harmonized datatype into a separate schema document with its own namespace (for reuse purposes) . Imported those definitions into this schema file, and added a prefix 'hdt:' to all locations where ASAM harmonized datatypes are used.

Replaced t_character by hdt:A_ASCII and removed type definition of t_character

Renamed type t_date to T_DATE

Removed type definitions of:

DT_DATE, DS_DIMENSION

Renamed sequence types: DS_DATE to S_DATE, DS_ASCIISTRING to

S_ASCIISTRING, DS_BLOB to S_BLOB, DS_BOOLEAN to S_BOOLEAN,

DS_BYTESTR to S_BYTESTR, DS_COMPLEX32 to S_COMPLEX32,

DS_COMPLEX64 to S_COMPLEX64, DS_FLOAT32 to S_FLOAT32, DS_FLOAT64

to S_FLOAT64, DS_ID to S_ID, DS_INT8 to S_INT8, DS_INT16 to

S_INT16, DS_INT32 to S_INT32, DS_INT64 to S_INT64

Removed type definition of time_sequence (use S_DATE instead)

Changed the following type definitions:

inst_attr_time

Renamed base_attributetype_enum to base_attribute_enum

Renamed element and type name relation_attribute to
application_relation

Renamed element enumeration_type to enumeration_name

Created two new type definitions (multiplicity, many) to provide
a more strict typing of the max_occurs element and changed the
max_occurs element accordingly

Changed type definitions of following types (replacing 'list' with
'restriction' as only one value may appear in instances) :

average, block_size, component_length, current_exp,
current_exp_den, datatype, default_datatype, default_rank,
default_type_size, factor, flags, flags_start_offset,
generation_parameters, global_flag, id, independent,
interpolation, length_exp, length_exp_den, luminous_intensity_exp,
luminous_intensity_exp_den, mass_exp, mass_exp_den, maximum,
measurement_begin, measurement_end, minimum, molar_amount_exp,
molar_amount_exp_den, number_of_rows, offset, ordinal_number,
parameter_datatype, rank, raw_datatype, standard_deviation,
start_offset, value_offset, instance_id, level

Removed 'nillable' from application_element contained in application
_model as at least name and id must appear as attribute contents.

Changed minOccurs of application_attribute from 0 to 2.

Changed type of 'value'-component of enumeration_item from
A_ASCIISTRING to A_INT32

August 2005 - Rainer Bartz rbartz@online.de

Removed element 'locale'

Corrected typographical errors in simpleType 'DS_ID', element
'inst_attr_asciistring'.

Re-Sorted some lists (mostly to alphabetical order)

Added/changed the type definitions of the following base attributes:

alias_names, application_model_type, application_model_version,
base_model_version, component_length, datatype, date, dimension,
external_references, meaning_of_aliases, ordinal_number,
parameter_datatype, raw_datatype, superuser_flag, timezone, value_type,
instance_id

Removed type definitions for the following base attributes:

component_datatype, value, quantity_class_enum

Changed complexType 'component' with regard to component_datatype
(now value_type)

Added the following items in the 'base_attributetype_enum':

component_length, meaning_of_aliases, raw_datatype, timezone,
appl_element_name

Removed the following items in the 'base_attributetype_enum':

appl_element_id, component_datatype, datatype_enum,
length_in_bytes, typespec_enum, value_offsets, value,
values_float64

Removed definition of complexType 'values' as this is a choice of only one element 'values_float64'; replaced type reference to 'values' by 'values_float64' in complexType 'real_and_unit' and 'value_with_unit'. This measure avoids a conflict with the base relation 'values' (which is missing for the moment).

Added the following items in the 'base_relation_enum':

parent, parent_sequence_part, scales, values

Removed the following items in the 'base_relation_enum':

alias_names, datatype, false, meaning_of_aliases,
quantity_instance, true, unit_instance

Added the following items in the 'elemtype_enum':

AoAny, AoTestDevice, InitialRightsAt, SecurityLevel

Renamed type 'component_datatype_enum' to 'typespec_enum'. Added the following items:

DT_BYTESTR, DT_BLOB, A_BOOLEAN_BEO, A_BYTE_BEO, A_ASCIISTRING_BEO,
DT_BYTESTR_BEO, DT_BLOB_BEO

Removed the following items from 'typespec_enum':

A_BITFIELD, A_BYTEFIELD, A_UNIT16, A_UINT32, DT_ENUM

Removed type definition for 'parameter_datatype_enum'

Added/changed the following complex types to the list of abstract declarations for base relations:

environment, external_component, local_column, name_mapping,
parent, parent_equipment, parent_equipment_part,
parent_sequence_part, parent_unt_under_test_part, quantities,
quantity, scales, submatrix, successors, units, values

Removed the following complex types from the list of abstract

declarations for base relations:

alias_names, datatype, meaning_of_aliases, quantity_instance,
unit_instance

Added the following complex types to the list of abstract

declarations for base elements:

AoTestDevice, InitialRightsAt, SecurityLevel

Removed type definitions for the following types as they were only
introduced for support of XPointer (which is not used):

t_id, t_ref, and t_ref_list

Removed type definitions for 'posLongInt' as it is assumed that this
type will not be needed.

Changed the element 'documentation': as XML files will not only
occur as a product of an export process, 'creation' seems to be
the better naming here.

Changed AoUnit into AoQuantity in types 'real_with_unit' and
'value_with_unit'

Changed base for the type definition 'inst_attr_complex32_seq'
to DS_COMPLEX32.

15 May 2004 - Mark Quinsland mark.quinsland@highqsoft.com

Added base attribute types to the Base Attribute Type enumeration:

alias_names, application_model_type, application_model_version,
base_model_version, datatype, date, entity_name,
external_references, max_test_level, objecttype,
quantity_classification, values

Added base attribute types to the Base Relation Type enumeration:

external_component, local_column, name_mapping, parameters,
parameter_set, parent_equipment, parent_equipment_part,
parent_unit_under_test_part

Corrected typographical error in the Base Attribute Type
enumeration:

length_in_bytes

Updated version to 1.1

7 December 2003 - Mark Quinsland mark.quinsland@highqsoft.com

Restored ODS Data Type Enumerations.

Note: The datatype enumerations are not to be confused with the name
of the datatypes. They are merely simple names for integers
that indicate data types. For example, the datatype

enumeration DT_DOUBLE is merely shorthand for the integer 7 which further signifies that the datatype is T_DOUBLE / A_FLOAT64. While T_DOUBLE datatypes may still be used within the ODS API, they may not be used in ODS ATF/XML and should be replaced by A_FLOAT64.

14 October 2003 - Mark Quinsland mark.quinsland@highqsoft.com

Added several data types from ASAM Data Type Harmonization project.

A_UINT8 8 bit unsigned integer

A_UINT16 16 bit unsigned integer

A_UINT32 32 bit unsigned integer

A_UINT64 64 bit unsigned integer

A_ASCIIIFIELD Sequence of up to 2048 bits

A_BITFIELD Sequence of up to 2048 bits

A_BCD Decimal digits represented by 4 binary digits

A_ENUM value used for the description of enumeration values

Corrected definition of A_BYTEFIELD to separate length from values

Removed Deprecated ODS Data Types replaced by ASAM data types

Deprecated Replaced By

T_BYTE A_INT8

T_SHORT A_INT16

T_LONG A_INT32

T_LONGLONG A_INT64

T_FLOAT A_FLOAT32

T_DOUBLE A_FLOAT64

T_COMPLEX A_COMPLEX32

T_DCOMPLEX A_COMPLEX64

T_BOOLEAN A_BOOLEAN

T_STRING A_ASCIISTRING

8 August 2003 - Mark Quinsland mark.quinsland@highqsoft.com

Added attribute for schema version.

Added locale element.

ODS data types replaced by harmonized ASAM datatypes labeled as deprecated.

Added t_id and t_ref types to allow xPointer enforcement of relationships between elements.

Added inverse_name to the relation_attribute complex type.

t_blob deprecated - replaced by complex type ct_blob

15 June 2003 - Mark Quinsland mark.quinsland@highqsoft.com

Added element definitions for instance attributes.

Added element definitions for application enumerations

Removed empty definition for instance data

15 April 2003 - Mark Quinsland mark.quinsland@highqsoft.com

Adapted to comply with ASAM Data Type Harmonization project.

Deprecated New

T_BYTE A_INT8

T_SHORT A_INT16

T_LONG A_INT32

T_LONGLONG A_INT64

T_FLOAT A_FLOAT32

T_DOUBLE A_FLOAT64

T_COMPLEX A_COMPLEX32

T_DCOMPLEX A_COMPLEX64

T_BOOLEAN A_BOOLEAN

T_STRING A_ASCIISTRING

20 March 2003 - Mark Quinsland mark.quinsland@highqsoft.com

Initial Version

</xsd:documentation>

</xsd:annotation>

<!--*****

*

* C O M P O N E N T S F O R A P P L I C A T I O N M O D E L S

*

* declare root element & type

-->

<xsd:elementname="atfx_file">

<xsd:complexType>

<xsd:sequence>

<xsd:elementref="documentation"minOccurs="0"/>

<xsd:elementname="base_model_version"

type="hdt:A_UTF8STRING"/>

<xsd:elementref="files"minOccurs="0"/>

```

<xsd:elementref="application_model"/>
<xsd:elementref="instance_data"minOccurs="0"/>
<!--reserve area for instance elements which are not validated by
this schema
-->
</xsd:sequence>
<xsd:attributename="version"type="hdt:A_UTF8STRING"
use="required"/>
</xsd:complexType>
</xsd:element>
<!--*****
****
* declare documentation components
*****
-->
<xsd:elementname="documentation">
<xsd:complexType>
<xsd:sequence>
<xsd:elementname="exported_by"type="hdt:A_UTF8STRING"
minOccurs="0"/>
<xsd:elementname="exporter"type="hdt:A_UTF8STRING"
minOccurs="0"/>
<xsd:elementname="export_date_time"type="hdt:A_UTF8STRING"
minOccurs="0"/>
<xsd:elementname="exporter_version"type="hdt:A_UTF8STRING"
minOccurs="0"/>
<xsd:elementname="short_description"type="hdt:A_UTF8STRING"
minOccurs="0"/>
<xsd:elementname="long_description"type="hdt:A_UTF8STRING"
minOccurs="0"/>
<xsd:elementname="escape_special_chars"type="hdt:A_BOOLEAN"
minOccurs="0"/>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
<!--*****
****
* declare external file components

```

```

*****
-->
<xsd:elementname="files">
<xsd:complexType>
<xsd:sequence>
<xsd:elementname="component"type="filecomponent"
nillable="true"maxOccurs="unbounded"/>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
<!--*****
* declare model meta-data components
*****
-->
<xsd:elementname="application_model">
<xsd:complexType>
<xsd:sequence>
<xsd:elementname="application_enumeration"
type="application_enumeration"
minOccurs="0"maxOccurs="unbounded"/>
<xsd:elementname="application_element"
type="application_element"
minOccurs="1"maxOccurs="unbounded"/>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
<!--*****
* Define structure for application element definitions
*****
-->
<xsd:complexTypename="application_element">
<xsd:sequence>
<xsd:elementname="name"type="hdt:A_UTF8STRING"/>
<xsd:elementname="basetype"type="elemtype_enum"/>
<xsd:elementname="inherits_from"type="hdt:A_UTF8STRING"
minOccurs="0"/>
<xsd:sequence minOccurs="0" maxOccurs="unbounded">

```

```

<xsd:elementname="application_attribute"
type="application_attribute"
minOccurs="0"maxOccurs="unbounded"/>
<xsd:elementname="relation_attribute"type="application_relation"
minOccurs="0" maxOccurs="unbounded"/>
</xsd:sequence>
</xsd:sequence>
</xsd:complexType>
<!--*****
* Define structure for application attribute definitions
*****-->
<xsd:complexTypeName="application_attribute">
<xsd:sequence>
<xsd:elementname="name"type="hdt:A_UTF8STRING"/>
<xsd:elementname="base_attribute"type="base_attribute_enum"
minOccurs="0"/>
<xsd:elementname="datatype"type="datatype_enum"
minOccurs="0"/>
<xsd:elementname="enumeration_type"type="hdt:A_UTF8STRING"
minOccurs="0"/>
<xsd:elementname="autogenerate"type="hdt:A_BOOLEAN"
minOccurs="0"/>
<xsd:elementname="obligatory"type="hdt:A_BOOLEAN"minOccurs="0"/>
<xsd:elementname="unique"type="hdt:A_BOOLEAN"minOccurs="0"/>
<xsd:elementname="length"type="hdt:A_UINT32"minOccurs="0"/>
</xsd:sequence>
</xsd:complexType>
<!--*****
* Define structure for application relation definitions
*****-->
<xsd:complexTypeName="application_relation">
<xsd:sequence>
<xsd:elementname="name"type="hdt:A_UTF8STRING"/>
<xsd:elementname="ref_to"type="hdt:A_UTF8STRING"/>
<xsd:elementname="base_relation"type="base_relation_enum"

```

```

minOccurs="0"/>
<xsd:elementname="min_occurs" type="hdt:A_UINT32"
minOccurs="0"/>
<xsd:elementname="max_occurs" type="multiplicity" minOccurs="0"/>
<xsd:elementname="inverse_name" type="hdt:A_UTF8STRING"
minOccurs="0"/>
</xsd:sequence>
</xsd:complexType>
<!--*****
* User-Defined Application Enumeration Objects
*****
-->
<xsd:complexTypename="application_enumeration">
<xsd:sequence>
<xsd:elementname="name" type="hdt:A_UTF8STRING"/>
<xsd:elementname="item" type="enumeration_item" nillable="true"
maxOccurs="unbounded"/>
</xsd:sequence>
</xsd:complexType>
<xsd:complexTypename="enumeration_item">
<xsd:sequence>
<xsd:elementname="name" type="hdt:A_UTF8STRING"/>
<xsd:elementname="value" type="hdt:A_UINT32"/>
</xsd:sequence>
</xsd:complexType>
<!--*****
* declare instance data components
*****
-->
<xsd:elementname="instance_data">
<xsd:complexType>
<xsd:sequence>
<xsd:anyprocessContents="skip"
minOccurs="1" maxOccurs="unbounded"/>
</xsd:sequence>
</xsd:complexType>
</xsd:element>

```



```

<!--*****
* local type definitions
*****
-->
<xsd:complexTypename="filecomponent">
<xsd:sequence>
<xsd:elementname="identifier"type="hdt:A_UTF8STRING"/>
<xsd:elementname="filename"type="hdt:A_UTF8STRING"/>
</xsd:sequence>
</xsd:complexType>
<xsd:simpleType>
<xsd:restrictionbase="hdt:A_UTF8STRING">
<xsd:enumerationvalue="Many"/>
</xsd:restriction>
</xsd:simpleType>
<xsd:simpleType>
<xsd:union memberTypes="hdt:A_UINT32 many"/>
</xsd:simpleType>
<xsd:simpleType>
<xsd:restrictionbase="hdt:A_UTF8STRING"/>
</xsd:simpleType>
<!--*****
* declare datatype enumeration;
* corresponding numbers increment continuously starting at 0
*****
-->
<xsd:simpleType>
<xsd:restrictionbase="hdt:A_UTF8STRING">
<xsd:enumerationvalue="DT_UNKNOWN"/>
<xsd:enumerationvalue="DT_STRING"/>
<xsd:enumerationvalue="DT_SHORT"/>
<xsd:enumerationvalue="DT_FLOAT"/>
<xsd:enumerationvalue="DT_BOOLEAN"/>
<xsd:enumerationvalue="DT_BYTE"/>
<xsd:enumerationvalue="DT_LONG"/>
<xsd:enumerationvalue="DT_DOUBLE"/>
<xsd:enumerationvalue="DT_LONGLONG"/>

```

```

<xsd:enumerationvalue="DT_ID"/>
<xsd:enumerationvalue="DT_DATE"/>
<xsd:enumerationvalue="DT_BYTESTR"/>
<xsd:enumerationvalue="DT_BLOB"/>
<xsd:enumerationvalue="DT_COMPLEX"/>
<xsd:enumerationvalue="DT_DCOMPLEX"/>
<xsd:enumerationvalue="DS_STRING"/>
<xsd:enumerationvalue="DS_SHORT"/>
<xsd:enumerationvalue="DS_FLOAT"/>
<xsd:enumerationvalue="DS_BOOLEAN"/>
<xsd:enumerationvalue="DS_BYTE"/>
<xsd:enumerationvalue="DS_LONG"/>
<xsd:enumerationvalue="DS_DOUBLE"/>
<xsd:enumerationvalue="DS_LONGLONG"/>
<xsd:enumerationvalue="DS_COMPLEX"/>
<xsd:enumerationvalue="DS_DCOMPLEX"/>
<xsd:enumerationvalue="DS_ID"/>
<xsd:enumerationvalue="DS_DATE"/>
<xsd:enumerationvalue="DS_BYTESTR"/>
<xsd:enumerationvalue="DT_EXTERNALREFERENCE"/>
<xsd:enumerationvalue="DS_EXTERNALREFERENCE"/>
<xsd:enumerationvalue="DT_ENUM"/>
<xsd:enumerationvalue="DS_ENUM"/>
</xsd:restriction>
</xsd:simpleType>
<!--*****
* further enumerations
*****-->
<xsd:simpleTypename="base_attribute_enum">
<xsd:restrictionbase="hdt:A_UTF8STRING">
<xsd:enumerationvalue="alias_names"/>
<xsd:enumerationvalue="application_model_type"/>
<xsd:enumerationvalue="application_model_version"/>
<xsd:enumerationvalue="attribute_name"/>
<xsd:enumerationvalue="average"/>
<xsd:enumerationvalue="base_model_version"/>

```

```
<xsd:enumerationvalue="block_size"/>
<xsd:enumerationvalue="component_length"/>
<xsd:enumerationvalue="current_exp"/>
<xsd:enumerationvalue="current_exp_den"/>
<xsd:enumerationvalue="datatype"/>
<xsd:enumerationvalue="date"/>
<xsd:enumerationvalue="default_datatype"/>
<xsd:enumerationvalue="default_dimension"/>
<xsd:enumerationvalue="default_mq_name"/>
<xsd:enumerationvalue="default_rank"/>
<xsd:enumerationvalue="default_type_size"/>
<xsd:enumerationvalue="description"/>
<xsd:enumerationvalue="dimension"/>
<xsd:enumerationvalue="entity_name"/>
<xsd:enumerationvalue="external_references"/>
<xsd:enumerationvalue="factor"/>
<xsd:enumerationvalue="filename_url"/>
<xsd:enumerationvalue="flags"/>
<xsd:enumerationvalue="flags_filename_url"/>
<xsd:enumerationvalue="flags_start_offset"/>
<xsd:enumerationvalue="generation_parameters"/>
<xsd:enumerationvalue="global_flag"/>
<xsd:enumerationvalue="id"/>
<xsd:enumerationvalue="independent"/>
<xsd:enumerationvalue="interpolation"/>
<xsd:enumerationvalue="length_exp"/>
<xsd:enumerationvalue="length_exp_den"/>
<xsd:enumerationvalue="luminous_intensity_exp"/>
<xsd:enumerationvalue="luminous_intensity_exp_den"/>
<xsd:enumerationvalue="mass_exp"/>
<xsd:enumerationvalue="mass_exp_den"/>
<xsd:enumerationvalue="max_test_level"/>
<xsd:enumerationvalue="maximum"/>
<xsd:enumerationvalue="meaning_of_aliases"/>
<xsd:enumerationvalue="measurement_begin"/>
<xsd:enumerationvalue="measurement_end"/>
<xsd:enumerationvalue="mime_type"/>
```

```

<xsd:enumerationvalue="minimum"/>
<xsd:enumerationvalue="molar_amount_exp"/>
<xsd:enumerationvalue="molar_amount_exp_den"/>
<xsd:enumerationvalue="name"/>
<xsd:enumerationvalue="number_of_rows"/>
<xsd:enumerationvalue="objecttype"/>
<xsd:enumerationvalue="offset"/>
<xsd:enumerationvalue="ordinal_number"/>
<xsd:enumerationvalue="parameter_datatype"/>
<xsd:enumerationvalue="password"/>
<xsd:enumerationvalue="pvalue"/>
<xsd:enumerationvalue="rank"/>
<xsd:enumerationvalue="raw_datatype"/>
<xsd:enumerationvalue="sequence_representation"/>
<xsd:enumerationvalue="standard_deviation"/>
<xsd:enumerationvalue="start_offset"/>
<xsd:enumerationvalue="superuser_flag"/>
<xsd:enumerationvalue="temperature_exp"/>
<xsd:enumerationvalue="temperature_exp_den"/>
<xsd:enumerationvalue="time_exp"/>
<xsd:enumerationvalue="time_exp_den"/>
<xsd:enumerationvalue="timezone"/>
<xsd:enumerationvalue="type_size"/>
<xsd:enumerationvalue="value_offset"/>
<xsd:enumerationvalue="values"/>
<xsd:enumerationvalue="valuesperblock"/>
<xsd:enumerationvalue="value_type"/>
<xsd:enumerationvalue="version"/>
<xsd:enumerationvalue="version_date"/>
</xsd:restriction>
</xsd:simpleType>
<xsd:simpleTypeName="base_relation_enum">
<xsd:restrictionbase="hdt:A_UTF8STRING">
<xsd:enumerationvalue="attribute_mapping"/>
<xsd:enumerationvalue="channel"/>
<xsd:enumerationvalue="children"/>
<xsd:enumerationvalue="default_unit"/>

```

```
<xsd:enumerationvalue="entity_mapping"/>
<xsd:enumerationvalue="environment"/>
<xsd:enumerationvalue="equipments"/>
<xsd:enumerationvalue="external_component"/>
<xsd:enumerationvalue="groups"/>
<xsd:enumerationvalue="is_scaled_by"/>
<xsd:enumerationvalue="local_column"/>
<xsd:enumerationvalue="local_columns"/>
<xsd:enumerationvalue="measurement"/>
<xsd:enumerationvalue="measurement_quantities"/>
<xsd:enumerationvalue="measurement_quantity"/>
<xsd:enumerationvalue="name_mapping"/>
<xsd:enumerationvalue="parameters"/>
<xsd:enumerationvalue="parameter_set"/>
<xsd:enumerationvalue="parent"/>
<xsd:enumerationvalue="parent_equipment"/>
<xsd:enumerationvalue="parent_equipment_part"/>
<xsd:enumerationvalue="parent_sequence"/>
<xsd:enumerationvalue="parent_sequence_part"/>
<xsd:enumerationvalue="parent_test"/>
<xsd:enumerationvalue="parent_unit_under_test"/>
<xsd:enumerationvalue="parent_unit_under_test_part"/>
<xsd:enumerationvalue="phys_dimension"/>
<xsd:enumerationvalue="predecessor"/>
<xsd:enumerationvalue="quantities"/>
<xsd:enumerationvalue="quantity"/>
<xsd:enumerationvalue="scales"/>
<xsd:enumerationvalue="sequences"/>
<xsd:enumerationvalue="submatrices"/>
<xsd:enumerationvalue="submatrix"/>
<xsd:enumerationvalue="successors"/>
<xsd:enumerationvalue="test"/>
<xsd:enumerationvalue="tests"/>
<xsd:enumerationvalue="unit"/>
<xsd:enumerationvalue="units"/>
<xsd:enumerationvalue="units_under_test"/>
<xsd:enumerationvalue="users"/>
```

```
<xsd:enumerationvalue="uuts"/>
</xsd:restriction>
</xsd:simpleType>
<xsd:simpleTypeName="elementype_enum">
<xsd:restrictionbase="hdt:A_UTF8STRING">
<xsd:enumerationvalue="AoAny"/>
<xsd:enumerationvalue="AoAttributeMap"/>
<xsd:enumerationvalue="AoEnvironment"/>
<xsd:enumerationvalue="AoExternalComponent"/>
<xsd:enumerationvalue="AoLocalColumn"/>
<xsd:enumerationvalue="AoLog"/>
<xsd:enumerationvalue="AoMeasurement"/>
<xsd:enumerationvalue="AoMeasurementQuantity"/>
<xsd:enumerationvalue="AoNameMap"/>
<xsd:enumerationvalue="AoParameter"/>
<xsd:enumerationvalue="AoParameterSet"/>
<xsd:enumerationvalue="AoPhysicalDimension"/>
<xsd:enumerationvalue="AoQuantity"/>
<xsd:enumerationvalue="AoQuantityGroup"/>
<xsd:enumerationvalue="AoSubmatrix"/>
<xsd:enumerationvalue="AoSubTest"/>
<xsd:enumerationvalue="AoTest"/>
<xsd:enumerationvalue="AoTestDevice"/>
<xsd:enumerationvalue="AoTestEquipment"/>
<xsd:enumerationvalue="AoTestEquipmentPart"/>
<xsd:enumerationvalue="AoTestSequence"/>
<xsd:enumerationvalue="AoTestSequencePart"/>
<xsd:enumerationvalue="AoUnitUnderTest"/>
<xsd:enumerationvalue="AoUnitUnderTestPart"/>
<xsd:enumerationvalue="AoUnit"/>
<xsd:enumerationvalue="AoUnitGroup"/>
<xsd:enumerationvalue="AoUser"/>
<xsd:enumerationvalue="AoUserGroup"/>
</xsd:restriction>
</xsd:simpleType>
</xsd:schema>
```

10.5.3 ASAM ODS 辅助模式文件

指定用来创建特定应用模式的辅助类型的 XML 模式可见下文。其也可使用纯文本文件格式，也可以从 ASAM 网站上下载得到。

如果节内容与独立可用的模式文件之间有不同，则独立的模式文件是其引用。

```
<?xml version="1.0"?>
<xsd:schematargetNamespace="http://www.asam.net/ODS/5.2.0/HelperSchema
"
xmlns="http://www.asam.net/ODS/5.2.0/HelperSchema"
xmlns:ods="http://www.asam.net/ODS/5.2.0/Schema"
xmlns:hdt="http://www.asam.net/AIS/2005/HDTypes"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
<!--The ASAM harmonized datatypes schema must be placed in the
working directory at the moment -->
<xsd:importnamespace="http://www.asam.net/AIS/2005/HDTypes"
schemaLocation="ASAM_HDTypes.xsd"/>
<!--The ASAM ASAM ODS schema must be placed in the working directory
at the moment -->
<xsd:importnamespace="http://www.asam.net/ODS/5.2.0/Schema"
schemaLocation="Schema.xsd"/>
<xsd:annotation>
<xsd:documentation>
ASAM-ODS Base Model Architecture
W3C Schema Version 1.0
Copyright 2009 - Association for Standardization of Automation
and Measuring Systems.
All rights reserved.
Revision History:
September 2009 - Rainer Bartz rbartz@online.de
Moved references to datatype enumeration as this is no longer part
of ASAM harmonized data types.
Changed data type of inioffset and flags_start_offset to A_INT64.
November 2008 - Rainer Bartz rbartz@online.de
Added datatypes dt_sbyte, dt_ushort, dt_ushort_beo, dt_ulong, and
dt_ulong_beo to typespec_enum (for component files)
February 2006 - Rainer Bartz rbartz@online.de
Replaced A_ASCIISTRING by A_UTF8STRING as ODS will herewith prepare
```

supporting UTF-8 character set.

Changes type of 'datatype' element within component from

'component_datatype' to 'typespec_enum'.

Removed type 'component_datatype'.

Changed all entries of 'typespec_enum' into lowercase.

December 2005 - Rainer Bartz rbartz@online.de

Removed component_datatype_enum due to email voting and redirected references to typespec_enum (with _FLAGS_ removed) .

October 2005 - Rainer Bartz rbartz@online.de

Created this schema by extracting part 2 from the combined schema

Removed complex type S_ASCIISTRING and defined complex types alias_names and meaning_of_aliases individually due to schema notation restrictions

Re-installed component_datatype and component_datatype_enum

typespec_enum has been modified to match the base model definition

Added all items of typespec_enum to component_datatype_enum

Changed er to external_reference in type external_references

</xsd:documentation>

</xsd:annotation>

<!--*****

*

* H E L P E R T Y P E S F O R A P P L I C A T I O N S C H E M A

*

-->

<xsd:simpleTypeName="values_float64">

<xsd:listitemType="hdt:A_FLOAT64"/>

</xsd:simpleType>

<xsd:simpleTypeName="values_float32">

<xsd:listitemType="hdt:A_FLOAT32"/>

</xsd:simpleType>

<xsd:simpleTypeName="values_date">

<xsd:listitemType="ods:T_DATE"/>

</xsd:simpleType>

<xsd:simpleTypeName="values_int32">

<xsd:listitemType="hdt:A_INT32"/>

</xsd:simpleType>


```

<xsd:simpleTypeName="values_int64">
<xsd:listitemType="hdt:A_INT64"/>
</xsd:simpleType>
<xsd:simpleTypeName="values_int16">
<xsd:listitemType="hdt:A_INT16"/>
</xsd:simpleType>
<xsd:simpleTypeName="values_character">
<xsd:listitemType="hdt:A_ASCII"/>
</xsd:simpleType>
<!--*****
* Subtypes
*****
-->
<xsd:simpleTypeName="asciistring_attribute">
<xsd:listitemType="hdt:A_UTF8STRING"/>
</xsd:simpleType>
<xsd:simpleTypeName="float32_attribute">
<xsd:listitemType="hdt:A_FLOAT32"/>
</xsd:simpleType>
<xsd:simpleTypeName="float64_attribute">
<xsd:listitemType="hdt:A_FLOAT64"/>
</xsd:simpleType>
<xsd:simpleTypeName="int8_attribute">
<xsd:listitemType="hdt:A_INT8"/>
</xsd:simpleType>
<xsd:simpleTypeName="int16_attribute">
<xsd:listitemType="hdt:A_INT16"/>
</xsd:simpleType>
<xsd:simpleTypeName="int32_attribute">
<xsd:listitemType="hdt:A_INT32"/>
</xsd:simpleType>
<xsd:simpleTypeName="time_attribute">
<xsd:listitemType="ods:T_DATE"/>
</xsd:simpleType>
<!--*****
* Sequence Subtypes
*****

```

```

-->
<xsd:simpleTypeName="S_BOOLEAN">
<xsd:listitemType="hdt:A_BOOLEAN"/>
</xsd:simpleType>
<xsd:simpleTypeName="S_BYTESTR">
<xsd:listitemType="hdt:A_INT8"/>
</xsd:simpleType>
<xsd:simpleTypeName="S_COMPLEX32">
<xsd:listitemType="hdt:A_COMPLEX32"/>
</xsd:simpleType>
<xsd:simpleTypeName="S_COMPLEX64">
<xsd:listitemType="hdt:A_COMPLEX64"/>
</xsd:simpleType>
<xsd:simpleTypeName="S_DATE">
<xsd:list itemType="ods:T_DATE"/>
</xsd:simpleType>
<xsd:simpleTypeName="S_FLOAT32">
<xsd:listitemType="hdt:A_FLOAT32"/>
</xsd:simpleType>
<xsd:simpleTypeName="S_FLOAT64">
<xsd:listitemType="hdt:A_FLOAT64"/>
</xsd:simpleType>
<xsd:simpleTypeName="S_INT8">
<xsd:listitemType="hdt:A_INT8"/>
</xsd:simpleType>
<xsd:simpleTypeName="S_INT16">
<xsd:listitemType="hdt:A_INT16"/>
</xsd:simpleType>
<xsd:simpleTypeName="S_INT32">
<xsd:listitemType="hdt:A_INT32"/>
</xsd:simpleType>
<xsd:simpleTypeName="S_INT64">
<xsd:listitemType="hdt:A_INT64"/>
</xsd:simpleType>
<!--*****
* Further subtypes
*****

```

```

-->
<!-- Simple Base Attribute Types -->
<xsd:simpleTypeName="asam_path">
<xsd:restrictionbase="hdt:A_UTF8STRING"/>
</xsd:simpleType>
<xsd:simpleTypeName="attribute_type">
<xsd:restrictionbase="hdt:A_UTF8STRING"/>
</xsd:simpleType>
<xsd:simpleTypeName="base_element_name">
<xsd:restrictionbase="hdt:A_UTF8STRING"/>
</xsd:simpleType>
<xsd:simpleTypeName="integerSequence">
<xsd:listitemType="hdt:A_INT32"/>
</xsd:simpleType>
<xsd:simpleTypeName="rule_text">
<xsd:restrictionbase="hdt:A_UTF8STRING"/>
</xsd:simpleType>
<xsd:simpleTypeName="reference">
<xsd:union memberTypes="hdt:A_UINT64 asam_path"/>
</xsd:simpleType>
<!--*****
* Complex types for instance attributes
*****-->
<xsd:complexType="instance_attributes">
<xsd:choicemaxOccurs="unbounded">
<xsd:elementname="inst_attr_asciistring"
type="inst_attr_asciistring"/>
<xsd:elementname="inst_attr_float32" type="inst_attr_float32"/>
<xsd:elementname="inst_attr_float64" type="inst_attr_float64"/>
<xsd:elementname="inst_attr_int8" type="inst_attr_int8"/>
<xsd:elementname="inst_attr_int16" type="inst_attr_int16"/>
<xsd:elementname="inst_attr_int32" type="inst_attr_int32"/>
<xsd:elementname="inst_attr_int64" type="inst_attr_int64"/>
<xsd:elementname="inst_attr_time" type="inst_attr_time"/>
</xsd:choice>
</xsd:complexType>

```

```

<xsd:complexTypeName="inst_attr_asciistring">
<xsd:simpleContent>
<xsd:extensionbase="hdt:A_UTF8STRING">
<xsd:attributename="name"type="hdt:A_UTF8STRING"
use="required"/>
<xsd:attributename="unit"type="reference"use="optional"/>
</xsd:extension>
</xsd:simpleContent>
</xsd:complexType>
<xsd:complexTypeName="inst_attr_float32">
<xsd:simpleContent>
<xsd:extensionbase="hdt:A_FLOAT32">
<xsd:attributename="name"type="hdt:A_UTF8STRING"
use="required"/>
<xsd:attributename="unit"type="reference"use="optional"/>
</xsd:extension>
</xsd:simpleContent>
</xsd:complexType>
<xsd:complexTypeName="inst_attr_float64">
<xsd:simpleContent>
<xsd:extensionbase="hdt:A_FLOAT64">
<xsd:attributename="name"type="hdt:A_UTF8STRING"
use="required"/>
<xsd:attributename="unit"type="reference"use="optional"/>
</xsd:extension>
</xsd:simpleContent>
</xsd:complexType>
<xsd:complexTypeName="inst_attr_int8">
<xsd:simpleContent>
<xsd:extensionbase="hdt:A_INT8">
<xsd:attributename="name"type="hdt:A_UTF8STRING"
use="required"/>
<xsd:attributename="unit"type="reference"use="optional"/>
</xsd:extension>
</xsd:simpleContent>
</xsd:complexType>
<xsd:complexTypeName="inst_attr_int16">

```

```

<xsd:simpleContent>
<xsd:extensionbase="hdt:A_INT16">
<xsd:attributename="name"type="hdt:A_UTF8STRING"
use="required"/>
<xsd:attributename="unit"type="reference"use="optional"/>
</xsd:extension>
</xsd:simpleContent>
</xsd:complexType>
<xsd:complexTypename="inst_attr_int32">
<xsd:simpleContent>
<xsd:extensionbase="hdt:A_INT32">
<xsd:attributename="name"type="hdt:A_UTF8STRING"
use="required"/>
<xsd:attributename="unit"type="reference"use="optional"/>
</xsd:extension>
</xsd:simpleContent>
</xsd:complexType>
<xsd:complexTypename="inst_attr_int64">
<xsd:simpleContent>
<xsd:extensionbase="hdt:A_INT64">
<xsd:attributename="name"type="hdt:A_UTF8STRING"
use="required"/>
<xsd:attributename="unit"type="reference"use="optional"/>
</xsd:extension>
</xsd:simpleContent>
</xsd:complexType>
<xsd:complexTypename="inst_attr_time">
<xsd:simpleContent>
<xsd:extensionbase="ods:T_DATE">
<xsd:attributename="name"type="hdt:A_UTF8STRING"
use="required"/>
<xsd:attributename="unit"type="reference"use="optional"/>
</xsd:extension>
</xsd:simpleContent>
</xsd:complexType>
<!--*****
* Complex type component

```

```

*****
-->
<xsd:complexTypename="component">
<xsd:sequence>
<xsd:elementname="identifier"type="hdt:A_UTF8STRING"/>
<xsd:elementname="datatype"type="typespec_enum"/>
<xsd:elementname="length"type="hdt:A_INT32"/>
<xsd:elementname="description"type="hdt:A_UTF8STRING"
minOccurs="0"/>
<xsd:elementname="inioffset"type="hdt:A_INT64"minOccurs="0"/>
<xsd:elementname="blocksize"type="hdt:A_INT32"minOccurs="0"/>
<xsd:elementname="valperblock"type="hdt:A_INT32"minOccurs="0"/>
<xsd:elementname="valoffsets"type="S_INT32"minOccurs="0"/>
</xsd:sequence>
</xsd:complexType>
<xsd:complexTypename="T_BLOB">
<xsd:sequence>
<xsd:elementname="text"type="hdt:A_UTF8STRING"/>
<xsd:elementname="bytefield"type="hdt:A_BYTEFIELD"/>
</xsd:sequence>
</xsd:complexType>
<xsd:complexTypename="T_EXTERNALREFERENCE">
<xsd:sequence>
<xsd:elementname="description"type="hdt:A_UTF8STRING"/>
<xsd:elementname="mimetype"type="hdt:A_UTF8STRING"/>
<xsd:elementname="location"type="hdt:A_UTF8STRING"/>
</xsd:sequence>
</xsd:complexType>
<!--*****
* Type Definitions for Base Attributes
*****
-->
<xsd:complexTypename="alias_names">
<xsd:sequenceminOccurs="0"maxOccurs="unbounded">
<xsd:elementname="an"type="hdt:A_UTF8STRING"/>
</xsd:sequence>
</xsd:complexType>

```

```

<xsd:simpleTypeName="application_model_type">
<xsd:restrictionbase="hdt:A_UTF8STRING"/>
</xsd:simpleType>
<xsd:simpleTypeName="application_model_version">
<xsd:restrictionbase="hdt:A_UTF8STRING"/>
</xsd:simpleType>
<xsd:simpleTypeName="attribute_name">
<xsd:restrictionbase="hdt:A_UTF8STRING"/>
</xsd:simpleType>
<xsd:simpleTypeName="average">
<xsd:restrictionbase="hdt:A_FLOAT64"/>
</xsd:simpleType>
<xsd:simpleTypeName="base_model_version">
<xsd:restrictionbase="hdt:A_UTF8STRING"/>
</xsd:simpleType>
<xsd:simpleTypeName="block_size">
<xsd:restrictionbase="hdt:A_INT32"/>
</xsd:simpleType>
<xsd:simpleTypeName="component_length">
<xsd:restrictionbase="hdt:A_INT32"/>
</xsd:simpleType>
<xsd:simpleTypeName="current_exp">
<xsd:restrictionbase="hdt:A_INT32"/>
</xsd:simpleType>
<xsd:simpleTypeName="current_exp_den">
<xsd:restrictionbase="hdt:A_INT32"/>
</xsd:simpleType>
<xsd:simpleTypeName="datatype">
<xsd:restrictionbase="ods:datatype_enum"/>
</xsd:simpleType>
<xsd:simpleTypeName="date">
<xsd:restrictionbase="ods:T_DATE"/>
</xsd:simpleType>
<xsd:simpleTypeName="default_datatype">
<xsd:restrictionbase="ods:datatype_enum"/>
</xsd:simpleType>
<xsd:simpleTypeName="default_dimension">

```

```

<xsd:listitemType="hdt:A_INT32"/>
</xsd:simpleType>
<xsd:simpleTypeName="default_mq_name">
<xsd:restrictionbase="hdt:A_UTF8STRING"/>
</xsd:simpleType>
<xsd:simpleTypeName="default_rank">
<xsd:restrictionbase="hdt:A_INT32"/>
</xsd:simpleType>
<xsd:simpleTypeName="default_type_size">
<xsd:restrictionbase="hdt:A_INT32"/>
</xsd:simpleType>
<xsd:simpleTypeName="description">
<xsd:restrictionbase="hdt:A_UTF8STRING"/>
</xsd:simpleType>
<xsd:simpleTypeName="dimension">
<xsd:listitemType="hdt:A_INT32"/>
</xsd:simpleType>
<xsd:simpleTypeName="entity_name">
<xsd:restrictionbase="hdt:A_UTF8STRING"/>
</xsd:simpleType>
<xsd:complexType="external_references">
<xsd:sequence minOccurs="0" maxOccurs="unbounded">
<xsd:element name="external_reference" type="T_EXTERNALREFERENCE"/>
</xsd:sequence>
</xsd:complexType>
<xsd:simpleTypeName="factor">
<xsd:restrictionbase="hdt:A_FLOAT64"/>
</xsd:simpleType>
<xsd:simpleTypeName="filename_url">
<xsd:restrictionbase="hdt:A_UTF8STRING"/>
</xsd:simpleType>
<xsd:simpleTypeName="flags">
<xsd:restrictionbase="S_INT16">
<xsd:minLength value="1"/>
</xsd:restriction>
</xsd:simpleType>
<xsd:simpleTypeName="flags_filename_url">

```



```

<xsd:restrictionbase="hdt:A_UTF8STRING"/>
</xsd:simpleType>
<xsd:simpleTypename="flags_start_offset">
<xsd:restrictionbase="hdt:A_INT64"/>
</xsd:simpleType>
<xsd:simpleTypename="generation_parameters">
<xsd:restrictionbase="S_FLOAT64">
<xsd:minLengthvalue="1"/>
</xsd:restriction>
</xsd:simpleType>
<xsd:simpleTypename="global_flag">
<xsd:restrictionbase="hdt:A_INT16"/>
</xsd:simpleType>
<xsd:simpleTypename="id">
<xsd:restrictionbase="hdt:A_INT64"/>
</xsd:simpleType>
<xsd:simpleTypename="independent">
<xsd:restrictionbase="hdt:A_BOOLEAN"/>
</xsd:simpleType>
<xsd:simpleTypename="interpolation">
<xsd:restrictionbase="interpolation_enum"/>
</xsd:simpleType>
<xsd:simpleTypename="length_exp">
<xsd:restrictionbase="hdt:A_INT32"/>
</xsd:simpleType>
<xsd:simpleTypename="length_exp_den">
<xsd:restrictionbase="hdt:A_INT32"/>
</xsd:simpleType>
<xsd:simpleTypename="luminous_intensity_exp">
<xsd:restrictionbase="hdt:A_INT32"/>
</xsd:simpleType>
<xsd:simpleTypename="luminous_intensity_exp_den">
<xsd:restrictionbase="hdt:A_INT32"/>
</xsd:simpleType>
<xsd:simpleTypename="mass_exp">
<xsd:restrictionbase="hdt:A_INT32"/>
</xsd:simpleType>

```

```

<xsd:simpleTypeName="mass_exp_den">
<xsd:restrictionbase="hdt:A_INT32"/>
</xsd:simpleType>
<xsd:simpleTypeName="max_test_level">
<xsd:restrictionbase="hdt:A_INT32"/>
</xsd:simpleType>
<xsd:simpleTypeName="maximum">
<xsd:restrictionbase="hdt:A_FLOAT64"/>
</xsd:simpleType>
<xsd:complexType="meaning_of_aliases">
<xsd:sequence minOccurs="0" maxOccurs="unbounded">
<xsd:element name="ma" type="hdt:A_UTF8STRING"/>
</xsd:sequence>
</xsd:complexType>
<xsd:simpleTypeName="measurement_begin">
<xsd:restrictionbase="ods:T_DATE"/>
</xsd:simpleType>
<xsd:simpleTypeName="measurement_end">
<xsd:restrictionbase="ods:T_DATE"/>
</xsd:simpleType>
<xsd:simpleTypeName="mime_type">
<xsd:restrictionbase="hdt:A_UTF8STRING"/>
</xsd:simpleType>
<xsd:simpleTypeName="minimum">
<xsd:restrictionbase="hdt:A_FLOAT64"/>
</xsd:simpleType>
<xsd:simpleTypeName="molar_amount_exp">
<xsd:restrictionbase="hdt:A_INT32"/>
</xsd:simpleType>
<xsd:simpleTypeName="molar_amount_exp_den">
<xsd:restrictionbase="hdt:A_INT32"/>
</xsd:simpleType>
<xsd:simpleTypeName="name">
<xsd:restrictionbase="hdt:A_UTF8STRING"/>
</xsd:simpleType>
<xsd:simpleTypeName="number_of_rows">
<xsd:restrictionbase="hdt:A_INT32"/>

```

```

</xsd:simpleType>
<xsd:simpleTypeName="objecttype">
<xsd:restrictionbase="hdt:A_INT64"/>
</xsd:simpleType>
<xsd:simpleTypeName="offset">
<xsd:restrictionbase="hdt:A_FLOAT64"/>
</xsd:simpleType>
<xsd:simpleTypeName="ordinal_number">
<xsd:restrictionbase="hdt:A_INT32"/>
</xsd:simpleType>
<xsd:simpleTypeName="parameter_datatype">
<xsd:restrictionbase="ods:datatype_enum"/>
</xsd:simpleType>
<xsd:simpleTypeName="password">
<xsd:restrictionbase="hdt:A_UTF8STRING"/>
</xsd:simpleType>
<xsd:simpleTypeName="pvalue">
<xsd:restrictionbase="hdt:A_UTF8STRING"/>
</xsd:simpleType>
<xsd:simpleTypeName="rank">
<xsd:restrictionbase="hdt:A_INT32"/>
</xsd:simpleType>
<xsd:simpleTypeName="raw_datatype">
<xsd:restrictionbase="ods:datatype_enum"/>
</xsd:simpleType>
<xsd:simpleTypeName="sequence_representation">
<xsd:restrictionbase="seq_rep_enum"/>
</xsd:simpleType>
<xsd:simpleTypeName="standard_deviation">
<xsd:restrictionbase="hdt:A_FLOAT64"/>
</xsd:simpleType>
<xsd:simpleTypeName="start_offset">
<xsd:restrictionbase="hdt:A_INT32"/>
</xsd:simpleType>
<xsd:simpleTypeName="superuser_flag">
<xsd:restrictionbase="hdt:A_INT16"/>
</xsd:simpleType>

```

```

<xsd:simpleTypeName="temperature_exp">
<xsd:restrictionbase="hdt:A_INT32"/>
</xsd:simpleType>
<xsd:simpleTypeName="temperature_exp_den">
<xsd:restrictionbase="hdt:A_INT32"/>
</xsd:simpleType>
<xsd:simpleTypeName="time_exp">
<xsd:restrictionbase="hdt:A_INT32"/>
</xsd:simpleType>
<xsd:simpleTypeName="time_exp_den">
<xsd:restrictionbase="hdt:A_INT32"/>
</xsd:simpleType>
<xsd:simpleTypeName="timezone">
<xsd:restrictionbase="hdt:A_UTF8STRING"/>
</xsd:simpleType>
<xsd:simpleTypeName="type_size">
<xsd:restrictionbase="hdt:A_INT32"/>
</xsd:simpleType>
<xsd:simpleTypeName="value_offset">
<xsd:restrictionbase="hdt:A_INT32"/>
</xsd:simpleType>
<xsd:simpleTypeName="valuesperblock">
<xsd:restrictionbase="hdt:A_INT32"/>
</xsd:simpleType>
<xsd:simpleTypeName="value_type">
<xsd:restrictionbase="typespec_enum"/>
</xsd:simpleType>
<xsd:simpleTypeName="version">
<xsd:restrictionbase="hdt:A_UTF8STRING"/>
</xsd:simpleType>
<xsd:simpleTypeName="version_date">
<xsd:restrictionbase="ods:T_DATE"/>
</xsd:simpleType>
<!--*****
* Items of ACLA, ACLI, ACLTemplate, SecurityLevel, InitialRightsAt
* (not ASAM ODS base attributes)
* (attribute_name is the name of a base attribute as well as of an

```

* item; as it is already specified in the list of type definitions

* for base attributes above, it may be omitted here)

-->

<xsd:simpleTypeName="appl_element_name">

<xsd:restrictionbase="hdt:A_UTF8STRING"/>

</xsd:simpleType>

<xsd:simpleTypeName="instance_id">

<xsd:restrictionbase="hdt:A_INT64"/>

</xsd:simpleType>

<xsd:simpleTypeName="level">

<xsd:restrictionbase="hdt:A_INT16"/>

</xsd:simpleType>

<xsd:simpleTypeName="ref_appl_elem_name">

<xsd:restrictionbase="hdt:A_UTF8STRING"/>

</xsd:simpleType>

<xsd:simpleTypeName="rights">

<xsd:restrictionbase="hdt:A_INT16"/>

</xsd:simpleType>

<!--*****

* Enumerations

-->

<xsd:simpleTypeName="typespec_enum">

<xsd:restrictionbase="hdt:A_UTF8STRING">

<xsd:enumerationvalue="dt_boolean"/>

<xsd:enumerationvalue="dt_byte"/>

<xsd:enumerationvalue="dt_short"/>

<xsd:enumerationvalue="dt_long"/>

<xsd:enumerationvalue="dt_longlong"/>

<xsd:enumerationvalue="ieeefloat4"/>

<xsd:enumerationvalue="ieeefloat8"/>

<xsd:enumerationvalue="dt_short_beo"/>

<xsd:enumerationvalue="dt_long_beo"/>

<xsd:enumerationvalue="dt_longlong_beo"/>

<xsd:enumerationvalue="ieeefloat4_beo"/>

<xsd:enumerationvalue="ieeefloat8_beo"/>

```

<xsd:enumerationvalue="dt_string"/>
<xsd:enumerationvalue="dt_bytestr"/>
<xsd:enumerationvalue="dt_blob"/>
<xsd:enumerationvalue="dt_bytestr_beo"/>
<xsd:enumerationvalue="dt_sbyte"/>
<xsd:enumerationvalue="dt_ushort"/>
<xsd:enumerationvalue="dt_ushort_beo"/>
<xsd:enumerationvalue="dt_ulong"/>
<xsd:enumerationvalue="dt_ulong_beo"/>
</xsd:restriction>
</xsd:simpleType>
<xsd:simpleTypeName="interpolation_enum">
<xsd:restrictionbase="hdt:A_UTF8STRING">
<xsd:enumerationvalue="no_interpolation"/>
<xsd:enumerationvalue="linear_interpolation"/>
<xsd:enumerationvalue="application_specific"/>
</xsd:restriction>
</xsd:simpleType>
<xsd:simpleTypeName="seq_rep_enum">
<xsd:restrictionbase="hdt:A_UTF8STRING">
<xsd:enumerationvalue="explicit"/>
<xsd:enumerationvalue="implicit_constant"/>
<xsd:enumerationvalue="implicit_linear"/>
<xsd:enumerationvalue="implicit_saw"/>
<xsd:enumerationvalue="raw_linear"/>
<xsd:enumerationvalue="raw_polynomial"/>
<xsd:enumerationvalue="formula"/>
<xsd:enumerationvalue="external_component"/>
<xsd:enumerationvalue="raw_linear_external"/>
<xsd:enumerationvalue="raw_polynomial_external"/>
<xsd:enumerationvalue="raw_linear_calibrated"/>
<xsd:enumerationvalue="raw_linear_calibrated_external"/>
</xsd:restriction>
</xsd:simpleType>
<!--*****
* Abstract declarations for ASAM Base Relations
*****

```

-->

```

<xsd:complexTypename="attribute_mapping"abstract="true"/>
<xsd:complexTypename="channel"abstract="true"/>
<xsd:complexTypename="children"abstract="true"/>
<xsd:complexTypename="default_unit"abstract="true"/>
<xsd:complexTypename="entity_mapping"abstract="true"/>
<xsd:complexTypename="environment"abstract="true"/>
<xsd:complexTypename="equipments"abstract="true"/>
<xsd:complexTypename="external_component"abstract="true"/>
<xsd:complexTypename="groups"abstract="true"/>
<xsd:complexTypename="is_scaled_by"abstract="true"/>
<xsd:complexTypename="local_column"abstract="true"/>
<xsd:complexTypename="local_columns"abstract="true"/>
<xsd:complexTypename="measurement"abstract="true"/>
<xsd:complexTypename="measurement_quantities"abstract="true"/>
<xsd:complexTypename="measurement_quantity"abstract="true"/>
<xsd:complexTypename="name_mapping"abstract="true"/>
<xsd:complexTypename="parameters"abstract="true"/>
<xsd:complexTypename="parameter_set"abstract="true"/>
<xsd:complexTypename="parent"abstract="true"/>
<xsd:complexTypename="parent_equipment"abstract="true"/>
<xsd:complexTypename="parent_equipment_part"abstract="true"/>
<xsd:complexTypename="parent_sequence"abstract="true"/>
<xsd:complexTypename="parent_sequence_part"abstract="true"/>
<xsd:complexTypename="parent_test"abstract="true"/>
<xsd:complexTypename="parent_unit_under_test"abstract="true"/>
<xsd:complexTypename="parent_unit_under_test_part"abstract="true"/>
<xsd:complexTypename="phys_dimension"abstract="true"/>
<xsd:complexTypename="predecessor"abstract="true"/>
<xsd:complexTypename="quantities"abstract="true"/>
<xsd:complexTypename="quantity"abstract="true"/>
<xsd:complexTypename="scales"abstract="true"/>
<xsd:complexTypename="sequences"abstract="true"/>
<xsd:complexTypename="submatrices"abstract="true"/>
<xsd:complexTypename="submatrix"abstract="true"/>
<xsd:complexTypename="successors"abstract="true"/>
<xsd:complexTypename="test"abstract="true"/>

```

```

<xsd:complexTypename="tests"abstract="true"/>
<xsd:complexTypename="unit"abstract="true"/>
<xsd:complexTypename="units"abstract="true"/>
<xsd:complexTypename="units_under_test"abstract="true"/>
<xsd:complexTypename="users"abstract="true"/>
<xsd:complexTypename="uuts"abstract="true"/>
<xsd:complexTypename="values"abstract="true"/>
<!--*****
* Abstract declarations for ASAM Base Elements
*****
-->
<xsd:complexTypename="AoAny"abstract="true"/>
<xsd:complexTypename="AoAttributeMap"abstract="true"/>
<xsd:complexTypename="AoEnvironment"abstract="true"/>
<xsd:complexTypename="AoExternalComponent"abstract="true"/>
<xsd:complexTypename="AoLocalColumn"abstract="true"/>
<xsd:complexTypename="AoLog"abstract="true"/>
<xsd:complexTypename="AoMeasurement"abstract="true"/>
<xsd:complexTypename="AoMeasurementQuantity"abstract="true"/>
<xsd:complexTypename="AoNameMap"abstract="true"/>
<xsd:complexTypename="AoParameter"abstract="true"/>
<xsd:complexTypename="AoParameterSet"abstract="true"/>
<xsd:complexTypename="AoPhysicalDimension"abstract="true"/>
<xsd:complexTypename="AoQuantity"abstract="true"/>
<xsd:complexTypename="AoQuantityGroup"abstract="true"/>
<xsd:complexTypename="AoSubmatrix"abstract="true"/>
<xsd:complexTypename="AoSubTest"abstract="true"/>
<xsd:complexTypename="AoTest"abstract="true"/>
<xsd:complexTypename="AoTestDevice"abstract="true"/>
<xsd:complexTypename="AoTestEquipment"abstract="true"/>
<xsd:complexTypename="AoTestEquipmentPart"abstract="true"/>
<xsd:complexTypename="AoTestSequence"abstract="true"/>
<xsd:complexTypename="AoTestSequencePart"abstract="true"/>
<xsd:complexTypename="AoUnitUnderTest"abstract="true"/>
<xsd:complexTypename="AoUnitUnderTestPart"abstract="true"/>
<xsd:complexTypename="AoUnit"abstract="true"/>
<xsd:complexTypename="AoUnitGroup"abstract="true"/>

```



```

<xsd:complexTypename="AoUser"abstract="true"/>
<xsd:complexTypename="AoUserGroup"abstract="true"/>
<!--*****
* Abstract declarations for Security - Not ASAM ODS base elements
*****-->
<xsd:complexTypename="ACLA"abstract="true"/>
<xsd:complexTypename="ACLI"abstract="true"/>
<xsd:complexTypename="ACLTemplate"abstract="true"/>
<xsd:complexTypename="InitialRightsAt"abstract="true"/>
<xsd:complexTypename="SecurityLevel"abstract="true"/>
</xsd:schema>

```

10.6 ATF/XML 文件例子

演示 XML 文件的内容如下. 尽管这个文件已经被彻底检查,但是它仍然具有与 ATF/XML 规范的不一致性和错误。如有疑问, 10.2 节给出的 ATF/XML 规范能说明。

```

<?xml version="1.0" encoding="UTF-8"?>
<atfx_file version="atfx_file: V1.2.0"
xmlns="http://www.asam.net/ODS/5.2.0/Schema">
<documentation>
<exported_by></exported_by>
<exporter>uctf</exporter>
<export_date_time>20051205131445</export_date_time>
<exporter_version>10.3.0.2525</exporter_version>
</documentation>
<base_model_version>asam30</base_model_version>
<application_model>
<application_enumeration>
<name>TestResult</name>
<item>
<name>Failed</name>
<value>0</value>
</item>
<item>
<name>Succeeded</name>
<value>1</value>
</item>

```

```

<item>
  <name>NotExecuted</name>
  <value>2</value>
</item>
</application_enumeration>
<application_element>
  <name>Environment</name>
  <basetype>AoEnvironment</basetype>
  <application_attribute>
    <name>Id</name>
    <base_attribute>id</base_attribute>
    <obligatory>true</obligatory>
    <unique>true</unique>
  </application_attribute>
  <application_attribute>
    <name>Name</name>
    <base_attribute>name</base_attribute>
    <obligatory>true</obligatory>
  </application_attribute>
  <relation_attribute>
    <name>Tests</name>
    <ref_to>Test</ref_to>
    <base_relation>tests</base_relation>
    <min_occurs>0</min_occurs>
    <max_occurs>Many</max_occurs>
    <inverse_name>Environment</inverse_name>
  </relation_attribute>
</application_element>
<application_element>
  <name>Test</name>
  <basetype>AoTest</basetype>
  <application_attribute>
    <name>Id</name>
    <base_attribute>id</base_attribute>
    <obligatory>true</obligatory>
    <unique>true</unique>
  </application_attribute>

```

```

<application_attribute>
<name>Name</name>
<base_attribute>name</base_attribute>
<obligatory>true</obligatory>
</application_attribute>
<relation_attribute>
<name>Environment</name>
<ref_to>Environment</ref_to>
<base_relation>environment</base_relation>
<min_occurs>0</min_occurs>
<max_occurs>1</max_occurs>
<inverse_name>Tests</inverse_name>
</relation_attribute>
<relation_attribute>
<name>Subtests</name>
<ref_to>Subtest</ref_to>
<base_relation>children</base_relation>
<min_occurs>0</min_occurs>
<max_occurs>Many</max_occurs>
<inverse_name>Test</inverse_name>
</relation_attribute>
</application_element>
<application_element>
<name>Subtest</name>
<basetype>AoSubTest</basetype>
<application_attribute>
<name>Id</name>
<base_attribute>id</base_attribute>
<obligatory>true</obligatory>
<unique>true</unique>
</application_attribute>
<application_attribute>
<name>Name</name>
<base_attribute>name</base_attribute>
<obligatory>true</obligatory>
</application_attribute>
<relation_attribute>

```

```

<name>Test</name>
<ref_to>Test</ref_to>
<base_relation>parent_test</base_relation>
<min_occurs>1</min_occurs>
<max_occurs>1</max_occurs>
<inverse_name>Subtests</inverse_name>
</relation_attribute>
<relation_attribute>
<name>Measurements</name>
<ref_to>Measurement</ref_to>
<base_relation>children</base_relation>
<min_occurs>0</min_occurs>
<max_occurs>Many</max_occurs>
<inverse_name>Subtest</inverse_name>
</relation_attribute>
<application_attribute>
<name>Result</name>
<datatype>DT_ENUM</datatype>
<enumeration_type>TestResult</enumeration_type>
<obligatory>true</obligatory>
</application_attribute>
</application_element>
<application_element>
<name>Measurement</name>
<basetype>AoMeasurement</basetype>
<application_attribute>
<name>Id</name>
<base_attribute>id</base_attribute>
<obligatory>true</obligatory>
<unique>true</unique>
</application_attribute>
<application_attribute>
<name>Name</name>
<base_attribute>name</base_attribute>
<obligatory>true</obligatory>
</application_attribute>
<application_attribute>

```

```

<name>StartTime</name>
<base_attribute>measurement_begin</base_attribute>
</application_attribute>
<relation_attribute>
<name>Subtest</name>
<ref_to>Subtest</ref_to>
<base_relation>test</base_relation>
<min_occurs>1</min_occurs>
<max_occurs>1</max_occurs>
<inverse_name>Measurements</inverse_name>
</relation_attribute>
<relation_attribute>
<name>MeasurementQuantities</name>
<ref_to>Measurementquantity</ref_to>
<base_relation>measurement_quantities</base_relation>
<min_occurs>0</min_occurs>
<max_occurs>Many</max_occurs>
<inverse_name>Measurement</inverse_name>
</relation_attribute>
<relation_attribute>
<name>Submatrices</name>
<ref_to>Submatrix</ref_to>
<base_relation>submatrices</base_relation>
<min_occurs>0</min_occurs>
<max_occurs>Many</max_occurs>
<inverse_name>Measurement</inverse_name>
</relation_attribute>
<relation_attribute>
<name>Responsible</name>
<ref_to>User</ref_to>
<min_occurs>0</min_occurs>
<max_occurs>1</max_occurs>
</relation_attribute>
<relation_attribute>
<name>Process</name>
<ref_to>Process</ref_to>
<min_occurs>0</min_occurs>

```

```

<max_occurs>1</max_occurs>
</relation_attribute>
</application_element>
<application_element>
<name>Measurementquantity</name>
<basetype>AoMeasurementQuantity</basetype>
<application_attribute>
<name>Id</name>
<base_attribute>id</base_attribute>
<obligatory>true</obligatory>
<unique>true</unique>
</application_attribute>
<application_attribute>
<name>Name</name>
<base_attribute>name</base_attribute>
<obligatory>true</obligatory>
</application_attribute>
<application_attribute>
<name>Datatype</name>
<base_attribute>datatype</base_attribute>
<obligatory>true</obligatory>
</application_attribute>
<relation_attribute>
<name>Measurement</name>
<ref_to>Measurement</ref_to>
<base_relation>measurement</base_relation>
<min_occurs>1</min_occurs>
<max_occurs>1</max_occurs>
<inverse_name>MeasurementQuantities</inverse_name>
</relation_attribute>
<relation_attribute>
<name>LocalColumns</name>
<ref_to>Localcolumn</ref_to>
<base_relation>local_columns</base_relation>
<min_occurs>0</min_occurs>
<max_occurs>Many</max_occurs>
<inverse_name>MeasurementQuantity</inverse_name>

```

```

</relation_attribute>
<relation_attribute>
<name>Quantity</name>
<ref_to>Quantity</ref_to>
<base_relation>quantity</base_relation>
<min_occurs>1</min_occurs>
<max_occurs>1</max_occurs>
</relation_attribute>
</application_element>
<application_element>
<name>Submatrix</name>
<basetype>AoSubmatrix</basetype>
<application_attribute>
<name>Id</name>
<base_attribute>id</base_attribute>
<obligatory>true</obligatory>
<unique>true</unique>
</application_attribute>
<application_attribute>
<name>Name</name>
<base_attribute>name</base_attribute>
<obligatory>true</obligatory>
</application_attribute>
<application_attribute>
<name>NumberOfRows</name>
<base_attribute>number_of_rows</base_attribute>
<obligatory>true</obligatory>
</application_attribute>
<relation_attribute>
<name>Measurement</name>
<ref_to>Measurement</ref_to>
<base_relation>measurement</base_relation>
<min_occurs>1</min_occurs>
<max_occurs>1</max_occurs>
<inverse_name>Submatrices</inverse_name>
</relation_attribute>
<relation_attribute>

```

```

<name>LocalColumns</name>
<ref_to>Localcolumn</ref_to>
<base_relation>local_columns</base_relation>
<min_occurs>0</min_occurs>
<max_occurs>Many</max_occurs>
<inverse_name>Submatrix</inverse_name>
</relation_attribute>
</application_element>
<application_element>
<name>Localcolumn</name>
<basetype>AoLocalColumn</basetype>
<application_attribute>
<name>Id</name>
<base_attribute>id</base_attribute>
<obligatory>true</obligatory>
<unique>true</unique>
</application_attribute>
<application_attribute>
<name>Name</name>
<base_attribute>name</base_attribute>
<obligatory>true</obligatory>
</application_attribute>
<application_attribute>
<name>GlobalFlag</name>
<base_attribute>global_flag</base_attribute>
<obligatory>true</obligatory>
</application_attribute>
<application_attribute>
<name>Flags</name>
<base_attribute>flags</base_attribute>
</application_attribute>
<application_attribute>
<name>Independent</name>
<base_attribute>independent</base_attribute>
<obligatory>true</obligatory>
</application_attribute>
<application_attribute>

```



```

<name>SequenceRepresentation</name>
<base_attribute>sequence_representation</base_attribute>
<obligatory>true</obligatory>
</application_attribute>
<application_attribute>
<name>Values</name>
<base_attribute>values</base_attribute>
<obligatory>false</obligatory>
</application_attribute>
<relation_attribute>
<name>MeasurementQuantity</name>
<ref_to>Measurementquantity</ref_to>
<base_relation>measurement_quantity</base_relation>
<min_occurs>0</min_occurs>
<max_occurs>1</max_occurs>
<inverse_name>LocalColumns</inverse_name>
</relation_attribute>
<relation_attribute>
<name>Submatrix</name>
<ref_to>Submatrix</ref_to>
<base_relation>submatrix</base_relation>
<min_occurs>1</min_occurs>
<max_occurs>1</max_occurs>
<inverse_name>LocalColumns</inverse_name>
</relation_attribute>
</application_element>
<application_element>
<name>Quantity</name>
<basetype>AoQuantity</basetype>
<application_attribute>
<name>Id</name>
<base_attribute>id</base_attribute>
<obligatory>true</obligatory>
<unique>true</unique>
</application_attribute>
<application_attribute>
<name>Name</name>

```

```

    <base_attribute>name</base_attribute>
    <obligatory>true</obligatory>
  </application_attribute>
  <application_attribute>
    <name>DefaultRank</name>
    <base_attribute>default_rank</base_attribute>
    <obligatory>true</obligatory>
  </application_attribute>
  <application_attribute>
    <name>DefaultDimension</name>
    <base_attribute>default_dimension</base_attribute>
  </application_attribute>
  <application_attribute>
    <name>DefaultDatatype</name>
    <base_attribute>default_datatype</base_attribute>
    <obligatory>true</obligatory>
  </application_attribute>
  <application_attribute>
    <name>DefaultTypeSize</name>
    <base_attribute>default_type_size</base_attribute>
    <obligatory>true</obligatory>
  </application_attribute>
  <application_attribute>
    <name>DefaultMqName</name>
    <base_attribute>default_mq_name</base_attribute>
    <obligatory>true</obligatory>
  </application_attribute>
  <relation_attribute>
    <name>DefaultUnit</name>
    <ref_to>Unit</ref_to>
    <base_relation>default_unit</base_relation>
    <min_occurs>1</min_occurs>
    <max_occurs>1</max_occurs>
    <inverse_name>Quantities</inverse_name>
  </relation_attribute>
</application_element>
<application_element>

```

```

<name>Unit</name>
<basetype>AoUnit</basetype>
<application_attribute>
<name>Id</name>
<base_attribute>id</base_attribute>
<obligatory>true</obligatory>
<unique>true</unique>
</application_attribute>
<application_attribute>
<name>Name</name>
<base_attribute>name</base_attribute>
<obligatory>true</obligatory>
</application_attribute>
<application_attribute>
<name>Factor</name>
<base_attribute>factor</base_attribute>
<obligatory>true</obligatory>
</application_attribute>
<application_attribute>
<name>Offset</name>
<base_attribute>offset</base_attribute>
<obligatory>true</obligatory>
</application_attribute>
<relation_attribute>
<name>PhysicalDimension</name>
<ref_to>Physicaldimension</ref_to>
<base_relation>phys_dimension</base_relation>
<min_occurs>1</min_occurs>
<max_occurs>1</max_occurs>
<inverse_name>Units</inverse_name>
</relation_attribute>
<relation_attribute>
<name>Quantities</name>
<ref_to>Quantity</ref_to>
<base_relation>quantities</base_relation>
<min_occurs>0</min_occurs>
<max_occurs>Many</max_occurs>

```

```

<inverse_name>DefaultUnit</inverse_name>
</relation_attribute>
</application_element>
<application_element>
<name>Physicaldimension</name>
<basetype>AoPhysicalDimension</basetype>
<application_attribute>
<name>Id</name>
<base_attribute>id</base_attribute>
<obligatory>true</obligatory>
<unique>true</unique>
</application_attribute>
<application_attribute>
<name>Name</name>
<base_attribute>name</base_attribute>
<obligatory>true</obligatory>
</application_attribute>
<application_attribute>
<name>LengthExponent</name>
<base_attribute>length_exp</base_attribute>
<obligatory>true</obligatory>
</application_attribute>
<application_attribute>
<name>MassExponent</name>
<base_attribute>mass_exp</base_attribute>
<obligatory>true</obligatory>
</application_attribute>
<application_attribute>
<name>TimeExponent</name>
<base_attribute>time_exp</base_attribute>
<obligatory>true</obligatory>
</application_attribute>
<application_attribute>
<name>CurrentExponent</name>
<base_attribute>current_exp</base_attribute>
<obligatory>true</obligatory>
</application_attribute>

```

```

<application_attribute>
<name>TemperatureExponent</name>
<base_attribute>temperature_exp</base_attribute>
<obligatory>true</obligatory>
</application_attribute>
<application_attribute>
<name>MolarAmountExponent</name>
<base_attribute>molar_amount_exp</base_attribute>
<obligatory>true</obligatory>
</application_attribute>
<application_attribute>
<name>LuminousIntensityExponent</name>
<base_attribute>luminous_intensity_exp</base_attribute>
<obligatory>true</obligatory>
</application_attribute>
<relation_attribute>
<name>Units</name>
<ref_to>Unit</ref_to>
<base_relation>units</base_relation>
<min_occurs>0</min_occurs>
<max_occurs>Many</max_occurs>
<inverse_name>PhysicalDimension</inverse_name>
</relation_attribute>
</application_element>
<application_element>
<name>User</name>
<basetype>AoUser</basetype>
<application_attribute>
<name>Id</name>
<base_attribute>id</base_attribute>
<obligatory>true</obligatory>
<unique>true</unique>
</application_attribute>
<application_attribute>
<name>Name</name>
<base_attribute>name</base_attribute>
<obligatory>true</obligatory>

```

```

</application_attribute>
<application_attribute>
<name>Password</name>
<base_attribute>password</base_attribute>
</application_attribute>
<relation_attribute>
<name>Groups</name>
<ref_to>Usergroup</ref_to>
<base_relation>groups</base_relation>
<min_occurs>0</min_occurs>
<max_occurs>Many</max_occurs>
<inverse_name>users</inverse_name>
</relation_attribute>
</application_element>
<application_element>
<name>Usergroup</name>
<basetype>AoUserGroup</basetype>
<application_attribute>
<name>Id</name>
<base_attribute>id</base_attribute>
<obligatory>true</obligatory>
<unique>true</unique>
</application_attribute>
<application_attribute>
<name>Name</name>
<base_attribute>name</base_attribute>
<obligatory>true</obligatory>
</application_attribute>
<relation_attribute>
<name>Users</name>
<ref_to>User</ref_to>
<base_relation>users</base_relation>
<min_occurs>0</min_occurs>
<max_occurs>Many</max_occurs>
<inverse_name>groups</inverse_name>
</relation_attribute>
<application_attribute>

```

```

<name>SuperuserFlag</name>
<base_attribute>superuser_flag</base_attribute>
<obligatory>true</obligatory>
</application_attribute>
</application_element>
<application_element>
<name>Process</name>
<basetype>AoAny</basetype>
<application_attribute>
<name>Id</name>
<base_attribute>id</base_attribute>
<obligatory>true</obligatory>
<unique>true</unique>
</application_attribute>
<application_attribute>
<name>Name</name>
<base_attribute>name</base_attribute>
<obligatory>true</obligatory>
</application_attribute>
<application_attribute>
<name>Description</name>
<base_attribute>description</base_attribute>
</application_attribute>
</application_element>
</application_model>
<instance_data>
<Environment>
<Id>90</Id>
<Name>MyEnvironment</Name>
<Tests>91</Tests>
</Environment>
<Test>
<Id>91</Id>
<Name>MyTest</Name>
<Environment>90</Environment>
<Subtests>92</Subtests>
</Test>

```

```

<Subtest>
<Id>92</Id>
<Name>MySubtest</Name>
<Test>91</Test>
<Measurements>93</Measurements>
<Result>Succeeded</Result>
</Subtest>
<Measurement>
<Id>93</Id>
<Name>MyMeasurement</Name>
<StartTime>20051202103115000000</StartTime>
<Subtest>92</Subtest>
<MeasurementQuantities>94 95 96 97 98</MeasurementQuantities>
<Submatrices>99</Submatrices>
<Responsible>108</Responsible>
<Process>2</Process>
</Measurement>
<Measurementquantity>
<Id>94</Id>
<Name>MyMqLong</Name>
<Datatype>DT_LONG</Datatype>
<Measurement>93</Measurement>
<LocalColumns>100</LocalColumns>
<Quantity>105</Quantity>
</Measurementquantity>
<Measurementquantity>
<Id>95</Id>
<Name>MyMqString</Name>
<Datatype>DT_STRING</Datatype>
<Measurement>93</Measurement>
<LocalColumns>101</LocalColumns>
<Quantity>105</Quantity>
</Measurementquantity>
<Measurementquantity>
<Id>96</Id>
<Name>MyMqFloat</Name>
<Datatype>DT_FLOAT</Datatype>

```



```

<Measurement>93</Measurement>
<LocalColumns>102</LocalColumns>
<Quantity>105</Quantity>
</Measurementquantity>
<Measurementquantity>
<Id>97</Id>
<Name>MyMqDouble</Name>
<Datatype>DT_DOUBLE</Datatype>
<Measurement>93</Measurement>
<LocalColumns>103</LocalColumns>
<Quantity>105</Quantity>
</Measurementquantity>
<Measurementquantity>
<Id>98</Id>
<Name>MyMqTime</Name>
<Datatype>DT_DATE</Datatype>
<Measurement>93</Measurement>
<LocalColumns>104</LocalColumns>
<Quantity>105</Quantity>
</Measurementquantity>
<Submatrix>
<Id>99</Id>
<Name>MyMeasurement</Name>
<NumberOfRows>2</NumberOfRows>
<Measurement>93</Measurement>
<LocalColumns>100 101 102 103 104</LocalColumns>
</Submatrix>
<Localcolumn>
<Id>100</Id>
<Name>MyMqLong</Name>
<GlobalFlag>15</GlobalFlag>
<Independent>1</Independent>
<SequenceRepresentation>explicit</SequenceRepresentation>
<Values>
<A_INT32>1 2</A_INT32>
</Values>
<MeasurementQuantity>94</MeasurementQuantity>

```

```

<Submatrix>99</Submatrix>
</Localcolumn>
<Localcolumn>
<Id>101</Id>
<Name>MyMqString</Name>
<GlobalFlag>15</GlobalFlag>
<Independent>0</Independent>
<SequenceRepresentation>explicit</SequenceRepresentation>
<Values>
<A_UTF8STRING>
<s>val1</s>
<s>val2</s>
</A_UTF8STRING>
</Values>
<MeasurementQuantity>95</MeasurementQuantity>
<Submatrix>99</Submatrix>
</Localcolumn>
<Localcolumn>
<Id>102</Id>
<Name>MyMqFloat</Name>
<GlobalFlag>15</GlobalFlag>
<Independent>0</Independent>
<SequenceRepresentation>explicit</SequenceRepresentation>
<Values>
<A_FLOAT32>700.320007324219 14.5299997329712</A_FLOAT32>
</Values>
<MeasurementQuantity>96</MeasurementQuantity>
<Submatrix>99</Submatrix>
</Localcolumn>
<Localcolumn>
<Id>103</Id>
<Name>MyMqDouble</Name>
<GlobalFlag>15</GlobalFlag>
<Independent>0</Independent>
<SequenceRepresentation>explicit</SequenceRepresentation>
<Values>
<A_FLOAT64>512.12 23.7</A_FLOAT64>

```

```

</Values>
<MeasurementQuantity>97</MeasurementQuantity>
<Submatrix>99</Submatrix>
</Localcolumn>
<Localcolumn>
<Id>104</Id>
<Name>MyMqTime</Name>
<GlobalFlag>15</GlobalFlag>
<Independent>0</Independent>
<SequenceRepresentation>explicit</SequenceRepresentation>
<Values>
<A_TIMESTRING>20050130121532000000 20050129115315000000</A_TIMESTRING>
</Values>
<MeasurementQuantity>98</MeasurementQuantity>
<Submatrix>99</Submatrix>
</Localcolumn>
<Quantity>
<Id>105</Id>
<Name>Length</Name>
<DefaultRank>0</DefaultRank>
<DefaultDimension>0</DefaultDimension>
<DefaultDatatype>DT_UNKNOWN</DefaultDatatype>
<DefaultTypeSize>0</DefaultTypeSize>
<DefaultMqName>has_no_name</DefaultMqName>
<DefaultUnit>106</DefaultUnit>
</Quantity>
<Unit>
<Id>106</Id>
<Name>m</Name>
<Factor>1.0</Factor>
<Offset>0.0</Offset>
<PhysicalDimension>107</PhysicalDimension>
</Unit>
<Physicaldimension>
<Id>107</Id>
<Name>m</Name>
<LengthExponent>1</LengthExponent>

```

```

<MassExponent>0</MassExponent>
<TimeExponent>0</TimeExponent>
<CurrentExponent>0</CurrentExponent>
<TemperatureExponent>0</TemperatureExponent>
<MolarAmountExponent>0</MolarAmountExponent>
<LuminousIntensityExponent>0</LuminousIntensityExponent>
<Units>106</Units>
</Physicaldimension>
<User>
<Id>108</Id>
<Name>Peter Sellers</Name>
<Password>*****</Password>
<Groups>110</Groups>
</User>
<User>
<Id>109</Id>
<Name>Todd Martin</Name>
<Password>*****</Password>
<Groups>110</Groups>
</User>
<Usergroup>
<Id>110</Id>
<Name>Division B</Name>
<Users>108 109</Users>
<SuperuserFlag>15</SuperuserFlag>
</Usergroup>
<Process>
<Id>2</Id>
<Name>QR-T22A-1</Name>
<Description>Process created by KX-234, attached to LQ-839, connection
Division B, signed off: K.Smithson</Description>
</Process>
</instance_data>
</atfx_file>

```

10.7 已知问题

在写作本章时，XML 包含规范还不被 W3C 定案，对于它的支持尚不普遍。

在写作本章时，ASAM XML Styleguide 规范在审查中，本章可能受到任何 Styleguide 变化的影响。

ATF/XML 规范的先前版本包含错误的例子，如由标签<enumtype>指定枚举类型。这是错误的，因为本标准总是将枚举标签定义为<enumeration_type>。然而，错误的 XML 文件可能存在于几个地方。ATF/XML 作者写作时必须符合实际规范。ATF/XML 读者也必须能够读懂这些错误的文件。

ATF/XML 规范的先前版本错误地包含了一些例子，在这些例子中标签被添加到指定值的数字格式的本地列中，而这是错误的。此外，<values>元素它们自身确实包含了这些值，没有内部元素指定这些值的数据类型。先前版本的 XML 文件可能存在于几个地方。ATF/XML 作者写作必须符合实际规范。ATF/XML 读者也必须能够读取这些旧式文件。

先前版本包含元素<locale>。ATF/XML 作者不能创建这个元素。ATF/XML 读者必须能够读取基于旧规范的文件。

先前版本将元素<application_enumeration>放在元素<application_element>之后。ATF/XML 作者必须符合实际序列。ATF/XML 读者必须能够读取基于旧规范的文件。

先前的版本允许指定来自列表的组件中值的数据类型，这个列表现在已经由基础模型中定义的 typespec_enum 取代。ATF/XML 作者写作必须符合 typespec_enum。ATF/XML 读者必须能够读取基于旧规范的文件。

先前版本允许在<instance_data>元素中以十六制形式写入字节流，以两个十六制数字指定每个字节（或者大写字母或者小写字母）。现在不再是是这种情况。相反地，每个字节必须被 0 到 255 范围内的十进制数字表示，这些数字分别被空格字符分开。这种变化允许这些值的 XML 验证。ATF/XML 作者写作必须符合实际表示形式。ATF/XML 读者也必须能够读取基于旧规范的文件。

10.8 附加资源

有关 XML，XML 模式，XML 内含物的更多信息请咨询 W3C XML 文件：

- 1) 可扩展的标记语言 (XML) <http://www.w3.org/XML/>;
- 2) XML 模式 Part 1: 结构 <http://www.w3.org/TR/xmlschema-1/>;
- 3) XML 模式 Part 2: 数据类型 <http://www.w3.org/TR/xmlschema-2/>;
- 4) XML 内含物 (XInclude) 版本 1.0 <http://www.w3.org/TR/xinclude/>。

11 MIME 类型与外部引用

11.1 简介

ASAM ODS 规范适用于试验测试环境中的数据集成。除了像测试这类正常的试验测试结果外，通常还有很多数据不适合用规定的对象类型描述。由于它们引用了类型 AoTest 或 AoUnitUndetTest 或其它应用元素，因此为这些数据的集成提供支持（例如 ODS 环境中作为 AoMeasurement 类型的快速傅立叶变换分析或者是 ODS 环境之外 AoUnitUndetTest 的碰撞试验的视频）。

被集成数据的类型是多种多样的，ODS 对此不做限制（例如视频、音频、分析数据、CAD 模型等），因此需要采样一种通用的方法包含有关这些数据的信息，并且允许它们可以物理保存在 ODS 中或者可

以通过引用说明 ODS 数据存储之外的数据。

为了说明被存储数据类型的信息，ASAM ODS 使用了 MIME 类型定义。本章规定了如何在 ASAM ODS 中使用 MIME 类型概念，此外还规定了 ASAM ODS 如何使用 MIME 类型支持对任何外部对象（ODS 之外的）的引用。

11.2 ASAM ODS MIME 类型的定义

11.2.1 作为可选基本属性的 MIME_TYPE

“mime_type”是个类似“version”的基本属性，如果该属性存在，则它的值支持以下所列的规则。ODS 中的 mime_type 只是应用元素的一个附加信息（属性）。

使用预定义的 MIME 类型机制不包括特定功能的定义（例如 ODS 服务器可以具有某些功能，而 ATF 文件则不能）。

对于 ODS 服务器而言，无需增加额外的功能。在读写该属性的医用中所有必要的功能都是需要的。

11.2.2 MIME_TYPE 字符串的内容

该属性的名称为“mime_type”。

该属性的类型与数据类型枚举名 DT_STRING 相关。

每个 mime_type 字符串必须用小写字母，具体见表 171。

mime_type 字符串由以下部分组成：

<application/x-><asam> "." <standardtype> "." (<subtype>)

表171 mime_type 字符串元素的含义

元素	含义
<application/x->	ASAM MIME 类型属于“application/x-”组（在注册后它们将属于“application”组）
<asam>	字符串的第一部分，定义它是个 ASAM 类型
<standardtype>	例如“aomeasurement”，定义它是哪种标准化的 MIME 类型（所有 ASAM ODS 基础元素都是标准类型，例如 aoany、aouser 等）
<subtype>	标准化的 mime_type 字符串可有附加的子类型（例如“specila-temperature-measurement”是 AoMeasurement 的子类型。子类型典型地在具体的应用模型中定义。）

在组（“application/x-”或“application”）之后，两部分之间要用一个圆点隔开。

注：除子类型之外的其它所有部分在 ODS 环境中都是已知的（例如因 AoMeasurement 的基础 ID，每个人都知道“aomeasurement”），为了减少冗余，ODS 定义：

——在 ODS 环境中，只有可选的 <subtype> 被用作 mime_type 字符串的短形式（例如：“special-temperature-measurement”）；

——在向 ODS 之外的环境导出或者桥接 ODS 对象的每个应用，都必须使用包含所有部分的长形式 mime_type 字符串（例如导出到 ATF 或者桥接到 ASAM-CEA 总线结构中时要使用

“application/x-asam.aomeasurement.special-temperature-measurement”)。

按照这些规则，mime_type 字符串可以是以下示例：

“application/x-asam.aomeasurement.special-temperature-measurement”；

“application/x-asam.aotest.fatiguetest”。

或者是 ODS 环境中的短形式：

“specialtemperature”。

ASAM ODS 中的每个基础元素也是一个标准化的 MIME 类型（例如 “AoAny”、“AoSubmatrix”、“AoUser” 等，要了解所有基础元素的列表，见 ASAM ODS 数据模型）。

11.2.3 通过 ODS API 读取 MIME 类型数据

由于 mime_type 属性是可选的，因此最初每个客户端必须查找应用元素是否有 “mime_type” 属性（不分大小写）。

如果该属性存在，则客户端必须检查它是否在实例中被赋值了。

如果它被赋值且可用，则客户端可以确信该字符串是个有效的 mime_type 字符串，并符合以下规则。

注：每个应用必须注意有两种类型的mime_type字符串：

- 长 MIME 类型：包含完整的 mime_type 字符串。只用于 ODS 以外的环境；
- 短 MIME 类型：不带任何 ODS 规范化前缀，只有子类型。仅用于 ODS 环境中。

11.2.4 通过 ODS API 写出 MIME 类型数据

每个写出 ODS 数据的应用都应支持 mime_type 属性，不论这些属性存在数据模型的何处。

写出数据的应用必须确保被写出的 mime_type 字符串支持以下规则。

注：每个应用必须注意有两种类型的mime_type字符串：

- 长 MIME 类型：包含完整的 mime_type 字符串。只用于 ODS 以外的环境；
- 短 MIME 类型：不带任何 ODS 规范化前缀，只有子类型。仅用于 ODS 环境中。

11.3 ASAM ODS 基础 MIME 类型

下表显示了 ODS 规范的基础 MIME 类型。更多的 MIME 类型可在已有和将来的 ASAM ODS 应用模型规范中定义。目前预定义的 ASAM ODS MIME 类型列表(见表 172)可从 ASAM 网络服务器获得。

表172 ASAM ODS 的基本 MIME 类型

application/x-asam.aoany
application/x-asam.aoattributemap
application/x-asam.aoenvironment
application/x-asam.aoexternalcomponent
application/x-asam.aoexternalcomponent.filenameurl
application/x-asam.aoexternalcomponent.flagsfilenameurl

application/x-asam.aolocalcolumn
application/x-asam.aolog
application/x-asam.aomeasurement
application/x-asam.aomeasurementquantity
application/x-asam.aonamemap
application/x-asam.aoparameter
application/x-asam.aoparameterset
application/x-asam.aophysicaldimension
application/x-asam.aoquantity
application/x-asam.aoquantitygroup
application/x-asam.aosubmatrix
application/x-asam.aosubtest
application/x-asam.aotest
application/x-asam.aotestdevice
application/x-asam.aotestequipment
application/x-asam.aotestequipmentpart
application/x-asam.aotestsequence
application/x-asam.aotestsequencepart
application/x-asam.aounit
application/x-asam.aounitgroup
application/x-asam.aounitundertest
application/x-asam.aounitundertestpart
application/x-asam.aouser
application/x-asam.aousergroup

11.4 外部引用的处理

对ODS环境之外的任何对象的引用都可通过另一个可选基础属性“external_references”实现。数据模型设计人员负责决定哪个应用元素将包含该属性。

在11.2.3和11.2.4节中已经描述了读和写的要求。

该属性可用于（可选地）ODS的每个应用元素，它包含一个3字符串的列表[0..n]；其中的每个字符串的数据类型由DataType枚举名DT_STRING说明。

描述：

这是文字描述，可用于保存对象的一些简单描述，便于客户无需打开此对象就可得到第一手信

息（某些“高级信息”）。

该描述的示例是：“这个对象包含整个载荷试验123的结果。”

contentType:

用于存储外部对象的MIME类型（关于contentType字符串的规则见前述。）

位置:

用于保存访问该对象的明确的定位字符串。它基于URL概念和CORBA 3.0 URL定义，由以下内容构成：

<protocol>://[<version>@][<host address>]/<key string>

（要了解URL的标准定义，可浏览互联网上它的规范）。

ASAM ODS为ODS对象定义了以下定位字符串：

"asamods://[<version>@]<asampath>"

定位符字符串的含义见表173。

表173 定位符字符串的含义

元素	含义
<version>	ODS规范的版本，它包括： — <major>：已发布ODS规范的主版本号 — “.” — <minor>：已发布ODS规范的小版本号 一个有效版本的例子是“5.2”
<asampath>	对象的ASAM路径。

ASAM ODS 将支持对任何外部对象（文件、服务器...）的引用，在 **ODS** 服务器中不打算评价这些对象。相关的应用（读、写、浏览等）负责使用被引用的对象。**ASAM ODS** 只负责通过 **API** 保存和导出这些对象的引用。

12 ASAM ODS RPC-API

12.1 程序声明

每个程序都有相应的数字识别码（亦称：程序号）和版本号。接口定义版本连接到正确的程序和呼叫参数。这将支持多版本同时运行。**ASAM ODS 5.2.0** 版的 **RPC-API** 与较早前的 **3.2** 版兼容。接口定义版（**aods.x**）没有改变，以避免客户端重复编译。

12.2 使用的数据类型

ASAM 已指定标准化的数据类型，并于 2002 年末予以了公布。这些数据类型都有标准化的名称，总是以 **A_** 开头。这些数据将会被 **ASAM ODS** 永久用于所有新规范中。

RPC-API 已经得到了多年的发展和规格化。当时，没有任何一个标准化的数据类型在 **ASAM** 协会得到认可。因此，**ASAM ODS** 决定对满足其需求的数据类型给予定义。这些数据类型总是以 **T_** 开头。

RPC-API 已在 ODS 服务器中使用，也被众多公司安装。A_模本对于 T_模本的代替，需要修改并替换已安装的服务器，也要改变通过 RPC-API 访问服务器的客户端（因为接口定义文件已被改变）。

此外，ASAM ODS 并不会进一步扩展或改变 RPC-API，其当前规格已被认为是固定的。

这就是为什么 ASAM ODS 已经决定不把标准的 ASAM 数据类型（A_）使用到 RPC-API 技术说明里和接口定义文件里的原因。

反而，ASAM ODS 技术说明书的第二章则是在阐述 T_模本和 A_模本的关系。应当注意的是数据类型的枚举和他们名字的枚举与 ASAM 数据类型说明书中所分的类相同，并且在多数情况下，这些数据类型本身是二进制兼容的。所以，如果需要，一对一映射可以被很容易的实现。

示例：基础模型对元 ID 的要求为 64 位的整数。5.4.3 说明了也可以使用一个 32 位的整数。这就是元 ID 的描述方式。

12.3 会话服务（开放式和封闭式）

当客户端试图访问 ASAM ODS 服务器时，需要打开一个会话。当客户端完成其任务时，则应当关闭会话。

尽管这是客户端被推荐采取的行为，但是服务器并不能保证这个行为的有效性，因为：

- a) 有可能每隔一段很长的时间（并非永久性的），客户端实现就会在某种条件下使会话被激活；
- b) 有可能会使客户端和服务器之间的网络连接出现中断的情况，因此客户端再也不能关闭会话。

然而，必须要避免的是，久而久之，一些服务器中处于意外激活状态的会话会越来越多。因此，ASAM ODS 服务器将会关闭所有在具体时间内对服务器不起作用的会话。这个超时期限也许会被设置在服务器启动的时候（使用一个三十分钟的默认值）。超时时间值是由服务器实现决定的。操作客户端执行的时候，应当注意，服务器管理员也许已经设置了不同时间值超时期限。

如果一个客户端试图在关闭会话后访问服务器，错误（E_NOTOPEN）就会被提示。必须认识到，客户端执行要在超时期限过期后进行，它是专门处理这个错误的。

12.3.1 AOP_OPENENV

（1）打开环境

这是服务器最先要调用的。服务器要核对访问权限并建立一个会话。属性也许会被设置为使用 AOP_SetPar（见下文）和 AOP_OPENEnv 上的自变量参数。通过这个访问反馈到的环境识别码将会被用在后续调用中（该网络地址被作为在后续调用中验证的附加识别码保存下来）。一旦会话被建立，一个冗余 AOP_OpenEnv 就会直接显示 S_OK，而不用再处理属性。

每一个新的会话都会有新的环境识别码，甚至于来自多用户系统中的客户端连接，比如 UNIX。打开环境程序见表 174。

表174 打开环境程序

OpenEnvRet AOP_OpenEnv (OpenEnvReq) = 1;

```

struct OpenEnvReq {
    AOP_NameV nvSeq<>;
};

union OpenEnvRet switch (AOP_Status retState) {
    case E_BSS: void;
    case E_SSS: void;
    case E_SECURITY: void;
    case E_MISC: void;
    case E_LICENSE: void;
    default: AOP_Id envId;
};

```

注意，任何一个缺省的返回状态（例如：E_SECURITY）就意味着一个错误。E_BSS 可能显示资源问题，如最大进程数或文件描述符过超载。

示例：关于安全的 nvSeq 示例。在该示例中，没有对 AOP_OpenEnv 的调用。

```

/* Create rpc client. */

cl = clnt_create (nodename, rpcNumber, AODSVERS3,"tcp");
/* RPC Client creation failed ? */

if (!cl) {
    /* Log connection failure. */

    sprintf (stderr, "Connect failed to %s\n", nodename);
    /* Handle client create error gracefully. */

    clnt_pcreateerror (nodename);
    /* Return with error status. */

    return (0);
}

/* Set timeout parameter. */

if (clnt_control (cl, CLSET_TIMEOUT, (char *) &timeout) !=TRUE) {
    sprintf (stderr, "Unable to set timeout %s\n",nodename);
    return (0);
}

/* Create credential.

* Since no keyserver is available on Linux, DES isn't possible.
*/

cl->cl_auth = authunix_create_default ();
/* Authentication valid ? */

if (!cl->cl_auth) {

```

```
        sprintf (stderr, "No authentication to %s\n", nodename);  
        return (0)  
    }  
}
```

12.3.2 AOP_CLOSEENV

关闭环境。
关闭环境，开始于 **OpenEnv**。如果环境被打开不止一次，那么环境必须也要不止一次被关闭。
关闭环境程序见表 175。

表175 关闭环境程序

CloseEnvRet AOP_CloseEnv (CloseEnvReq) = 2;
struct CloseEnvReq { /* to close open env. */ AOP_Id envId; }; struct CloseEnvRet{ AOP_Status retState; };

12.4 元信息服务

获得应用程序结构及其与基础结构的映射信息。

12.4.1 AOP_GETAPPLINF

获得应用程序结构信息。
这个功能反馈出应用程序结构的所有信息。这是一个带有应用程序名和模本（基础元素 AOP_BasElem）的应用元素的列表（标示符=AID）以及应用元素间的参考列表。
获取程序结构信息见表 176。

表176 获取程序结构信息

GetApplInfRet AOP_GetApplInf (GetApplInfReq) = 14;
struct GetApplInfReq { AOP_Id envId; /* environ. handle */ }; struct ApplInfSeq { /* application element infos */ AOP_Id aiAId; /* application element id */ AOP_BasElem aiBId; /*basic element Id */ AOP_Name aiName; /*application element name */ }; struct ApplRelSeq { /* application references infos*/

```
AOP_Id      arAId1;      /* from (0,1) resp. (n) */
AOP_Id      arAId2;      /* to (n) resp. (m) */
long        arRefNr;      /* attr number, 0 for n:m */
AOP_Name     arName;      /* reference name */
AOP_Flags    arConstr;     /* Constraint (e.g. not NULL) */
};

struct ApplInf {
    ApplInfSeq  aiSeq<>;      /*sequence of appelem info */
    ApplRelSeq  arSeq<>;      /*sequence of appref info */
};

union GetApplInfRet switch (AOP_Status retState) {
    case E_NOTOPEN: void;
    case E_BSS: void;
    case E_SSS: void;
    case E_DAC: void;
    case E_SECURITY: void;
    case E_MISC: void;
    default: ApplInf applInf;
};
```

12.4.2 AOP_GETATTR

该函数用于获得应用元素属性信息，程序见表 177。

客户端按请求划分元素识别码(aid 和 idd)。如果实例识别码(元素识别码的一部分)是 Any(=0)，则只返回应用程序属性，反之，已发送实例的实例属性也被返回。

表177 获得应用元素属性信息

```
GetAttrRet AOP_GetAttr (GetAttrReq) = 12;
```

```

struct AttrSeq {
    AOP_Name      aBName;      /* basic name */
                                /* 0 if not basic */
    AOP_Name      aAName;      /* appl. name */
    AOP_DataType  aDataType;    /* attribute data type */
    AOP_Id        aUnit;       /* unit if global defined */
};

struct AttrInf {
    AOP_BasElem  aBId;         /* basic element Id */
    AOP_Id       aBId;         /* application element Id */
    AttrSeq      aSeq<>;
};

struct GetAttrReq {
    AOP_Id       envId;        /* environ. handle */
    AOP_ElemId   elemId        /* application element */
};

union GetAttrRet switch (AOP_Status retState) {
                                /* return value structure */
    case E_NOTOPEN: void;
    case E_BSS: void;
    case E_SSS: void;
    case E_DAC: void;
    case E_SECURITY: void;
    case E_MISC: void;
    default: AttrInf attrInf; /* including entity */
};

```

目前没有办法通过协议层来改变元信息，因此，使用 ODS 5.2.0 版本发布的服务，其元信息都是只读的。

出于兼容性的考虑，这些结构应当和他们原来保持一样。

这个请求只被用来返回应用程序结构，没有实例信息，只返回应用属性的名称。

因为实例属性使用 **AOP_GetInstAttr**，这个函数返回实例属性名称，该实例属性不是应用模型的一部分。

12.5 应用元素值服务

包括通过使用应用元素索引提取和储存应用元素实例。

12.5.1 AOP_GETINSTREF

该函数用于获得实例索引，程序见表 178。

通过查询已被纳入索引范围的实例应用元素，就可以得到所有的实例。其被索引的应用元素将会被

AOP_GetApplInf 返回。利用索引名称就可以区分两个应用元素间的多重申请索引。如果一个空串有基础索引（根据基础结构定义，基础索引总是独一无二的）就说明它正在被使用。一个给定字符串与可用的索引名称是相匹配的，n:m 索引上的名称要么从 SVCREF 要么从 SVCATTR 获取。

表178 获得实例索引

GetInstRefRet AOP_GetInstRef (GetInstRefReq) = 25;
struct GetInstRefReq { AOP_Id envId; /* environ. handle */ AOP_ElemId elemId; /* instance */ AOP_Name refName; AOP_Id aId; /* of type */ }; union GetInstRefRet switch (AOP_Status retState) { /* return value structure */ case E_NOTOPEN: void; case E_BSS: void; case E_SSS: void; case E_DAC: void; case E_SECURITY: void; case E_MISC: void; default: AOP_ElemId elemList<>; };

12.5.2 AOP_SETINSTREF

该函数用于设置实例索引，程序见表 179。
这个方法可以被用来创建或清除 n:m 索引。
AOP_PutVal 用于 n:1 的情况。

表179 设置实例索引

SetInstRefRet AOP_SetInstRef (SetInstRefReq) = 28;
struct SetInstRefReq { AOP_Id envId; /* environ. handle */ AOP_ElemId elemId1; /* one element */ AOP_ElemId elemId2; /* the other */ AOP_Name refName; long onoff; /* insert (onoff=1) or remove (onoff=0) */ }; struct SetInstRefRet {

```
AOP_Status retState;

};
```

12. 5. 3 AOP_GETVAL

该函数用于获得应用元素值，程序见表 180。

这是获得应用元素值的标准方法。有两种不同方法可被实现：

a) 通过索引

主要通过浏览（例如：请为我执行 ID723 的所有测试方法或者请为我测试 ID7895 的所有实例）。它可以允许 AODS 数据模型把 “has a” 作为其实例索引，如关于 API 的章节里说明的一样。

b) 通过元素值

在选择应用元素的实例时，通过指定满足选择结果的条件来实现。因为 AODS 协议还没有查询语言，所以能被实现的可能性很低，例如：

```
select * from test where date='19951201000000' and testtype='Endurance
Test'.
```

选择方法可以指定索引元素识别码或者提供非空的选择列表。如果先通过索引两者都可以得到，然后结果通过检测与选择列表匹配（使用 RDBMS 两个方法可以在同一步骤完成）。

元素识别码和索引名称在 AOP_GetInstRef 上的使用可采用同一种方法（见上文）。

表180 获得应用元素值

```
GetValRet AOP_GetVal (GetValReq) = 21;

struct AOP_NameS { /* by value select */
    AOP_Name        name;          /* attribute name */
    AOP_ValMap       valMap;        /* range or pattern */
    AOP_SelOpcode    selOpcode;     /* type of selection */
};

typedef AOP_NameS NSSeq<>;

struct GetValReq { /* selection request structure */
    AOP_Id           envId; /* environ. handle */
    AOP_Id           applId; /* application element Id (from) */
    AOP_NameU        nuSeq<>; /* sequence of to be reported attr's */
    AOP_NameS        nsSeq<>; /* sequence of select fields */
    AOP_ElemId       elemId; /* child/parent reference (where) */
    AOP_Name         refName; /* named reference identifier */
};

union GetValRet switch (AOP_Status retState) {
    /* select return value structure */
    case E_NOTOPEN: void;
    case E_BSS: void;
```



```

    case E_SSS: void;
    case E_DAC: void;
    case E_SECURITY: void;
    case E_MISC: void;
    default: AOP_NameV nvSeq<>; /* sequence of resulting attributes
*/
};

enum AOP_SelOpcode { /* Type of selection */
    SOMATCH, /* match == */
    SOUMATCH, /* unmatched != */
    SORANGE, /* range < x > */
    SOLESS, /* less < */
    SOGREATER, /* greater > */
    SOLESSEQ, /* less equal <= */
    SOGREATEREQ, /* greater equal >= */
    SOINSET, /* within set in */
    SONOTINSET, /* not within set !in */
    SOORDER, /* order by */
    SOGROUP, /* group by */
    SOINSENSITIVE, /* insensitive search */
    SOLIKE, /* like GetVale */
    SONOTLIKE, /* not like GetVale */
    SONULL, /* is NULL GetVale */
    SONOTNULL, /* is NOT NULL GetVale */
    SOOPAND, /* operator AND GetVale */
    SOOPOR, /* operator OR GetVale */
    SOOPNOT, /* operator NOT GetVale */
    SOOPBOPEN, /* operator bracket open ( GetVale */
    SOOPBCLOSE /* operator bracket close ) GetVale */
};

```

注意，AOP_NameS.valMap 结构可包含一个值（SOMATCH，.....）或二个值（SORABGE）或一至多个值（SOINSET, SONOTINSET）。为了满足通过显式指定实例识别码进行数据获取的 AODS 需求，必须将 AOP NameS 设置为：

```

name = <IdAttr>
valMap.AOP_ValMap_u.lval.lval[0] = <iid>
selOpcode = SOMATCH

```

其中，IdAttr 是一个应用元素的 ID-基础属性，使用 GetAttr 并已被命名有应用程序名。为了获得 ID-应用程序属性名称，可以在结构中设定 alret 来提示 aop getattr 的反馈。

```

char* GetIdAttr (GetAttrRet *alret) {
    AttrSeq *aSeq; /* attribute info */
    int aNr; /* attribute number */

```

```
aSeq = alret->GetAttrRet_u.attrInf.aSeq.aSeq_val;
for (aNr = 0; aNr < alret->GetAttrRet_u.attrInf.aSeq.aSeq_len; aNr++)
{
    if (!strcmp(aSeq->aBName,"ID")
        return aSeq->aAName;    /* pointer to application name */
    aSeq++;
}
return NULL;
}
```

需要注意如下情况:

- a) 在 SOMATCH 里没有通配符能被字符串使用。只能用 SOLIKE 或 SONOTLIKE 来代替使用;
- b) 不像字符串, SOINSENSITIVE 不用分大小写;
- c) SOORDER 和 SOGROUP 不需要 nsSeq 来使用 valMap, 因为这些运算符只有属性, 而没有值;
- d) SOORDER 允许为属性排序。这个排序命令通常很容易获得(低级值往往最先获得)。

12.5.4 AOP_GETVALE

该函数用于获得应用元素值(ODS 5.2.0.版的扩展版(RPC)), 程序见表 181。

如果需要从多个应用元素中选择数据或者想要使用布尔操作符, 就可以使用这个方法。于是, 分段请求就有可能被提出。

表181 获得应用元素值

<pre>GetValERet AOP_GetValE (GetValEReq) = 26; /*Enhanced Query structure, allows - joins of application elements - to specify ranges and patterns (Values are always arrays. This allows i.e. on selection to pass 2 or more values for e.g.: in/notin, range from - to) - to group selections with AND, OR and brackets (open, close) - to sort the result (order by) - to specify wildcards (including escape character) - partial access of the result (e.g. first 100, next 100, ..) To cover the AODS requirement of making data access by explicitly specifying instance id (direct access to single element) it is necessary to define the selection elements AOP_AIDNames with selOpcode = SOMATCH AID = <application element ID> name = <Idattr></pre>

```

        value.AOP_ValMap_u.lval.lval[0] = <iid>
    IdAttr is the first aBName reportet using GetAttr
*/
enum AOP_SelOpcode { /* Type of selection */
    SOMATCH,          /* match                ==          */
    SOUMATCH,         /* unmatched            !=          */
    SORANGE,          /* range                < x >      */
    SOLESS,           /* less                 <          */
    SOGREATER,        /* greater              >          */
    SOLESSEQ,         /* less equal           <=         */
    SOGREATEREQ,      /* greater equal        >=         */
    SOINSET,          /* within set           in          */
    SONOTINSET,       /* not within set       !in         */
    SOORDER,          /* order by             */
    SOGROUP,          /* group by             */
    SOINSENSITIVE,    /* insensitive search   */
    SOLIKE,           /* like                 */
    SONOTLIKE,        /* not like             */
    SONULL,           /* is NULL              */
    SONOTNULL,        /* is NOT NULL          */
    SOOPAND,          /* operator AND         */
    SOOPOR,           /* operator OR          */
    SOOPNOT,          /* operator NOT         */
    SOOPBOPEN,        /* operator bracket open ( */
    SOOPBCLOSE        /* operator bracket close ) */
};
/* ----- AID + Name + Unit */
struct AOP_AIDNameU {
    AOP_Id    aid;      /* application element ID */
    AOP_Name  name;     /* attribute name */
    AOP_Id    unitId;   /* requested unit */
};
typedef AOP_AIDNameU ANUSeq<>;
/* ----- AID + Name + Unit + Value(s) */
struct AOP_AIDNameV {
    AOP_Id    aid;      /* application element ID */
    AOP_Name  name;     /* attribute name */
    AOP_Id    unitId;   /* requested unit */
    AOP_ValMap valMap;  /* values */
};
typedef AOP_AIDNameV ANVSeq<>;
enum AOP_ReportListType { /* type of report list */
    RLCOMPLETE, /* all attributes */

```

```

    RLSELECTIVE /* given list of attributes */
};

union AOP_ReportList switch (AOP_ReportListType rlType) {
    case RLCOMPLETE:
        AOP_Id      aidSeq<>; /* list of application elements - all
                                attributes will be reported */

    case RLSELECTIVE:
        AOP_AIDNameU anuSeq<>; /* list of attribute to be reported */
    default: void;
};

struct AOP_AIDNameS { /* by value select */
    AOP_SelOpcode    selOpcode; /* type of selection */
    AOP_Id           aid; /* application element ID */
    AOP_Name         name; /* attribute name */
    AOP_ValMap       valMap; /* range or pattern */
    AOP_SortDir      dir; /* sort order */
};

typedef AOP_AIDNameS ANSSeq<>;
struct AOP_RefDef { /* reference definition */
    AOP_Id          fromAid; /* start point (AID) of reference */
    AOP_ElemId      elemId; /* child/parent reference (where) */
    AOP_Name        refName; /* named reference identifier */
};

enum AOP_ReqType { /* type of request */
    RTDEFAULT,      /* force query and close cursor */
    RTOPEN,         /* force query and keep cursor open */
    RTCONTINUE,     /* continue reading cursor */
    RTCLOSE         /* close open cursor */
};

struct GetValEReq { /* enhanced query request structure */
    AOP_Id          envId; /* environ. handle */
    AOP_ReqType      reqType; /* request type */
    AOP_ReportList   repList; /* definition of report list */
    AOP_AIDNameS     nsSeq<>; /* sequence of conditions */
    AOP_RefDef       refDef; /* reference definition */
    long            rowCnt; /* number of requested rows
                            - rowCnt<=0 report all rows */
};

union GetValERet switch (AOP_Status retState) {
    /* select return value structure */
    case E_NOTOPEN: void;
    case E_BSS: void;
    case E_SSS: void;
};

```

```

    case E_DAC: void;
    case E_SECURITY: void;
    case E_MISC: void;
    default: ANVSeq result<>; /* sequence of resulting attribute
                                sequences */
};

```

AOP_ReportList 上的应用元素必须具有一个索引, 指向 AOP_RefDef.fromAid 使用的应用元素的, 或者指向被选定属性值 AOP_AIDNameS.aid 使用的应用元素。如果没有索引, 应用元素实例中就不会被报告。

示例1: 该示例阐述了 aop_getvale 的使用。根据其特性, 发出了如下一个新的请求:

- 1) 无分割;
- 2) 无定义属性-只定义应用元素;
- 3) 未选定标准;
- 4) 未明确定义合并 — 若不模糊则没有必要, 该示例并未使用合并;
- 5) 未给出参考实例。

最后一行包括方法调用。

```

GetValEReq vereq;
GetValERet *veret;
...
vereq.envId = sessid;
vereq.reqType = RTOPEN; /* force a new select */
vereq.rowCnt = -1; /* report the whole result */
/* request all attributes (select *) */
vereq.repList.rlType = RLCOMPLETE; /* select * */
vereq.repList.AOP_ReportList_u.aidSeq.aidSeq_len = 1;
vereq.repList.AOP_ReportList_u.aidSeq.aidSeq_val =
malloc(sizeof(AOP_Id));
vereq.repList.AOP_ReportList_u.aidSeq.aidSeq_val[0] = aid;
/* no select conditions */
vereq.nsSeq.nsSeq_len = 0;
vereq.nsSeq.nsSeq_val = NULL;
/* no explicit join definition */
vereq.jdSeq.jdSeq_len = 0;
vereq.jdSeq.jdSeq_val = NULL;
/* no reference definition used */
vereq.refDef.fromAid = 0;

```

```

vereq.refDef.refName = strdup("");
vereq.refDef.elemId.aid = 0;
vereq.refDef.elemId.iid = 0;
veret = (GetValERet *) aop_getvale_3 (&vereq, cl);

```

示例2: 该代码示例依靠以下规定:

- 测试序列的名称是“MOTID”;
- 测量名称是“Messung”;
- 数量名称是“GroNameDef”;
- 数量群名称是“GroGruName”;
- 只选定一个标准: 测量名称=“267_FU”。

```

/* request specific attributes */
vereq.repList.rlType = RLSELECTIVE; /* select with attribute list */
vereq.repList.AOP_ReportList_u.anuSeq.anuSeq_len = 2;
vereq.repList.AOP_ReportList_u.anuSeq.anuSeq_val = malloc (2*sizeof (AOP_AIDNameU));
vereq.repList.AOP_ReportList_u.anuSeq.anuSeq_val[0].aid = 2;
vereq.repList.AOP_ReportList_u.anuSeq.anuSeq_val[0].name =strdup ("MOTID");
vereq.repList.AOP_ReportList_u.anuSeq.anuSeq_val[0].unitId = 0;
vereq.repList.AOP_ReportList_u.anuSeq.anuSeq_val[1].aid = 4;
vereq.repList.AOP_ReportList_u.anuSeq.anuSeq_val[1].name =strdup ("Messung");
vereq.repList.AOP_ReportList_u.anuSeq.anuSeq_val[1].unitId = 0;
/* one select condition */
vereq.nsSeq.nsSeq_len = 1;
vereq.nsSeq.nsSeq_val = malloc (sizeof (AOP_AIDNameS));
vereq.nsSeq.nsSeq_val[0].selOpcode = SOMATCH;
vereq.nsSeq.nsSeq_val[0].aid = 4;
vereq.nsSeq.nsSeq_val[0].name = strdup ("Messung");
vereq.nsSeq.nsSeq_val[0].valMap.dtyp = DT_STRING;
vereq.nsSeq.nsSeq_val[0].valMap.AOP_ValMap_u.aval.aval_len=strlen ("267_FU")+1;
vereq.nsSeq.nsSeq_val[0].valMap.AOP_ValMap_u.aval.aval_val=strdup ("267_FU");
vereq.nsSeq.nsSeq_val[0].dir = 0;
/* request specific attributes */
vereq.repList.rlType = RLSELECTIVE; /* select with attribute list */
vereq.repList.AOP_ReportList_u.anuSeq.anuSeq_len = 2;
vereq.repList.AOP_ReportList_u.anuSeq.anuSeq_val =malloc (2*sizeof (AOP_AIDNameU));
vereq.repList.AOP_ReportList_u.anuSeq.anuSeq_val[0].aid = 7;
vereq.repList.AOP_ReportList_u.anuSeq.anuSeq_val[0].name=strdup ("GroNameDef");

```

```
vereq.repList.AOP_ReportList_u.anuSeq.anuSeq_val[0].unitId = 0;
vereq.repList.AOP_ReportList_u.anuSeq.anuSeq_val[1].aid = 11;
vereq.repList.AOP_ReportList_u.anuSeq.anuSeq_val[1].name=strdup ("GroGruName");
vereq.repList.AOP_ReportList_u.anuSeq.anuSeq_val[1].unitId = 0;
```

示例3: 在该示例代码中，也有一个选定的标准：数量群名称= "B*"。

```
/* one select condition */
vereq.nsSeq.nsSeq_len = 1;
vereq.nsSeq.nsSeq_val = malloc (sizeof (AOP_AIDNames));
vereq.nsSeq.nsSeq_val[0].selOpcode = SOLIKE;
vereq.nsSeq.nsSeq_val[0].aid = 7;
vereq.nsSeq.nsSeq_val[0].name = strdup ("GroNameDef");
vereq.nsSeq.nsSeq_val[0].valMap.dtyp = DT_STRING;
vereq.nsSeq.nsSeq_val[0].valMap.AOP_ValMap_u.aval.aval_len =strlen ("B*") + 1;
vereq.nsSeq.nsSeq_val[0].valMap.AOP_ValMap_u.aval.aval_val =strdup ("B*");
vereq.nsSeq.nsSeq_val[0].dir = 0;
```

示例4:

```
/* one join definition */
vereq.jdSeq.jdSeq_len = 1;
vereq.jdSeq.jdSeq_val = malloc (sizeof (AOP_JoinDef));
vereq.jdSeq.jdSeq_val[0].fromAid = 7;
vereq.jdSeq.jdSeq_val[0].toAid = 11;
vereq.jdSeq.jdSeq_val[0].refName = strdup ("GROEBEZ");
vereq.jdSeq.jdSeq_val[0].joinType = JTOUTER;
```

12.5.5 AOP_GETINSTATTR

该函数用于获得一些元素的实例属性，程序见表 182。

实例属性是不被物理数据(元数据)模型模式化的属性。它们可以被附加于每一个带有元素的实例。实例属性被储存在一个单独的表上，并且属性名称和值可以通过这个方法被索引。这个结果取决于元标识符的值。如果只设置了 elemId.aid，那么 AOP_GetInstAttr 将会返回所有指定应用元素可获得的实例属性信息(名称、数据类型和单元，而没有值)。如果元标识符定位了一个实例((elemId.aid 和 elemId.iid 均有效)，那么属于特定实例的所有实例属性将会被返回。但不支持 DT_BLOB 和 DT_BYTESTR 这两种数据模本。

表182 获得一些元素的实例属性

GetInstAttrRet	AOP_GetInstAttr (GetInstAttrReq) = 22;

```
struct GetInstAttrReq { /* selection request structure */
    AOP_Id      envId; /* environ. handle */
    AOP_ElemId  elemId; /* application element Id */
};

union GetInstAttrRet switch (AOP_Status retState) {
    /* select return value structure */
    case E_NOTOPEN: void;
    case E_BSS: void;
    case E_SSS: void;
    case E_DAC: void;
    case E_SECURITY: void;
    case E_MISC: void;
    default: AOP_NameV nvSeq<>; /* sequence of resulting attributes */
};
```

12.5.6 AOP_PUTVAL

该函数用于应用元素的存储操作，程序见表 183。

这个请求被客户端用于在应用元素上插入、上载和删除操作。这个请求只能在一个访问上处理一个实例。客户端请求的运算操作，是由请求结构里的值进行默认进行的。如果没有实例（在元标识中 `idd==0`），指定一个插入操作等于进行了一个新的应用元素实例的建立，新的实例标识符就会被返回。如果实例标识被指定了，操作就会在 `nvSeq` 参数下进行。一个空的 `nvSeq` 通常是在给出的标识上进行一个删除操作，否则，就会执行一个上载操作。

- 注1：一个已被删除实例的二次输入行为不属于 ODS 的一部分。服务器不能分辨新的实例输入和一个已经被删除实例的二次输入。因此，被二次输入的实例就会得到一个新的实例标识（总之，访问将会通过 ASAM 得以实现）。
- 注2：作为物理存储器当前的技术说明，限制了实例属性的名称长度，最多只能输入 30 个字符，客户端必须注意不要超出这个限制。

表183 应用元素的存储操作

```
PutValRet AOP_PutVal (PutValReq) = 27;

struct PutValReq { /* update request structure */
    AOP_Id      envId; /* environ. handle */
    AOP_ElemId  elemId; /* application element Id */
    AOP_NameV   nvSeq<>; /* sequence of values */
};

union PutValRet switch (AOP_Status retState) {
    /* select return value structure */
    case E_NOTOPEN: void;
    case E_BSS: void;
    case E_SSS: void;
    case E_DAC: void;
```



```
case E_SECURITY: void;
case E_MISC: void;
default: AOP_Id iid; /* storage instance id */
};
```

12.6 测量和子阵服务

以下函数表示实现的固有操作部分（不是元模型）。所有的方法都需要测量应用元素的实例识别码作为输入参数。如果没有应用元素识别码，服务器就会默认 Aomeasurement 模本的应用元素。

12.6.1 AOP_GETVALATTR

该函数用于获得子阵属性信息，程序见表 184。

服务器返回指定子阵的频道列表和子阵中的测量点数。子阵的细分对于客户端来说是透明的。如果 pmatid 是 0，那么所有子阵的返回信息都属于给定测量。

表184 获得子阵属性信息

GetValAttrRet AOP_GetValAttr (GetValAttrReq)
<pre>struct ValAttrSeq { /* linked list of attributes */ AOP_Name name; /* appl. structure name */ AOP_Id qtyId; /* reference to dict. (0 allowed) */ AOP_Id unitId; /* UnitId */ AOP_IndepFl indepFl; /* independent flag */ AOP_ImplFl implFl; /* implicit flag */ AOP_DataType dataType; /* attribute data type */ }; struct ValAttr { /* measurement attribute cursor list */ AOP_Id pmatId; /* PMatId */ long numPnt; /* number of measurement point in partial matrix */ /* should be extended for min/max */ ValAttrSeq vaSeq<>; /* attribute sequence */ }; struct GetValAttrReq { /* request for attribute list */ AOP_Id envId; /* environ. handle */ AOP_ElemId elemId; /* measurement id */ AOP_Id pmatId; /* partial matrix id (0 = all) */ }; union GetValAttrRet switch (AOP_Status retState) { /* return value structure */ case E_NOTOPEN: void;</pre>

```
case E_BSS: void;
case E_SSS: void;
case E_DAC: void;
case E_SECURITY: void;
case E_MISC: void;
default: ValAttr valAttr<>;/*column info for 1 or more PMat's */
};
```

QtyId 指的是 AoQuantity 类型应用元素的实例 (idd)。

UnitId 指的是 AoUnit 基础类型应用元素的实例 (idd)。

IndepFI = 1 指的是一单独列。

隐含列指示器会改变他们自身值的解读，见表 185。通常，值都是显式存储的。对某些数据（例如：时间驱动取样上的时间标记），可计算每一行的值。将提供以下变量（n=子阵行数）：

表185 对于隐式列区间计算值的算法

计数值 (numPnt)	在第n行的数值意义
1	$x_n = x_1$ 常量
2	$x_n = x_1 + (n-1) * x_2$ 线性表达式
3	$x_n = x_1 + ((n-1) \bmod (x_3 - x_1) / x_2) * x_2$ 锯齿形曲线

只有数值数据类型可以被隐性存储，使用已定义列数据类型就可以进行计算。公式 $(x_3 - x_1) / x_2$ 被圆整，以 x_1 作为每一个锯齿形曲线周期的开头。

DT_DATE 不是一种数值数据模本。

已测量值不支持 DT_BLOB 数据模本。

12. 6. 2 AOP_GETVALINF

该函数用于获得子阵信息，程序见表 186。

这个操作用来返回被测量子阵的信息。

与 AOP_GetValAttr 类似，其主要目的是提供采用列区间信息关联多重子阵方法的信息。

表186 获得子阵信息

```
GetValInfRet AOP_GetValInf (GetValInfReq) = 35;
struct ValInfSeq { /* partial matrix interrelations */
    AOP_Id      pmatId;
    AOP_Name    pmatName; /* sprintf of pmatId */
    AOP_Name    presortNames <>; /* columns to sort in pmat */
    AOP_SortDir presortDir; /* sortdirection */
    AOP_Name    joinNames <>; /* join columns */
    AOP_JoinTyp joinTyp;
```

```

    AOP_Name      postsortNames <>; /* columns to sort on result */
    AOP_SortDir    postsortTyp;
};
struct GetValInfReq {
    AOP_Id        envId;
    AOP_ElemId    elemId; /* aid has to be measurement */
};
union GetValInfRet switch (AOP_Status retState) {
    /* info fetch return value structure */

    case E_NOTOPEN: void;
    case E_BSS: void;
    case E_SSS: void;
    case E_DAC: void;
    case E_SECURITY: void;
    case E_MISC: void;
    default: ValInfSeq iSeq<>; /* sequence of info's */
};
```

JoinNames 和 joinTyp 字段说明了不同测量子矩阵之间的关系。目前，只有 joinTyp MERGE 在使用，而 joinNames 则是单独列的名称。presortNames、presortDir、presortNames 和 presortDir 字段目前还没有投入使用。JoinNames 和 joinTyp 字段说明了不同测量子矩阵之间的关系。目前，只有 joinTyp MERGE 在使用，而 joinNames 则是单独列的名称。presortNames、presortDir、presortNames 和 presortDir 字段目前还没有投入使用。

12.6.3 AOP_GETVALVAL

该函数用于获得测量值，程序见表 187。
这个服务将返回一个或多个子阵的测量数据。也可能通过指定列和局部测量点数来传输子集。

表187 获得测量值

```

GetValValRet AOP_GetValVal (GetValValReq) = 32;
struct ValValSeq { /* linked list of reported vectors */
    AOP_Name      name; /* MQ Name */
    AOP_Id        qtyId; /* dictionary Id (ref) */
    AOP_Id        unitId; /* unit Id */
    AOP_IndepFl    indepFl; /* independent flag */
    AOP_ImplFl     implFl; /* implicit (versus explicit) */
    AOP_ValMap     valMap; /* vector for values */
    AOP_ValFlags   valFlags; /* vector for values flags */
};
/* the partial matrix now holds 2 numbers to be able to preserve
* origin of data
```

```

*/
struct ValVal { /* local column values */
    AOP_Id      pmatId; /* pMatId (0 .. yet undef) */
    long        mPntBase; /* local point number base */
    long        mPntCnt; /* # local points */
    ValValSeq    vvSeq<>; /* measurement value list */
};

struct GetValValReq { /* request for measurement results */
    AOP_Id envId; /* environ. handle */
    AOP_ElemId elemId; /* elementId, bid to elemId.aid has to be
                        BE_MEA */
    AOP_Id pmatId; /* partial matrix number */
    long mPntBase; /* from local measurement pointnumber */
    long mPntCnt; /* number of measurement points requested */
    AOP_NameU nuSeq<>; /* sequence of attributes to report */
};

union GetValValRet switch (AOP_Status retState) {
    /* select return value structure */

    case E_NOTOPEN: void;
    case E_BSS: void;
    case E_SSS: void;
    case E_DAC: void;
    case E_SECURITY: void;
    case E_MISC: void;
    default: ValVal valVal<>; /* partial matrices */
};

```

mPntBase 的值以 1 开头，且必须是正整数。

为了获得更高的效率，一个值的所有标记都被编成一个字节，即一个标记将占用一个字节。为了获得单个字节，位掩码的常量被定义。可支持下列值的标记：

- a) AO_VF_VALID (0x01) - 有效值；
- b) AO_VF_VISIBLE (0x02) - 必须为可视值；
- c) AO_VF_UNMODIFIED (0x04) - 未被修改值；
- d) AO_VF_DEFINED (0x08) - 被定义值（这个标记也用于通过基极层来标记价值矩阵的间隙）。

12.6.4 AOP_PUTVALVAL

该函数用于在测量中纳入值，程序见表 188。

为了区分上载、插入、添加、删除 valVal 提供的信息，进行如下说明：

- 1) 添加到第一个匹配子矩阵，或者，如果没有找到相匹配子矩阵，可以用指定的局部列编号创建一个新子矩阵。局部列的属性和值可以通过 vvSeq_val 获得；

- 2) 删除所有被 **pmatId** 识别的子矩阵;
 - 3) 从所有局部列中删除指定的子矩阵, 元素编号 (**mPntCnt**) 从最先指定的元素 (**mPntBase**) 开始;
 - 4) 添加到所有指定子矩阵的局部列, 子矩阵元素在 **vvSeq_val** 中给出。指定子矩阵的局部列编号必须和请求结构中给出的局部列编号一致。局部列的顺序并不重要, 其名称比较关键。而局部列的描述信息, 例如独立隐性标志必须和服务器中的数据一致。如果子矩阵不存在, 那么它将会被创立。如果有一个新的局部列, 那么客户端就不能指定一个 **pmatid** (**pmatid=0**, 见案例一), 因为服务器必须选择一个正确的子矩阵, 或者建立一个新的子矩阵;
 - 5) 插入到指定子矩阵的所有局部列, 其子矩阵的元素在 **vvSeq_val** 中给出。数据会在指定的第一点出现之后被插入, 如果指定的第一点在现存局部列范围外, 则数据就会被添加。指定子矩阵的局部列编号必须和请求结构中给出的局部列编号一致。局部列的顺序并不重要, 其名称比较关键。而局部列的描述信息, 例如独立隐性标志, 必须与服务器中的数据一致;
 - 6) 替换指定子矩阵的指定局部列元素编号, 每一个局部列的值编号 (**vvSeq_val.valMap.AOP_ValMap_u.lval.lval_len**) 必须和元素指定编号一致。((**mPntCnt**))。
- 数据存储阶段服务器行为一览表见表 189。

表188 数据存储阶段服务器行为一览表

pmatId	mPntBase	mPntCnt	vvSeqLen	action
0	any	any	0	invalid
0	0	0	>0	1.)
>0	0	0	0	2.)
>0	>0	0	0	invalid
>0	>0	>0	0	3.)
>0	0	any	>0	4.)
>0	>0	0	>0	5.)
>0	>0	>0	>0	6.)

最后要更改的元素必须在局部列的现有规格之内。当插入位置超出现有范围, 插入就会被自动转换成添加。如果全部列应同一请求 (结构匹配、列的顺序并不重要) 提供时, 那么对现有子矩阵的添加和插入才能被授权。也许用“每一列”的方法可以实现替换, 但是被替换点的编号必须与同一案例中的这个请求所提供的点编号匹配。目前没有“由值”定点的可能, 例如, 使用隐性列来定义局部测量点编号。

表189 在测量中纳入值

PutValValRet PutValVal (PutValValReq)	
<pre> struct PutValValReq { /* request for measurement results */ AOP_Id envId; /* environ. handle */ AOP_ElemId elemId; /* elementId, bid to elemId.aid has to be BE_MEA */ </pre>	

```
ValVal      valVal<>; /* sequence of headers + columns */
                                   /* (only one PMat for now per call) */
};
union PutValValRet switch (AOP_Status retState) {
                                   /* select return value structure */
    case E_NOTOPEN: void;
    case E_BSS: void;
    case E_SSS: void;
    case E_DAC: void;
    case E_SECURITY: void;
    case E_MISC: void;
    default: AOP_Id pmatId; /* pmatid given by server */
};
```

12.7 属性处理

属性就是名-值对，名称类型为 `aods_Name`，值类型为 `aop_ValMap`，它能获得 AODS-认可的包括数组在内的任意数据类型。这种元组可以被传输到服务器，但不会被永久保存。主要目的是确定服务器对一客户端的行为参数（属性属于会话）。如果服务器这一端有其他外挂程序（如格式解码器），参数就可以提供附加信息（例如，不同种类的公式或用于公式中的附加变量的选择）。

12.7.1 AOP_SETPAR

该函数用于设置参数，程序见表 190。
在当前可以设定的服务器参数见表 191。

表190 设置参数

```
SetParRet AOP_SetPar (SetParReq) = 51;
struct SetParReq {
    AOP_Id      envId; /* environ. handle */
    AOP_NameV    nvSeq<>; /* parameters */
};
struct SetParRet {
    AOP_Status   retState;
};
```

表191 在当前可以设定的服务器参数

参数	用途	示例
USER	设置用户名和密码	<code>nvSeq[0].name = 'USER'</code> <code>nvSeq[0].valMap.aval =</code> <code><Username>\0<Password>\0</code>

WILDCARD_ONE	对于一个字符匹配的通配符	nvSeq[0].name = 'WILDCARD_ONE' nvSeq[0].valMap.aval = "?"
WILDCARD_ALL	对于零个或多个字符匹配的通配符	nvSeq[0].name = 'WILDCARD_ALL' nvSeq[0].valMap.aval = "*"
WILDCARD_ESC	转义符	nvSeq[0].name = 'WILDCARD_ESC' nvSeq[0].valMap.aval = "\\"

12. 7. 2 AOP_GETPAR

该函数用于获取参数，程序见表 192。
当前可供查询的服务器参数见表 193。

表192 获取参数

GetParRet AOP_GetPar (GetParReq) = 52;
struct SetParReq { AOP_Id envId; /* environ. handle */ AOP_NameV nvSeq<>; /* parameters */ };
struct GetParReq { AOP_Id envId; /* environ. handle */ AOP_NameU nuSeq<> /* parameter name-unit tuples */ };
union GetParRet switch (AOP_Status retState) { case E_NOTOPEN: void; case E_BSS: void; case E_SSS: void; case E_DAC: void; case E_SECURITY: void; case E_MISC: void; default: AOP_NameV nvSeq<>; /* parameters */ };

表193 当前可供查询的服务器参数

参数	数值
USER	用户名
ODSVERSION	支持的界面ODS-API版本（如5.2.0）
WILDCARD_ONE	对于单字符匹配的通配符
WILDCARD_ALL	对于零个或多个字符匹配的通配符
WILDCARD_ESC	转义符
FILE_NOTATION	字符串值'UNC_WIN','UNC_UNIX'或'URL'中之一描述了这样一种句法（带有某些语义），其中外部文件的位置得以指定（详见5.10.6）
FILE_MODE	'ABSOLUTE','SINGLE_VOLUME'或'MULTI_VOLUME'的一个字符串值指定了服

	务器管理外部文件位置的方式（详见5.10.6）
FILE_SYMBOLS	提供ODS服务器已知的，作为字符串列表的（用逗号分开）的所有文件符号名列表。文件符号指定外部文件的根路径（详见5.10.6）
FILE_ROOT	指定外部组件文件的根路径（详见5.10.6）
FILE_ROOT_EXTREF	指定外部参考文件的根路径（详见5.10.6）
细化的文件符号名称	由FILE_SYMBOLS给出的字符串列表的每一个文件符号名是相关变量本身的名称；每一个都指定了外部文件的根路径（详见5.10.6）

12.8 安全服务

随着 ASAM ODS 5.0 版本的使用，一个新的安全问题的处理已经被引入以支持用户、用户群、不同的元素访问权限、属性和/或元素实例。关于 ASAM ODS 安全概念的详细描述见第 5 章。

用户和用户群是标准 API 方法可以访问的应用元素（AOP_Getval, AOP_Putval）。每一个用户都可以被分配到一个或多个用户群。这些 m:n 索引可以用 AOP_SetInstRef 来设置。所有其他服务只能通过下列 Security-API 方法来揭示。

访问权限和直接访问终端必须加以区分。访问权限存储在 ACL 词条目录里（如访问控制结构 AOP_Acs），而直接访问终端存储在 ACL 模板里（如原始权限结构 AOP_Irs）。ACL 模板用来存储“默认权限”，将被复制到新实例的 ACL 词条目录里。

为了管理与安全相关的信息，一组类型定义已经通过安全服务措施被指定使用，见表 194。

表194 安全类型定义

```
typedef long AOP_Rights; /* Used for Rights-Values (bitmasked) */
enum AOP_SetType { /* how to modify bitmasked values */
    SET, /* Set the value to new bits (clears all bits before) */
    ADD, /* Adds the new bits to the existing ones (OR-function) */
    REMOVE /* Removes the bits from the existing value (~AND function)
*/
};
struct AOP_Acs { /* Access control structure */
    AOP_Id      groupId; /* Usergroup Id */
    AOP_ElemId  elemId; /* Application element+instance Id */
    AOP_Name    attrName; /* Application attribute name */
    AOP_Rights  rights; /* rights-value (bitmasked) 5-bits */
};
struct AOP_Irs { /* Initial rights structure */
    AOP_Id      groupId; /* Usergroup Id */
    AOP_ElemId  elemId; /* Application element+instance Id */
    AOP_Rights  rights; /* rights-value (bitmasked) */
    AOP_Id      refAid; /* referencing application element. If set to
                        0, its been used by all elements that refer
                        to it */
};
```


};

AOP_Rights 类型指定了一组访问权限，位码为一“长整型”变量。权限的含义和位掩码在第二章中详细说明。这相当于 ACLI, ACLA, ACLTemplate 等字符实体里的项目权限（详见第 7 章），然而应当注意的是，基础模型将这个项目指定为 16 比特的整数值。

在位码值提供给一个方法时，AOP_SetType 枚举法才指定服务器的预期行为。无论任何时候，访问权限（AOP_Rights）或安全等级总要被设置，就属于这种情况。

（结构化的）AOP_Acs 模本指定一个 ACL 词条目录，相当于 ACLI 和 ACLA 词条目录（见第四章），它们在基础模型中是独立的实体，但此处被组合在同一结构中。

- a) 如果 AOP_Acs 元素识别码类型的变量同时包含应用元素识别码和实例识别码，那么 attrName 就应当为空（它将被服务器忽略），并且这个变量代表了一个 ACLI 数据集（实例安全性）；
- b) 如果 AOP_Acs 元素识别码类型的变量只包含一个应用元素识别码（因此实例识别码为 0），并且 attrName 为空，那么这个变量代表了一个 ACLA 数据集（元素安全性）；
- c) 如果 AOP_Acs 元素识别码只包含一个应用元素识别码（因此实例识别码为 0），那么这个变量代表了一个 ACLA 数据集（属性安全性）。

（结构化的）AOP_Irs 模本指定一个 ACL 模板，这相当于一个 ACL 模板词条目录（见第四章）。应当注意的是，无论是在 ACLI, ACLA 还是在 ACL 模板里，应用元素和索引应用元素（RefAID）都不是被其识别码所指定，而是被其（长很多）的名称指定。

AOP_GetApplInf（）方法可用来查找应用元素名称与应用程序识别码的映射。

12.8.1 AOP_SETSECURITYLEVEL

该函数用于定义一些元素里的安全等级，程序见表 195。

这些模本可以被参数< seclevel>使用（被混合）和定义：

- 1) 元素不安全（< seclevel>=0）；
- 2) 元素安全开启（保护所有元素，0 位设置）；
- 3) 实例安全开启（保护所有实例，1 位设置）；
- 4) 属性安全开启（保护所有属性，2 位设置）。

表195 定义元素安全级别

SetSecurityLevelRet	AOP_SetSecurityLevel (SetSecurityLevelReq) = 69;
struct SetSecurityLevelReq { /* setsecuritylevel-request structure */	
AOP_Id	envId; /* environ. handle */
AOP_Id	applId; /* application element id */
AOP_Flags	seclevel; /* security level (bitmasked) */
AOP_SetType	setType; /* set/add/remove security level bits */
};	
union SetSecurityLevelRet switch (AOP_Status retState) {	
/* select return value structure */	

```
case E_NOTOPEN: void;
case E_BSS: void;
case E_SSS: void;
case E_DAC: void;
case E_SECURITY: void;
case E_MISC: void;
default: AOP_Flags seclevel; /* return security-level */
};
```

12.8.2 AOP_GETSECURITYLEVEL

该函数用于获取特定元素-id 的当前安全等级，程序见表 196。

表196 获取特定运输安全级别

```
GetSecurityLevelRet AOP_GetSecurityLevel (GetSecurityLevelReq) = 70;
struct GetSecurityLevelReq { /* getsecuritylevel-request structure */
    AOP_Id    envId; /* environ. handle */
    AOP_Id    applId; /* application element id */
};
union GetSecurityLevelRet switch (AOP_Status retState) {
    /* select return value structure */

    case E_NOTOPEN: void;
    case E_BSS: void;
    case E_SSS: void;
    case E_DAC: void;
    case E_SECURITY: void;
    case E_MISC: void;
    default: AOP_Flags seclevel; /* return security-level */
};
```

12.8.3 AOP_SETRIGHTS

该函数用于设置某些数据对象和用户组的访问控制权限（ACL），程序见表 197。

表197 设置某些数据对象和用户组的访问权限（ACL）

```
SetRightsRet AOP_SetRights (SetRightsReq) = 61;
struct SetRightsReq { /* setrights-request structure */
    AOP_Id    envId; /* environ. handle */
    AOP_Acs    acl<>; /* ACL */
    AOP_SetType setType; /* set/add/remove bits from rights in
ACL */
};
```

```
struct SetRightsRet {
    AOP_Status    retState;
};
```

12.8.4 AOP_GETRIGHTS

该函数用于获取某些数据对象和用户组的访问控制权限（ACL），程序见表 198。

表198 获取某些数据对象和用户组的访问权限（ACL）

```
GetRightsRet AOP_GetRights (GetRightsReq) = 62;

struct GetRightsReq { /* getrights-request structure */
    AOP_Id envId; /* environ. handle */
    AOP_ElemId elemId<>; /* Application element+instance Ids */
    AOP_Name attrName<>; /* Application attribute names (list of
                           elements + list of attributes at the
                           same time is not allowed) */
};

union GetRightsRet switch (AOP_Status retState) {
    /* select return value structure */

    case E_NOTOPEN: void;
    case E_BSS: void;
    case E_SSS: void;
    case E_DAC: void;
    case E_SECURITY: void;
    case E_MISC: void;
    default: AOP_Acs acl<>; /* ACL will be returned */
};
```

需求结构非常通用，允许获取所有三个安全级别的权限：

- a) 如果列表 attrName<>是空的，则该方法提供与应用元素和/或实例关联的所有 ACL 条目的信息。
列表 elemId<>中的每项说明了一个应用元素（）或一个实例；服务器将收集和返回与它们关联的所有 ACL 条目。
elemId 列表中的项若包含了无效的应用元素或实例标识符，则该项将被忽略。
这只与“元素安全”或“实例安全”的情况有关。
- b) 如果列表 attrName<>是非空的，则该方法提供与一个应用元素的特定应用属性关联的所有 ACL 条目。
这种情况下涉及到的应用元素是列表 elemId<>中第一项说明的内容（该列表中的其它项将被忽略）。
列表 attrName<>中的每项描述了一个应用属性名；服务器将收集和返回与该应用属性关联的

所有 ACL 条目。

列表 `attrName<>` 中的项若包含了所涉及应用元素的无效应用属性名，则该项将被忽略。

- c) 允许考虑“元素安全”和“实例安全”的 ACL 条目的组合需求。
- d) 对“属性安全”的 ACL 条目需求不能与其它两种安全等级组合。

12.8.5 AOP_SETINIRIGHTS

该函数用于定义用于某些数据对象和用户组的 ACL 模板，程序见表 199。

表199 用于某些数据对象和用户组的 ACL 模板

```
SetIniRightsRet AOP_SetIniRights (SetIniRightsReq) = 63;

struct SetIniRightsReq { /* setinirights-request structure */
    AOP_Id          envId; /* environ. handle */
    AOP_Irs         irl<>; /* Initial rights list */
    AOP_SetType     setType; /* set/add/remove bits from rights */
};

struct SetIniRightsRet {
    AOP_Status      retState;
};
```

12.8.6 AOP GETINIRIGHTS

该函数用于获取某些数据对象和用户组的 ACL 模板，程序见表 200。

表200 获取某些数据对象和用户组的 ACL 模板

```
GetIniRightsRet AOP_GetIniRights (GetIniRightsReq) = 64;

struct GetIniRightsReq { /* getinirights-request structure */
    AOP_Id      envId; /* environ. handle */
    AOP_ElemId  elemId<>; /* List of application element+instance
                           Ids */
};

union GetIniRightsRet switch (AOP_Status retState) {
    /* select return value structure */

    case E_NOTOPEN: void;
    case E_BSS: void;
    case E_SSS: void;
    case E_DAC: void;
    case E_SECURITY: void;
    case E_MISC: void;
    default: AOP_Irs irl<>; /* Initial rights list will be returned */
};
```

12.8.7 AOP_SETINIRIGHTSREF

使用该方法，可以将应用关系标记为一个 ACL 引用，程序见表 201。

当插入一个新实例时，可解析该关系的值（对一个元素的引用）且可从被引用的实例获取 ACL 模板。这对于存储数据依赖型 ACL 模板是有用的，可在插入时引用它们。这方面的详细描述（属于 ASAM ODS 安全概念的组成内容）参见第 5 章。

该方法允许将实体 InitialRightsAt 的数据集提交给服务器（参见第 7 章的描述）。

表201 将应用关系标记为一个 ACL 引用

```
SetIniRightsRefRet AOP_SetIniRightsRef (SetIniRightsRefReq) = 65;
struct SetIniRightsRefReq { /* setinirightsref-request structure */
    AOP_Id    envId; /* environ. handle */
    AOP_Id    applId; /* application element Id */
    AOP_Name  refAttr; /* Reference Attribute */
    AOP_Flags set; /* use (1) or don't use (0) thereference-attribute
                    for ACLtemplates */
};
union SetIniRightsRefRet switch (AOP_Status retState) {
    /* select return value structure */
    case E_NOTOPEN: void;
    case E_BSS: void;
    case E_SSS: void;
    case E_DAC: void;
    case E_SECURITY: void;
    case E_MISC: void;
    default: AOP_Flags set; /* set-value will be returned */
};
```

12.8.8 AOP_GETINIRIGHTSREF

该方法返回被标记为 ACL 引用的引用关系列表，程序见表 202。

表202 返回被标记为 ACL 引用的引用关系列表

```
GetIniRightsRefRet AOP_GetIniRightsRef (GetIniRightsRefReq) = 66;
struct GetIniRightsRefReq { /* getinirightsref-request structure */
    AOP_Id envId; /* environ. handle */
    AOP_Id applId; /* Application element Id */
};
union GetIniRightsRefRet switch (AOP_Status retState) {
    /* select return value structure */
    case E_NOTOPEN: void;
    case E_BSS: void;
```

```
case E_SSS: void;
case E_DAC: void;
case E_SECURITY: void;
case E_MISC: void;
default: AOP_Name attr<>; /* attribute-list will be returned */
};
```

12.8.9 AOP_SETCURRENTINIRIGHTS

使用该方法，客户端可以定义会话特定的 ACL 模板，程序见表 203。这种调用将定义一个 ACL 模板集合，用于在当前会话之后的插入方法（参数<set>必须等于 1）。如果参数<set>为 0（假），则前面定义的会话特定 ACL 模板将不再用于插入方法中。

表203 定义会话特定的 ACL 模板

```
SetCurrentIniRightsRet
AOP_SetCurrentIniRights (SetCurrentIniRightsReq) = 67;
struct SetCurrentIniRightsReq { /* setcurrentinirights-request
                                structure */
    AOP_Id    envId; /* environ. handle */
    AOP_Irs    irl<>; /* Initial ACL to use in subsequentcreation of
                        new instances */
    AOP_Flags set; /* Use (1) this ACL or don't use (0) this ACL for
further                insertsanyomore */
};
union SetCurrentIniRightsRet switch (AOP_Status retState) {
    /* select return value structure */
    case E_NOTOPEN: void;
    case E_BSS: void;
    case E_SSS: void;
    case E_DAC: void;
    case E_SECURITY: void;
    case E_MISC: void;
    default: AOP_Flags set; /* set-value will be returned */
};
```

12.8.10 AOP_SETPASSWORD

该函数为用户设置一个新密码，程序见表 204。

如果当前登录的用户是超级用户组的成员，则他不需要在<oldpwd>中提供当前的密码。“普通”用户必须在<oldpwd>中提供当前的密码，他们可以在<newpwd>中修改自己的密码。“普通”用户不能修改其它用户的密码，即使他们在<oldpwd>中提供了（其他用户）当前的正确密码。

表204 为用户设置一个新密码

```

SetPasswordRet AOP_SetPassword (SetPasswordReq) = 68;

struct SetPasswordReq { /* setpassword-request structure */
    AOP_Id      envId; /* environ. handle */
    AOP_Id      userid; /* user-id to set the password for */
    AOP_String   oldpwd; /* current password */
    AOP_String   newpwd; /* new password */
};

union SetPasswordRet switch (AOP_Status retState) {
    /* select return value structure */

    case E_NOTOPEN: void;
    case E_BSS: void;
    case E_SSS: void;
    case E_DAC: void;
    case E_SECURITY: void;
    case E_MISC: void;
    default: AOP_Flags ok; /* ok=1 if password changed otherwise
                           ok=0 */
};

```

12.8.11 用户验证

当打开一个会话时，用户需要用户名和密码登录。该用户名和密码的验证是通过检查会话参数 **USER** 来实现的。因此客户端应用必须调用 **AOP_SetPar()**，其参数 **USER** 包含<username>和<password>两个值（字符串列表，用空字符分隔，见 **AOP_SetPar**）。客户端应用提供的密码必须是非加密的，服务器端将对密码进行加密处理（和 **ODS 3.0** 版本兼容）。

服务器将检查信息是否有效。如果信息无效，则返回错误消息。

对于在一个相同的打开会话中客户端发出的进一步请求，都无需再次提供用户名和密码。

12.8.12 ODS 中的密码加密（RPC）

密码加密是用户验证的关键部分，因为绝不能用明码方式（非加密方式）保存密码或者采用允许密码解密的加密算法。**ASAM ODS** 使用的加密算法是 **MD5**（**Ronald L. Rivest** 开发的消息摘要算法）。

MD5 算法能基于明码计算得到加密的密码，是个开放且易于使用的算法。对于加密过的算法，几乎不可能：

- a) 反转算法找出明码；
- b) 找出另一个具有相同加密密码的密码。

12.9 错误处理

有两种错误：**RPC** 失效（例如连接中断、使用非兼容请求导致无法解码参数、超时、.....）和应用级失效（例如丢失了强制属性、未找到应用元素、...）。当客户端得到一个 **NULL** 指针而不是返回的数

据时可检查出一个错误。RPC 的错误原因可通过 `aop_geterr` 确定。目前使用的有限数量的联合识别符，将来可扩展。程序见表 205。

表205 错误处理程序

```
enum AOP_Status { /* error codes */
    S_OK = 0, /* OK */
    /* version 2 states */
    E_BSS = 1, /* basic system error state */
    W_BSS = 2, /* basic system warning state */
    E_SSS = 3, /* --"-- error */
    W_SSS = 4, /* subsystem (e.g. FORINT, RDBMS) warning */
    /* version 1 states and version 2 states not
       yet moved to new state interface; states
       which haven't been used are removed */
    E_NOTOPEN = 13, /* not yet open, environment */
    E_DAC = 21, /* data access error */
    /* for detail see return structure */
    /* Version 5.0 states */
    E_SECURITY= 41, /* security violation */
    E_MISC = 99 /* to be defined */
};
```

12.10 RPC-API 特定的方面

12.10.1 序列表达的限制

RPC-API 不能完全支持 `seq_rep_enum` 中所列的类型，只支持以下类型：

- a) `explicit` 显式的；
- b) `implicit_constant` 隐式常数；
- c) `implicit_linear` 隐式线型；
- d) `implicit_saw` 隐式观察。

这是由于只有两个接口参数可用于在隐式和显式或者常数、线型和观察之间选择。

12.10.2 继承概念与 RPC-API

RPC-API 支持机床概念。然而当使用 RPC-API 方法时需要考虑下列特定的行为，以访问超类或子类应用元素的实例。

12.10.2.1 通过 `AOP_GETVALE` 实现读取访问

要完全支持继承只能借助于 `AOP_GetValE()`，该函数提供了返回多个部分（结果集合）中结果的可能性。结果集合是必要的，因为属性列表依赖于实例类型，要为每个实例类型创建一个独立的、同类的结果集合。

该接口函数的处理根据是访问一个超类还是子类而有所不同。

a) 子类

要访问一个子类，则超类和子类的所有属性都应当是可用的。这意味着如果报告列表是空的 (Select *), 也要返回该子类实例的所有属性。

b) 超类

当访问超类和子类时，结果可由多个部分组成。如果 AOP_GetValE () 的请求结构说明了超类的应用元素 ID 和一个或多个子类或来自超类及报告列表中一个或多个子类的属性的应用元素 ID，则表示超类和子类被访问了。对于结果中出现的每个类，要返回一个单独的结果集合。根据实例类型要将实例组合进入同类结果集合中。

如果在请求结构中只说明了超类或超类属性的应用元素 ID，则只能访问超类的信息。然而所有实例都是可见的，包括属于一个子类的实例。如果结果仅限于超类的实例，则需要显式说明所作的选择 (派生自基础属性 “objecttype” 的应用属性=超类的应用元素 ID)。

12.10.2.2 通过 AOP_GETVAL 实现读取访问

当访问一个表达超类的应用元素 (通过使用 AOP_GetVal ()) 时，只可使用相关应用元素的属性。然而对于访问一个子类而言，所有属性 (来自超类和子类) 都是可用的。该函数只能返回被访问的应用元素信息。当访问超类时，所有实例是可见的，也包括那些属于子类的实例。就像 AOP_GETVALE 函数显式说明一个选择准则一样，可将视图限定为只有超类的实例可访问。

12.10.2.3 写访问

当进行写操作时，必须表明给定实例类型的应用元素 ID。服务器将在相关应用元素、超类与子类的数据库表中分配属性，并在超类的类型属性 (名为 “objecttype” 的基础属性) 中填入合适的实例类型 (AID 类)。实例的 ID 是通过超类序列来确定的，适用于超类部分和子类部分。

12.11 调用序列示例

示例：该例说明客户端如何与服务器对话，为了简化，省去了参数。

```
// Open Session
AOP_Openenv      //returns Environment-Id

// Get Metainformation (elements, attributes, references)
AOP_GetApplInf   //returns MetaInformation

// Do some work with attributes in elements...
AOP_GetVal       //returns attribute-values (see also AOP_GetValE)
AOP_PutVal       //modify, insert or delete attribute-values

// Do some work with measurement values (partial matrixes)
AOP_GetValAttr   //returns partial matrix structure (attributes)
```

```
AOP_GetValVal    //returns partial matrix
AOP_PutValVal    //modify, insert or delete values in partial matrix

//Close Session
AOP_CloseEnv     //Closes the session
```

12.12 ASAM ODS RPC-API IDL

示例：ASAM ODS RPC-API IDL 程序示例。

```
/*
 *
 * ASAM - ODS 5.2.0 client server ONC RPC interface definition
 *
 *
 * 2006/09/21 Cleanup by Maintenance Group
 * 2003/06/02 Added AoModelMapper, AoLog Gerald Sammer
 *
 ****
 */
/*
 * The interface-definition is based on ONC. The necessary tools for
 * creation of C-headerfiles, client and server stubs and conversion
 * functions (xdr) is free available (RPCGEN)
 *
 * For any method available thru the protocol level (client server)
 * the interface is defined by setting up request and return
 * structures.
 *
 * Version support
 * -----
 * This server supports version 3.0, 3.2 and ODS 5.2.0/RPC
 */
/* ===== platform adaptions
 * adapt aods.h for OS - dependencies to make aods.x more portable
 * Note that rpcgen uses the local cpp (preprocessor)
 */
%
```

```

/* -----begin of section defined by aods.x -----
% */
#ifdef RPC_HDR
#define _AODS_INTERFACE_VERSION "$Revision: 3.2 $"
#ifdef _AODS_H
#define _AODS_H
#endif
%
#ifdef _WIN32
/* remove definitions used by Visual C++ on NT
% * and add winsock.h if not using GNU on WIN32
% */
#include <rpc/rpc.h>
#ifndef __CYGWIN32__
#include <winsock.h>
#undef S_OK
#undef E_NOTIMPL
#endif
#endif

#ifdef hpux
/* Do not edit, automatically created by rpcgen
% * the HP-UX rpcgen doesnt include rpc.h into aods.h but into
% * the .c files created -> force to put it into aods.h if not yet done
% * since VMS comes without rpcgen hpux is used for VMS too
% * but dont forget to remove the include <rpc/rpc.h> line from aods_xdr.c
% * and stubs manually (or by awk) when using those files on VMS
% */
#ifdef vms
#ifdef _RPCXDR_LOADED
#include <ucx$rpcxdr.h>
#endif
#else
#ifdef _STDLIB_INCLUDED
#include <stdlib.h>
#endif
#endif

```

```

#ifdef hpux10
#ifndef _RPC_RPC_INCLUDED
else
#ifndef _RPC_TYPES_INCLUDED
#endif
#include <rpc/rpc.h>
#endif

#endif
#endif
%
%extern unsigned long rpc_number; /* RPC program number settable */
#endif
#ifdef RPC_SVC
%
%/* include authentication handling on server side
%*/
#include "aods_auth.h"
#endif
#ifdef RPC_CLNT
%
%/* for some systems where rpcgen of foreign system is used struct timeval
% * may be undefined for TIMEOUT default definition
% */
#include <time.h>
%
%/* nonstandard emulation of ListEnv
%*/
#define AOP_ListEnv getenv ("AODSKNOWNENV")
%
#endif
%/* ----- end of aods.x defined section -----
% */
%

/* ===== interface definition */
/* Some AODS data types

```

```

* Any module may include the protocoll compiler resulting
* header file and use those constants
*/

const MAXAOPNAMLEN = 128;
typedef string AOP_Name<MAXAOPNAMLEN>;
typedef AOP_Name AOP_NameSeq<>;
const MAXSTRINGLEN = 2000;          /* maximum string length */
typedef string AOP_String<MAXSTRINGLEN>;
const MAXBSTREAMLEN = 1048576;      /* maximum byte stream length */
typedef unsigned char bstream<MAXBSTREAMLEN>;
typedef long AOP_Id;                 /* Identifiers/Handles always int */
typedef long AOP_Flags;              /* Used for Flags */
typedef short AOP_ValFlags<>;        /* Used for Flags to a Value */

/* ===== return states */
enum AOP_Status { /* error codes */
    S_OK = 0,          /* OK */
                        /* version 2 states */
    E_BSS = 1,         /* basic system error state */
    W_BSS = 2,         /* basic system warning state */
    E_SSS = 3,         /* --"-- error */
    W_SSS = 4,         /* subsystem (e.g. FORINT, RDBMS) warning */
                        /* version 1 states and version 2 states not yet
                        moved to new state interface states which havn't
                        been used are removed */
    E_NOTOPEN = 13,    /* not yet open, environment */
    E_DAC = 21,        /* data access error */
                        /* for detail see return structure */
    E_SECURITY = 41,   /* security violation */
    E_LICENSE = 42,    /* license error */
    E_MISC = 99        /* to be defined */
};

enum AOP_DAC_Status { /* subcodes used in version 1, replaced by x_SSS */
    DACINVATTR,        /* referenced attribute doesnt exist */
    DACNOPRIV,         /* privilege violation */
    DACMANDATTR,       /* mandatory attribute missing */
    DACCONSTRAINT,     /* constraint conflict */

```

```

DACOTHER

};

/* ===== general type definitions */
enum AOP_DataType { /* known data types */
    DT_RESERVED = 0, /* 0 is called DT_UNKNOWN in the other chapters of
                       the ASAM ODS specification */
    DT_STRING = 1, /* string */
    DT_SHORT = 2, /* short integer */
    DT_FLOAT = 3, /* float */
    DT_BYTE = 5, /* byte */
    DT_LONG = 6, /* long integer */
    DT_DOUBLE = 7, /* double */
    DT_DATE = 10, /* date format */
    DT_BYTESTR = 11, /* byte string */
    /* blob holds beside the binary data (no architecture adaption)
     * a blob type which may be used to tell the client who may
     * interpret those data
     */
    DT_BLOB = 12 /* includes blob identifier */
};

/* These types are used to allow general purpose clients to interpret the
 * data
 */
enum AOP_BaseElem { /* Basic Entities */
    BE_RESERVED = 0, /* unknown base element */
    BE_ENV = 1, /* Environment */
    BE_TST = 2, /* Subtest */
    BE_MEA = 3, /* Measurement */
    BE_MEQ = 4, /* measured quantity */
    BE_DNA = 11, /* quantity name */
    BE_DNG = 12, /* quantity group */
    BE_DUN = 13, /* unit */
    BE_DUG = 14, /* unit group */
    BE_DIM = 15, /* physical dimension */
    BE_UUT = 21, /* UnitunderTest */
    BE_UUP = 22, /* --"-- part */
    BE_INS = 23, /* TestEquipment */

```

```

BE_INP = 24, /* --"-- part */
BE_SEQ = 25, /* Test sequence */
BE_SEP = 26, /* --"-- part */
BE_USR = 34, /* user */
BE_UGR = 35, /* user group */
BE_TEST = 36, /* Test */
BE_TESTDEVICE = 37, /* Testdevice */
BE_SUBMATRIX = 38, /* AoSubmatrix */
BE_LOCALCOLUMN = 39, /* AoLocalColumn */
BE_EXTCOMP = 40, /* AoExternalComponent */
BE_LOG = 43, /* AoLog */
BE_PARAMETER = 44, /* AoParameter */
BE_PARAMATERSET = 45, /* AoParameterSet */
BE_NAMEMAP = 46, /* AoNameMap */
BE_ATTRIBUTEMAP = 47 /* AoAttributeMap */
};

enum AOP_SelOpcode { /* Type of selection */
    SOMATCH, /* match == */
    SOUMATCH, /* unmatched != */
    SORANGE, /* range < x > */
    SOLESS, /* less < */
    SOGREATER, /* greater > */
    SOLESSEQ, /* less equal <= */
    SOGREATEREQ, /* greater equal >= */
    SOINSET, /* within set in */
    SONOTINSET, /* not within set !in */
    SOORDER, /* order by */
    SOGROUP, /* group by */
    SOINSENSITIVE, /* insensitive search */
    SOLIKE, /* like */
    SONOTLIKE, /* not like */
    SONULL, /* is NULL */
    SONOTNULL, /* is NOT NULL */
    SOOPAND, /* operator AND */
    SOOPOR, /* operator OR */
    SOOPNOT, /* operator NOT */
    SOOPBOPEN, /* operator bracket open ( */

```

```

        SOOPBCLOSE /* operator bracket close ) */
};
/* see paper about multidimensional measured data arrays
 * for explanation on following flag
 */
enum AOP_IndepFl { /* Flag for independent channel */
    INDNONE, /* no information (dependend) */
    INDEPEND, /* independent */
    SCALING /* scales another column */
};
/* implicit means that the value may be calculated by a predefined
 * procedure. For implicit columns one (constant), two (start, increment)
 * or three values (start, increment, end) are stored within the column
 */
enum AOP_ImplFl { /* Flag for implicit storage */
    IMPLNONE, /* explicit */
    IMPLICIT /* implicit */
};
enum AOP_SortDir { /* Sort direction */
    ASCENDING, /* lower to higher */
    DESCENDING /* higher to lower */
};
enum AOP_JoinTyp { /* way of joining partial mat. */
    JOIN, /* exact matches only */
    MERGE /* all */
};

/* =====
 * General structures
 */
/* ----- ElementId
 */
struct AOP_ElemId { /* Unique identification of instance */
    AOP_Id aid; /* application element id */
    AOP_Id iid; /* instance id */
};
typedef AOP_ElemId AOP_ElemIdSeq<>;

```



```

/* ----- ValMap
    This part describes the structures of an value map, this structure
    is common for all value accesses (Application Element Attribute Values
    and Local Column Values)
    Note that a string vector is a NULL separated byte array.
    The number of elements (strings) is equal to the number of NULL characters
    within the array. Therefore length is the number of bytes and not number
    of elements.
    BYTESTR is similar to STRING (each element may hold multiple characters)
    except that instead of NULL-termination the count is included to allow
    binary data.
    Blob may be used only at places where no multiple instances occur.
*/
struct Blob {
    char bhdr<>;
    unsigned char bval<>;
};
union AOP_ValMap switch (AOP_DataType dtyp) {
    case DT_STRING:
        char aval<>; /* strings */
    case DT_SHORT:
        short sval<>; /* short integers */
    case DT_FLOAT:
        float fval<>; /* floats */
    case DT_LONG:
        long lval<>; /* long integers */
    case DT_DOUBLE:
        double dval<>; /* double floats */
    case DT_BYTE:
        unsigned char bval<>; /* single bytes */
    case DT_DATE:
        char tval<>; /* date/time */
    case DT_BYTESTR:
        bstream cval<>; /* counted byte streams */
    case DT_BLOB:
        Blob blob; /* never as array !!! */
}

```

```

        default:
            void;

};

/* ----- Name + Unit
*/

struct AOP_NameU {
    AOP_Name name;
    AOP_Id unitId;
};

typedef AOP_NameU NUSeq<>;

/* ----- Name + Unit + Value (s)
*/

struct AOP_NameV {
    AOP_Name name;
    AOP_Id unitId;
    AOP_ValMap valMap;
};

typedef AOP_NameV NVSeq<>;

/* =====
* Request and return structure definition section
*/

/* ----- list environements
To emulate Master server (or service information brooker) an environnement
variable on client side should be called (see above) .
*/

/* ----- open environnement
* Note that current servers are restricted to one environnement a time
* Argument usage is free for implementors-
* RDBMS implementation uses DBLOGON to (optionally) specify the
* database logon string to be used.
*/

struct OpenEnvReq { /* to open env. */
    AOP_NameV nvSeq<>; /* environnement open arguments */
};

union OpenEnvRet switch (AOP_Status retState) {
    /* return value structure */

```

```

    case E_BSS: void;
    case E_SSS: void;
    case E_SECURITY: void;
    case E_MISC: void;
    case E_LICENSE: void;
    default: AOP_Id envId; /* descriptor/handle/id whatever */
};
/* ----- close environnement
*/
struct CloseEnvReq { /* to close open env. */
    AOP_Id envId;
};
struct CloseEnvRet {
    AOP_Status retState;
};

/* ----- structure query
Whole basic structure has to be returned including all relations.
This is a list of application elements (Identifier = AID) together with
application name and type (basic element AOP_BasElem)
In an additional list the relations (AOP_BasElem - pairs)
will be returned
*/
struct GetApplInfReq {
    AOP_Id envId; /* environ. handle */
};
struct ApplInfSeq { /* application element infos */
    AOP_Id aiAId; /* application element id */
    AOP_BasElem aiBId; /* basic element Id */
    AOP_Name aiName; /* application element name */
};
struct ApplRelSeq { /* application relations infos */
    AOP_Id arAId1; /* from (0,1) resp. (n) */
    AOP_Id arAId2; /* to (n) resp. (m) */
    long arRefNr; /* attr number, 0 for n:m */
    AOP_Name arName; /* reference name */
    AOP_Flags arConstr; /* Constraint (e.g. not NULL) */

```

```

};

struct ApplInf {
    ApplInfSeq aiSeq<>; /* sequence of appelem info */
    ApplRelSeq arSeq<>; /* sequence of appref info */
};

union GetApplInfRet switch (AOP_Status retState) {
    case E_NOTOPEN: void;
    case E_BSS: void;
    case E_SSS: void;
    case E_DAC: void;
    case E_SECURITY: void;
    case E_MISC: void;
    default: ApplInf applInf;
};

/* ----- references (instance - level)
This query tries to get instances related to a given one thru
    allowed (see ApplInf) relations
*/

struct GetInstRefReq {
    AOP_Id envId; /* environ. handle */
    AOP_ElemId elemId; /* instance */
    AOP_Name refName;
    AOP_Id aId; /* of type */
};

union GetInstRefRet switch (AOP_Status retState) {
    /* return value structure */

    case E_NOTOPEN: void;
    case E_BSS: void;
    case E_SSS: void;
    case E_DAC: void;
    case E_SECURITY: void;
    case E_MISC: void;
    default: AOP_ElemId elemList<>;
};

/* ---- get attributes -----

```

Structure to describe attributes of application elements.
 Applies to any entity also in many cases aBName matches aAName
 due to attrAName unchangeable on application level
 The request specifies the elementId. The prototype uses only AId part
 of elementId (no instance attributes available)

```

*/
struct AttrSeq { /* sequence of attributes */
    AOP_Name aBName; /* basic name */
                    /* 0 if not basic */
    AOP_Name aAName; /* appl. name */
    AOP_DataType aDataType; /* attribute data type */
    AOP_Id aUnit; /* unit if global defined */
};

struct AttrInf { /* attribute info including header */
    AOP_BasElem aBId; /* basic element Id */
    AOP_Id aAId; /* application element Id */
    AttrSeq aSeq<>;
};

struct GetAttrReq { /* request for attribute list */
    AOP_Id envId; /* environ. handle */
    AOP_ElemId elemId; /* application element */
};

union GetAttrRet switch (AOP_Status retState) {
    /* return value structure */

    case E_NOTOPEN: void;
    case E_BSS: void;
    case E_SSS: void;
    case E_DAC: void;
    case E_SECURITY: void;
    case E_MISC: void;
    default: AttrInf attrInf; /* including entity */
};

/* ----- get values for application elements
  Query structure, allows to specify ranges and patterns for queries.
  Values are always arrays. This allows i.e. on selection to pass 2 or
  more values for e.g.:  range from - to

```

To cover the AODS requirement of making data access by explicitly specifying instance id (direct access to single element) it is necessary to define the selection elements AOP_NameS with

```

    name = <Idattr>
    selOpcode = SOMATCH
    value.AOP_ValMap_u.lval.lval[0] = <iid>
    IdAttr is the first aBName reportet using GetAttr
*/
struct AOP_NameS { /* by value select */
    AOP_Name name; /* attribute name */
    AOP_ValMap valMap; /* range or pattern */
    AOP_SelOpcode selOpcode; /* type of selection */
};
typedef AOP_NameS NSSeq<>;
struct GetValReq { /* selection request structure */
    AOP_Id envId; /* environ. handle */
    AOP_Id applId; /* application element Id (from) */
    AOP_NameU nuSeq<>; /* sequence of to be reported attr's */
    AOP_NameS nsSeq<>; /* sequence of select fields */
    AOP_ElemId elemId; /* child/parent reference (where) */
    AOP_Name refName; /* named reference identifier */
};
union GetValRet switch (AOP_Status retState) {
    /* select return value structure */

    case E_NOTOPEN: void;
    case E_BSS: void;
    case E_SSS: void;
    case E_DAC: void;
    case E_SECURITY: void;
    case E_MISC: void;
    default: AOP_NameV nvSeq<>; /* sequence of resulting attributes */
};

/* ----- enhanced get values for application elements
Enhanced Query structure, allows
- joins of application elements

```

- to specify ranges and patterns
(Values are always arrays. This allows i.e. on selection to pass 2 or more values for e.g.: in/notin, range from - to)
- to group selections with AND, OR and brackets (open, close)
- to sort the result (order by)
- to specify wildcards (including escape character)
- partial access of the result (e.g. first 100, next 100, ..)

To cover the AODS requirement of making data access by explicitly specifying instance id (direct access to single element) it is necessary to define the selection elements AOP_AIDNameS with

```
selOpcode = SOMATCH
AID = <application element ID>
name = <Idattr>
value.AOP_ValMap_u.lval.lval[0] = <iid>
```

IdAttr is the first aBName reportet using GetAttr

```
*/
/* ----- AID + Name + Unit
*/
struct AOP_AIDNameU {
    AOP_Id aid; /* application element ID */
    AOP_Name name; /* attribute name */
    AOP_Id unitId; /* requested unit */
};
typedef AOP_AIDNameU ANUSeq<>;
/* ----- AID + Name + Unit + Value (s)
*/
struct AOP_AIDNameV {
    AOP_Id aid; /* application element ID */
    AOP_Name name; /* attribute name */
    AOP_Id unitId; /* requested unit */
    AOP_ValMap valMap; /* values */
};
typedef AOP_AIDNameV ANVSeq<>;
typedef ANVSeq ANVSeqSeq<>;
enum AOP_ReportListType { /* type of report list */
    RLCOMPLETE, /* all attributes */
    RLSELECTIVE /* given list of attributes */
}
```

```

};

union AOP_ReportList switch (AOP_ReportListType rlType) {
    case RLCOMPLETE:
        AOP_Id aidSeq<>; /* list of application elements - all attributes
            will be reported */
    case RLSELECTIVE:
        AOP_AIDNameU anuSeq<>; /* list of attribute to be reported */
    default: void;
};

struct AOP_AIDNames { /* by value select */
    AOP_SelOpcode selOpcode; /* type of selection */
    AOP_Id aid; /* application element ID */
    AOP_Name name; /* attribute name */
    AOP_ValMap valMap; /* range or pattern */
    AOP_SortDir dir; /* sort order */
};

typedef AOP_AIDNames ANSSeq<>;

struct AOP_RefDef { /* reference definition */
    AOP_Id fromAid; /* start point (AID) of reference */
    AOP_ElemId elemId; /* child/parent reference (where) */
    AOP_Name refName; /* named reference identifier */
};

enum AOP_JoinType { /* join type */
    JTDEFAULT, /* force inner join */
    JTOUTER /* force outer join on destination AID */
};

struct AOP_JoinDef { /* join definition */
    AOP_Id fromAid; /* start point (AID) of reference */
    AOP_Id toAid; /* destination (AID) */
    AOP_Name refName; /* named reference identifier */
    AOP_JoinType joinType; /* join type */
};

typedef AOP_JoinDef JDSeq<>;

enum AOP_ReqType { /* type of request */
    RTDEFAULT, /* force query and close cursor */
    RTOPEN, /* force query and keep cursor open */
};

```



```

    RTCONTINUE, /* continue reading cursor */
    RTCLOSE /* close open cursor */
};

struct GetValEReq { /* enhanced query request structure */
    AOP_Id envId; /* environ. handle */
    AOP_ReqType reqType; /* request type */
    AOP_ReportList repList; /* definition of report list */
    AOP_AIDNameS nsSeq<>; /* sequence of conditions */
    AOP_JoinDef jdSeq<>; /* sequence of join definitions */
    AOP_RefDef refDef; /* reference definition */
    long rowCnt; /* number of requested rows- rowCnt<=0
                  report all rows */
};

union GetValERet switch (AOP_Status retState) {
    /* select return value structure */

    case E_NOTOPEN: void;
    case E_BSS: void;
    case E_SSS: void;
    case E_DAC: void;
    case E_SECURITY: void;
    case E_MISC: void;
    default: ANVSeq result<>; /* sequence of resulting attribute sequences
                               */
};

/* ----- get instance attribute information/values
The result depends on the values of elemId. If only elemId.aid is set,
AOP_GetInstAttr will return information (name, data type and unit but
not the values) about all InstanceAttributes available for specified
application element.
If elemId referencing one instance (elemId.aid and elemId.iid are valid)
all InstanceAttributes belonging to the specified instance will be
returned.
*/

struct GetInstAttrReq { /* selection request structure */
    AOP_Id envId; /* environ. handle */
    AOP_ElemId elemId; /* application element Id */

```

```

};

union GetInstAttrRet switch (AOP_Status retState) {
    /* select return value structure */

    case E_NOTOPEN: void;
    case E_BSS: void;
    case E_SSS: void;
    case E_DAC: void;
    case E_SECURITY: void;
    case E_MISC: void;
    default: AOP_NameV nvSeq<>; /* sequence of resulting attributes */
};

/* ----- insert/update/delete application element
*/

struct PutValReq { /* update request structure */
    AOP_Id envId; /* environ. handle */
    AOP_ElemId elemId; /* application element Id */
    AOP_NameV nvSeq<>; /* sequence of values */
};

union PutValRet switch (AOP_Status retState) {
    /* select return value structure */

    case E_NOTOPEN: void;
    case E_BSS: void;
    case E_SSS: void;
    case E_DAC: void;
    case E_SECURITY: void;
    case E_MISC: void;
    default: AOP_Id iid; /* storage instance id */
};

/* ----- for getting partial matrix info
*/

struct ValInfSeq { /* partial matrix interrelations */
    AOP_Id pmatId;
    AOP_Name pmatName; /* GIDAS requirement (sprintf of pmatId) */
    AOP_Name presortNames <>; /* columns to sort in pmat */
    AOP_SortDir presortDir; /* sortdirection */
};

```

```

    AOP_Name joinNames <>; /* join columns */
    AOP_JoinTyp joinTyp;
    AOP_Name postsortNames <>; /* columns to sort on result */
    AOP_SortDir postsortTyp;
};

typedef ValInfSeq ValInfo<>;
struct GetValInfReq {
    AOP_Id envId;
    AOP_ElemId elemId; /* aid has to be measurement */
};

union GetValInfRet switch (AOP_Status retState) {
    /* info fetch return value structure */

    case E_NOTOPEN: void;
    case E_BSS: void;
    case E_SSS: void;
    case E_DAC: void;
    case E_SECURITY: void;
    case E_MISC: void;
    default: ValInfSeq iSeq<>; /* sequence of info's */
};

/* ----- get information on measured quantity/val
The GetAttr/GetVal methods apply to the "general purpose" application
elements and not to the area of measurementpointnumbers, partial
matrix. , measured quantity, ... (see AODS-datamodel) .
For this area AODS uses methods with applicationspecific behaviour.
This method list measured quantity names and some attributes of them for
one partial matrix.
A list of partial matrices available may be fetch using GetValInf.
*/

struct ValAttrSeq { /* linked list of attributes */
    AOP_Name name; /* appl. structure name */
    AOP_Id qtyId; /* reference to dict. (0 allowed) */
    AOP_Id unitId; /* UnitId */
    AOP_IndepFl indepFl; /* independent flag */
    AOP_ImplFl implFl; /* implicit flag */
    AOP_DataType dataType; /* attribute data type */

```

```

};
struct ValAttr { /* measurement attribute cursor list */
    AOP_Id pmatId; /* PMatId */
    long numPnt; /* number of measurement point in partial matrix */
                /* should be extended for min/max */
    ValAttrSeq vaSeq<>; /* attribute sequence */
};
struct GetValAttrReq { /* request for attribute list */
    AOP_Id envId; /* environ. handle */
    AOP_ElemId elemId; /* measurement id */
    AOP_Id pmatId; /* partial matrix id (0 = all) */
};
union GetValAttrRet switch (AOP_Status retState) {
                /* return value structure */

    case E_NOTOPEN: void;
    case E_BSS: void;
    case E_SSS: void;
    case E_DAC: void;
    case E_SECURITY: void;
    case E_MISC: void;
    default: ValAttr valAttr<>; /* column info for 1 or more PMat's */
};
/* ----- partial matrix data (values)
    The range of data is selectable by specifying the column(s) (vrName)
    and the local measurement point number(s) (mPntNum mPntCnt)
*/
struct ValValSeq { /* linked list of reported vectors */
    AOP_Name name; /* MQ Name */
    AOP_Id qtyId; /* dictionary Id (ref) */
    AOP_Id unitId; /* unit Id */
    AOP_IndepFl indepFl; /* independent flag */
    AOP_ImplFl implFl; /* implicit (versus explicit) */
    AOP_ValMap valMap; /* vector for values */
    AOP_ValFlags valFlags; /* vector for values flags */
};
/* the partial matrix now holds 2 numbers to be able to preserve
* origin of data

```

```

*/
struct ValVal { /* local column values */
    AOP_Id pmatId; /* pMatId (0 .. yet undef) */
    long mPntBase; /* local point number base */
    long mPntCnt; /* # local points */
    ValValSeq vvSeq<>; /* measurement value list */
};

struct GetValValReq { /* request for measurement results */
    AOP_Id envId; /* environ. handle */
    AOP_ElemId elemId; /* elementId, bid to elemId.aid has to be BE_MEA */
    AOP_Id pmatId; /* partial matrix number */
    long mPntBase; /* from local measurement point number */
    long mPntCnt; /* number of measurement points requested */
    AOP_NameU nuSeq<>; /* sequence of attributes to report */
};

union GetValValRet switch (AOP_Status retState) {
    /* select return value structure */

    case E_NOTOPEN: void;
    case E_BSS: void;
    case E_SSS: void;
    case E_DAC: void;
    case E_SECURITY: void;
    case E_MISC: void;
    default: ValVal valVal<>; /* partial matrices */
};

/* update/write/delete partial matrix -----
 * usage: See programmers guide
 */

struct PutValValReq { /* request for measurement results */
    AOP_Id envId; /* environ. handle */
    AOP_ElemId elemId; /* elementId, bid to elemId.aid has to be BE_MEA */
    ValVal valVal<>; /* sequence of headers + columns */
    /* (only one PMat for now per call) */
};

union PutValValRet switch (AOP_Status retState) {
    /* select return value structure */

    case E_NOTOPEN: void;

```

```

    case E_BSS: void;
    case E_SSS: void;
    case E_DAC: void;
    case E_SECURITY: void;
    case E_MISC: void;
    default: AOP_Id pmatId; /* pmatid given by server */
};

/* =====
* Connect two instances (n:m)
*/
struct SetInstRefReq {
    AOP_Id envId; /* environ. handle */
    AOP_ElemId elemId1; /* one element */
    AOP_ElemId elemId2; /* the other */
    AOP_Name refName;
    long onoff; /* insert or remove */
};
struct SetInstRefRet {
    AOP_Status retState;
};
/* =====
* Parametrization of ASAM services
* This method is introduced to have a general purpose method to
* influence behaviour of server by client including integration
* of proprietary functions
*/
struct SetParReq {
    AOP_Id envId; /* environ. handle */
    AOP_NameV nvSeq<>; /* parameters */
};
struct SetParRet {
    AOP_Status retState;
};
struct GetParReq {
    AOP_Id envId; /* environ. handle */
    AOP_NameU nuSeq<>; /* parameter name-unit tuples */

```

```

};

union GetParRet switch (AOP_Status retState) {
    case E_NOTOPEN: void;
    case E_BSS: void;
    case E_SSS: void;
    case E_DAC: void;
    case E_SECURITY: void;
    case E_MISC: void;
    default: AOP_NameV nvSeq<>; /* parameters */
};

/* extended error information
*/

struct GetErrReq {
    AOP_Id envId; /* environ. handle */
};

union GetErrRet switch (AOP_Status retState) {
    case S_OK: void;
    case E_NOTOPEN: void;
    case E_SECURITY: void;
    case E_BSS: AOP_String ebsMsg;
    case W_BSS: AOP_String wbsMsg;
    case E_SSS: AOP_String essMsg;
    case W_SSS: AOP_String wssMsg;
    case E_MISC: AOP_String errMsg;
    case E_DAC: AOP_DAC_Status dacCode;
};

/* =====
* ASAM ODS Security methods
* The following methods are used for get/set security on data
* (Modifying ACL information) .
*/

typedef long AOP_Rights; /* Used for Rights-Values (bitmasked) */
enum AOP_SetType { /* how to modify bitmasked values */
    SET, /* Set the value to new bits (clears all bits before) */
    ADD, /* Adds the new bits to the existing ones (OR-function) */
    REMOVE /* Removes the bits from the existingvalue (~AND function) */

```

```

};

struct AOP_Acs { /* Access control structure */
    AOP_Id groupId; /* Usergroup Id */
    AOP_ElemId elemId; /* Application element+instance Id */
    AOP_Name attrName; /* Application attribute name */
    AOP_Rights rights; /* rights-value (bitmasked) 5-bits */
};

typedef AOP_Acs AOP_Acl<>;

struct AOP_Irs { /* Initial rights structure */
    AOP_Id groupId; /* Usergroup Id */
    AOP_ElemId elemId; /* Application element+instance Id */
    AOP_Rights rights; /* rights-value (bitmasked) */
    AOP_Id refAid; /* referencing application element. If set to 0, its been
                    used by all elements that refer to it */
};

typedef AOP_Irs AOP_IrsSeq<>;

struct GetRightsReq { /* getrights-request structure */
    AOP_Id envId; /* environ. handle */
    AOP_ElemId elemId<>; /* Application element+instance Ids */
    AOP_Name attrName<>; /* Application attribute names (list of elements
                        + list of attributes at the same time is not
                        allowed) */
};

union GetRightsRet switch (AOP_Status retState) {
    /* select return value structure */

    case E_NOTOPEN: void;
    case E_BSS: void;
    case E_SSS: void;
    case E_DAC: void;
    case E_SECURITY: void;
    case E_MISC: void;
    default: AOP_Acs acl<>; /* ACL will be returned */
};

struct SetRightsReq { /* setrights-request structure */
    AOP_Id envId; /* environ. handle */
    AOP_Acs acl<>; /* ACL */
    AOP_SetType setType; /* set/add/remove bits from rights in ACL */
};

```



```

};

struct SetRightsRet {
    AOP_Status retState;
};

struct GetIniRightsReq { /* getinirights-request structure */
    AOP_Id envId; /* environ. handle */
    AOP_ElemId elemId<>; /* List of application element+instance Ids */
};

union GetIniRightsRet switch (AOP_Status retState) {
    /* select return value structure */

    case E_NOTOPEN: void;
    case E_BSS: void;
    case E_SSS: void;
    case E_DAC: void;
    case E_SECURITY: void;
    case E_MISC: void;
    default: AOP_Irs irl<>; /* Initial rights list will be returned */
};

struct SetIniRightsReq { /* setinirights-request structure */
    AOP_Id envId; /* environ. handle */
    AOP_Irs irl<>; /* Initial rights list */
    AOP_SetType setType; /* set/add/remove bits from rights */
};

struct SetIniRightsRet {
    AOP_Status retState;
};

struct GetIniRightsRefReq { /* getinirightsref-request structure */
    AOP_Id envId; /* environ. handle */
    AOP_Id applId; /* Application element Id */
};

union GetIniRightsRefRet switch (AOP_Status retState) {
    /* select return value structure */

    case E_NOTOPEN: void;
    case E_BSS: void;
    case E_SSS: void;
    case E_DAC: void;
    case E_SECURITY: void;

```

```

        case E_MISC: void;
        default: AOP_Name attr<>; /* attribute-list will be returned */
};

struct SetIniRightsRefReq { /* setinirightsref-request structure */
    AOP_Id envId; /* environ. handle */
    AOP_Id applId; /* application element Id */
    AOP_Name refAttr; /* Reference Attribute */
    AOP_Flags set; /* use (1) or don't use (0) the reference-attribute for
                    ACL templates */
};

union SetIniRightsRefRet switch (AOP_Status retState) {
    /* select return value structure */

    case E_NOTOPEN: void;
    case E_BSS: void;
    case E_SSS: void;
    case E_DAC: void;
    case E_SECURITY: void;
    case E_MISC: void;
    default: AOP_Flags set; /* set-value will be returned */
};

struct SetCurrentIniRightsReq { /* setcurrentinirights-request structure */
    AOP_Id envId; /* environ. handle */
    AOP_Irs irl<>; /* Initial ACL to use in subsequent creation of new
                    instances */
    AOP_Flags set; /* Use (1) this ACL or don't use (0) this ACL for further
                    inserts anymore */
};

union SetCurrentIniRightsRet switch (AOP_Status retState) {
    /* select return value structure */

    case E_NOTOPEN: void;
    case E_BSS: void;
    case E_SSS: void;
    case E_DAC: void;
    case E_SECURITY: void;
    case E_MISC: void;
    default: AOP_Flags set; /* set-value will be returned */
};

```

```

struct SetPasswordReq { /* setpassword-request structure */
    AOP_Id envId; /* environ. handle */
    AOP_Id userid; /* user-id to set the password for */
    AOP_String oldpwd; /* current password */
    AOP_String newpwd; /* new password */
};

union SetPasswordRet switch (AOP_Status retState) {
    /* select return value structure */

    case E_NOTOPEN: void;
    case E_BSS: void;
    case E_SSS: void;
    case E_DAC: void;
    case E_SECURITY: void;
    case E_MISC: void;
    default: AOP_Flags ok; /* ok=1 if password changed, else ok=0 */
};

struct SetSecurityLevelReq { /* setsecuritylevel-request structure */
    AOP_Id envId; /* environ. handle */
    AOP_Id applId; /* application element id */
    AOP_Flags seclevel; /* security level (bitmasked) */
    AOP_SetType setType; /* set/add/remove security level bits */
};

union SetSecurityLevelRet switch (AOP_Status retState) {
    /* select return value structure */

    case E_NOTOPEN: void;
    case E_BSS: void;
    case E_SSS: void;
    case E_DAC: void;
    case E_SECURITY: void;
    case E_MISC: void;
    default: AOP_Flags seclevel; /* return security-level */
};

struct GetSecurityLevelReq { /* getsecuritylevel-request structure */
    AOP_Id envId; /* environ. handle */
    AOP_Id applId; /* application element id */
};

union GetSecurityLevelRet switch (AOP_Status retState) {

```

```

/* select return value structure */

case E_NOTOPEN: void;
case E_BSS: void;
case E_SSS: void;
case E_DAC: void;
case E_SECURITY: void;
case E_MISC: void;
default: AOP_Flags seclevel; /* return security-level */
};

/*=====
* procedures
*/
program AODSPROG {
    version AODSVERS3 {
        /* Open Environement ----- */
        OpenEnvRet AOP_OpenEnv (OpenEnvReq) = 1;

        /* Close Environement ----- */
        CloseEnvRet AOP_CloseEnv (CloseEnvReq) = 2;

        /* Metainfo - queries ----- */
        GetApplInfRet AOP_GetApplInf (GetApplInfReq) = 14;
        GetAttrRet AOP_GetAttr (GetAttrReq) = 12;
        /* Metainfo - inserts ----- */

        /* Data query ----- */
        GetValRet AOP_GetVal (GetValReq) = 21;
        GetInstAttrRet AOP_GetInstAttr (GetInstAttrReq) = 22;
        GetInstRefRet AOP_GetInstRef (GetInstRefReq) = 25;
        GetValERet AOP_GetValE (GetValEReq) = 26;

        /* Data update on application elements ----- */
        PutValRet AOP_PutVal (PutValReq) = 27;
        SetInstRefRet AOP_SetInstRef (SetInstRefReq) = 28;

        /* Operations on measured values level ----- */

```

```

GetValAttrRet AOP_GetValAttr (GetValAttrReq) = 31;

/* candidate for removal, use GetValVal with zero size */
GetValValRet AOP_GetValVal (GetValValReq) = 32;
GetValInfRet AOP_GetValInf (GetValInfReq) = 35;
PutValValRet AOP_PutValVal (PutValValReq) = 38;

/* Utility functions ----- */
SetParRet AOP_SetPar (SetParReq) = 51;
GetParRet AOP_GetPar (GetParReq) = 52;
GetErrRet AOP_GetErr (GetErrReq) = 59;

/* Security functions ----- */
SetRightsRet AOP_SetRights (SetRightsReq) = 61;
GetRightsRet AOP_GetRights (GetRightsReq) = 62;
SetIniRightsRet AOP_SetIniRights (SetIniRightsReq) = 63;
GetIniRightsRet AOP_GetIniRights (GetIniRightsReq) = 64;
SetIniRightsRefRet AOP_SetIniRightsRef (SetIniRightsRefReq) = 65;
GetIniRightsRefRet AOP_GetIniRightsRef (GetIniRightsRefReq) = 66;
SetCurrentIniRightsRet AOP_SetCurrentIniRights (
                                SetCurrentIniRightsReq) = 67;
SetPasswordRet AOP_SetPassword (SetPasswordReq) = 68;
SetSecurityLevelRet AOP_SetSecurityLevel (SetSecurityLevelReq) = 69;
GetSecurityLevelRet AOP_GetSecurityLevel (GetSecurityLevelReq) = 70;
} = 3;

/* Program number for all procedures ----- */
} = rpc_number;

```

13 ASAM ODS OO-API

13.1 概述

13.1.1 一般信息

ASAM ODS OO-API（面向对象应用编程接口）主要是为了用各种编程语言和解释器能方便地对任何 ASAM ODS 数据进行细颗粒度的读写访问而规定的。面向对象的方法是本部分的基础，但非面向对象编程语言也可用于应用编程。

目前 OO-API 的实现是基于 CORBA 的对象传输机制和访问媒介，但这不是 ASAM ODS 的前提要求，也可以实现任何其他分布式对象访问方式。这隐含地意味着遵循 OO-API 规范并不能确保产品的兼

容性。因此每个服务器或客户端实现必须说明它是基于什么技术（例如 CORBA v2.0）以及顺利运行所需的第三方软件是什么（例如必须说明它运行在何种平台上等等），若对兼容性有质疑，可通过参加官方组织的测试解决，详见 ASAM 网站介绍。

与 ASAM ODS 的 RPC-API（见第 13 章）相反，即使是处理少量的数据，OO-API 细粒度的访问也可能潜在地产生额外的网络流量。ASAM ODS OO-API 的主要设计目的是为了便于编程和易于使用，性能和网络流量的优化还不是一个重要问题。然而 ASAM ODS OO-API 的实现可在内部使用 ASAM ODS RPC-API 以改善网络流量。

ASAM ODS OO-API 的功能是完全根据 CORBA IDL（公共对象请求代理结构-接口定义语言）定义的，选择 CORBA IDL 主要是基于以下原因：

——因为它是一种定义完善的描述语言；

——因为已有各类应用程序可为不同语言（例如 C++ 和 JAVA）生成 CORBA 存根和框架代码。OO-API 语言联编可用于 C、C++、TCL、Perl、Python、交互式数据语言 IDL、VB、JAVA、Smalltalk、Delphi 等。

该方法的主要优势是易于实现且简化了功能调用，每次调用不需使用复杂的结构，只需要少量的参数。这些参数本身也具有简单结构（例如字符串、长整数、浮点数、字符串列表、值列表等）。

OO-API 对象层次结构的起始点总是 ASAM ODS 的 Factory 对象(AoFactory)，一旦该对象存在，就可用于创建一个 ASAM ODS 会话(AoSession)的随机数。此外该对象还提供了一般信息，例如 Factory 的名称和类型、描述以及当前接口版本。

以下所列的 AoFactory 和 AoSession 以及所有必需的数据类型定义和接口定义也都是 ODS.IDL 文件的组成部分：

- ApplicationStructure, ApplicationElement, ApplicationAttribute, ApplicationRelation;
- ApplElemAccess;
- BaseStructure, BaseElement, BaseAttribute, BaseRelation;
- Blob;
- Column;
- ElemResultSetExtSeqIterator;
- EnumerationDefinition;
- InstanceElement, InstanceElementIterator;
- Measurement;
- NameIterator, NameValueIterator, NameValueUnitIdIterator, NameValueUnitIterator,;
- NameValueUnitSequenceIterator;
- Query, QueryEvaluator;
- SMatLink;
- SubMatrix;
- ValueMatrix。

使用“Application”接口组来创建、处理和查询应用模型的信息，通常不可能或不期望改变应用模型，应用模型一般在项目的设计阶段被创建，因而很少改动，因为改变应用模型将导致大量的映射工作

或者重新构建底层数据库。应用模型描述了真实数据是如何组织和构造的——因此应用模型中的信息一般被当作元数据。

“AppElemAccess”接口是为了在少量的客户/服务器访问循环中访问大量数据，使用该接口可改善插入/删除/更新操作的性能。“Base”接口组用于获取 ASAM ODS 基础模型的信息，基础模型是只读的，应用程序不能更改该模型中的信息。

“Instance”组用于创建、修改和删除数据对象，这些对象的结构在应用模型中定义。

Measurement、SubMatrix、SMatLink、ValueMatrix、Column 和 Blob 对象用于处理数组数据。

Query 接口定义了基于应用模型的选择。该对象对于创建数据结构子集从而减少需求的数据数量而言是必需的。查询接口设计的目的是针对 COS（OMG 的公共对象服务）查询服务，然而这里虽然引入了 Query 接口，但这一版本的 ODS 规范还不支持它，因为 ASAM 查询语言的定义尚未完成。客户还不能使用 Query、QueryEvaluator 和 AoSession.createQueryEvaluator 等接口，服务端对这些接口中用到的所有方法都必需抛出一个 AO_NOT_IMPLEMENTED 异常。

客户端只能修改一个活动事务处理中的数据，不允许插入、删除或更新没有事务处理的数据。

13.2 OO-API 的接口

以下各节内容所列的 OO-API 的对象、元素和函数都是按照字母顺序排序的，此外分别列出了调用顺序、参数（带类型）、参数传递、返回类型和返回值。最后给出了 JAVA 示例。

注：传递参数通常是区分大小写的，其中可使用通配符，“？”表示一个匹配的字符，“*”表示匹配字符的序列。

13.2.1 AOFACTORY

(1) AOFACTORY_GETDESCRIPTION

目的：获取 ASAM ODS 的 factory 描述。如果没有可用的描述信息，则返回空字符串且不引发异常。服务器将从 AoEnvironment 实例的基础属性 “description” 中调出描述信息。

参数：无。

Java 调用序列：T_STRING description = aoFactory.getDescription（）；

返回：

返回名称：description

返回类型：T_STRING

ASAM ODS factory 的说明

示例：Java 语言

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

可能的异常详见 13.3.18 节如下函数：

——AO_IMPLEMENTATION_PROBLEM；

——AO_NOT_IMPLEMENTED；

——AO_NO_MEMORY。

(2) AOFACORY_GETINTERFACEVERSION

目的：获取 ASAM ODS 的 factory 的接口版本信息。每个 ODS 版本的接口版本是个固定字符串。该字符串是服务器端实现的接口版本。字符串的格式如下：**V<Major Version Nr>.<Minor Version Nr>.<Revision Nr>**

其中：

- Major Version Nr (numeric: [0-9]+)
- Minor Version Nr (numeric: [0-9]+)
- Revision Nr (numeric: [0-9]+)

版本示例：'V5.2.0'.

参数：无。

Java 调用序列：T_STRING interfaceVersion = aoFactory.getInterfaceVersion () ;

返回：

返回名称：interfaceVersion

返回类型：T_STRING

ASAM ODS factory 的版本。

示例：Java 语言

```
String interfaceVersion;

interfaceVersion = aoFactory.getInterfaceVersion () ;
```

可能的异常详见 13.3.18 节如下函数：

- AO_IMPLEMENTATION_PROBLEM;
- AO_NOT_IMPLEMENTED;
- AO_NO_MEMORY。

(3) AOFACORY_GETNAME

目的：获取 ASAM ODS factory 的名称。如果名称信息不可用，则返回空字符串且不引发异常。

参数：无。

Java 调用序列：T_STRING factoryName = aoFactory.getName () ;

返回：

返回名称：factoryName

返回类型：T_STRING

ASAM ODS factory 的名称。

示例：Java 语言

```
String name;

name = aoFactory.getName () ;
```

可能的异常详见 13.3.18 节如下函数：

- AO_IMPLEMENTATION_PROBLEM;
- AO_NOT_IMPLEMENTED;

——AO_NO_MEMORY。

(4) AOFACORY_GETTYPE

目的：获取 ASAM ODS 的 factory 类型。如果类型信息不可用，则返回空字符串且不引发异常。

服务器通过 AoEnvironment 实例的基础属性 “application_model_type” 调出类型信息。

参数：无。

Java 调用序列：T_STRING factoryType = aoFactory.getType ()。

返回：

返回名称：factoryType

返回类型：T_STRING

ASAM ODS factory 的类型。

示例：Java 语言

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

可能的异常详见 13.3.18 节如下函数：

——AO_IMPLEMENTATION_PROBLEM;

——AO_NOT_IMPLEMENTED;

——AO_NO_MEMORY。

(5) AOFACORY_NEWSESSION

目的：在 ASAM ODS 服务器上建立一个新会话。服务器通常会检查会话活动并且在经过一段不活动的时间周期后关闭该会话。

参数：auth (Type = T_STRING)

可能包含授权信息的字符串。目前支持下列值：

USER 用户的名称

PASSWORD 用户的密码

CREATE_COSESSION_ALLOWED 说明是否可在该会话中创建一个共同会话 (TRUE, FALSE)。默认值是 FALSE。在会话中是只读的。该值可以按照任意顺序说明，且必须用逗号分隔。

示例：

```
"USER=hans, PASSWORD=secret, CREATE_COSESSION_ALLOWED=TRUE"
```

Java 调用序列：AoSession aoSession = aoFactory.newSession (auth) ;

返回：

返回名称：aoSession

返回类型：AoSession

创建新的 ASAM ODS session..

示例：示例：Java 语言

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

AoSession session;

session = aoFactory.newSession ("USER=hans, PASSWORD=secret");
```

可能的异常详见 13.3.18 节如下函数：

- 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。

13.2.2 AOSESSION

每次客户端要访问 ASAM ODS 服务器时就需要打开一个会话，当客户端完成会话后，应当关闭该会话。

尽管这是客户端推荐的操作，但不能为服务器确保这样的操作，因为：

- 可能在某些特定条件下客户端实现会将会话活动保持一段很长的时间；
- 可能由于客户端和服务器之间的网络连接中断，导致客户端不能再关闭会话。

但是必需避免服务器上无意义的活动会话数量随着时间增加得越来越多，因此 ASAM ODS 服务器将关闭在一段时间内不与服务器进行交互的所有会话。通常当服务器启动时将设定该时间值（推荐的默认值为 30 分钟），该时间值如何设置取决于服务器的实现。客户端实现应注意服务器管理员可能会设置不同的时间值。

如果客户端视图在会话关闭后访问服务器，则会产生一个 CORBA 异常。客户端实现需要注意这是时间过期后产生的异常，应当捕获并处理该异常。

(1) AOSESSION_ABORTTRANSACTION

目的：中止（回退）一个事务。事务中所做的更改将丢失。

参数：None.

Java 调用序列：aoSession.abortTransaction ()。

返回：None.

示例：Java 语言

```
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 ( ) ;
}
```

可能的异常详见 13.3.18 节如下函数：

——AO_ACCESS_DENIED;
 ——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE;
 ——AO_TRANSACTION_NOT_ACTIVE。

(2) AOSESSION_CLOSE

目的：关闭 ASAM ODS 服务器的会话。提交活动事务。

参数：None.

Java 调用序列：aoSession.close ()。

返回：None.

示例：Java 语言

```
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 ( ) ;
}
```

可能的异常详见 13.3.18 节如下函数：

——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;

——AO_SESSION_NOT_ACTIVE。

(3) AOSESSION_COMMITTRANSACTION

目的：提交一个事务。在事务中所做的更改将成为永久性的。

参数：None.

Java 调用序列：aoSession.commitTransaction () ；

返回：None.

示例：Java 语言

```
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 ( ) ;
}
```

可能的异常详见 13.3.18 节如下函数：

——AO_ACCESS_DENIED;
——AO_CONNECTION_LOST;
——AO_IMPLEMENTATION_PROBLEM;
——AO_NOT_IMPLEMENTED;
——AO_NO_MEMORY;
——AO_SESSION_NOT_ACTIVE;
——AO_TRANSACTION_NOT_ACTIVE。

(4) AOSESSION_CREATEBLOB

目的：在服务器上创建一个带有接口 Blob 的新对象。此对象可用于创建数据类型 DT BLOB 的属性值。

参数：None.

Java 调用序列：Blob blob = aoSession.createBlob () ；

返回：返回名称：blob

返回类型: Blob

在服务器上生成的 blob 对象的引用。

可能的异常 (详见 13.3.18 节):

- AO_BAD_PARAMETER;
- AO_CONNECTION_LOST;
- AO_IMPLEMENTATION_PROBLEM;
- AO_NOT_IMPLEMENTED;
- AO_NO_MEMORY;
- AO_SESSION_NOT_ACTIVE。

(5) AOSESSION_CREATECOSESSION

目的: 创建一个共同会话。共同会话与登录时的原始任务相同。除了使用相同的身份验证之外, 共同会话与原始会话没有关系。带有组件的客户机应用程序可以将所有组件自己的会话提供给 the ASAM ODS 服务器。因此, 组件可以在准备好会话时关闭任务, 并且组件可以启动自己的事务, 而不与其他组件发生任何冲突。在接口 AoFactory 的 newSession 方法的“auth”参数中变量 CREATE_COSESSION_ALLOWED 设置为 TRUE 时, 才允许创建共同任务。当调用 createSession 时将创建一个新会话, 否则拒绝访问 AO_ACCESS_DENIED。

参数: None.

Java 调用序列: AoSession cosession = aoSession.createCoSession ();

返回: 返回名称: cosession

返回类型: AoSession

共同会话。

可能的异常详见 13.3.18 节如下函数:

- AO_CONNECTION_LOST;
- AO_IMPLEMENTATION_PROBLEM;
- AO_NOT_IMPLEMENTED;
- AO_NO_MEMORY;
- AO_ACCESS_DENIED。

(6) AOSESSION_CREATEQUERYEVALUATOR

目的: 创建 QueryEvaluator 对象

参数: None.

Java 调用序列: QueryEvaluator queryEvaluator = aoSession.createQueryEvaluator ();

返回: 返回名称: queryEvaluator

返回类型: QueryEvaluator

创建新的查询计算对象

可能的异常详见 13.3.18 节如下函数:

——AO_ACCESS_DENIED;
——AO_BAD_PARAMETER;
——AO_CONNECTION_LOST;
——AO_IMPLEMENTATION_PROBLEM;
——AO_NOT_IMPLEMENTED;
——AO_NO_MEMORY;
——AO_SESSION_NOT_ACTIVE。

(7) AOSESSION_FLUSH

目的：变更参数。。

参数：None.

Java 调用序列：aoSession.flush () ；

返回：None.

示例：Java 语言

```
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 ( ) ;  
}
```

可能的异常（详见 13.3.18 节）：

——AO_ACCESS_DENIED;
——AO_CONNECTION_LOST;
——AO_IMPLEMENTATION_PROBLEM;
——AO_NOT_IMPLEMENTED;
——AO_NO_MEMORY;

——AO_SESSION_NOT_ACTIVE;
 ——AO_TRANSACTION_NOT_ACTIVE。

(8) AOSESSION_GETAPPLELEMACCESS

目的：从当前会话获取应用元素访问对象。

参数：None.

Java 调用序列：ApplElemAccess applElemAccess = aoSession.getApplElemAccess () ;

返回：返回名称：applElemAccess

返回类型：ApplElemAccess

应用元素访问对象。

示例：Java 语言

```
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 ( ) ;
}
```

可能的异常详见 13.3.18 节如下函数：

——AO_ACCESS_DENIED;
 ——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

(9) AOSESSION_GETAPPLICATIONSTRUCTURE

目的：通过返回带有接口 **ApplicationStructure** 的对象，从当前会话获取应用程序模型。关于应用模型的完整信息可以通过该接口获得。

参数：None.

Java 调用序列：ApplicationStructure applicationStructure = aoSession.getApplicationStructure

() ;

返回：返回名称：applicationStructure

返回类型：ApplicationStructure

应用模型。

示例：Java 语言

```
import org.asam.ods.AoSession;
import org.asam.ods.ApplicationStructure;
// Create a new session with aoFactory.
AoSession session;
session = aoFactory.newSession ("");
// Session successfully created ?
if (session != null) {
// Get Application Model.
ApplicationStructure as =
session.getApplicationStructure ();
...
// Close the session.
session.close ();
}
```

可能的异常详见 13.3.18 节如下函数：

- AO_ACCESS_DENIED;
- AO_CONNECTION_LOST;
- AO_IMPLEMENTATION_PROBLEM;
- AO_NOT_IMPLEMENTED;
- AO_NO_MEMORY;
- AO_SESSION_NOT_ACTIVE。

(10) AOSESSION_GETAPPLICATIONSTRUCTUREVALUE

目的：从当前的会话获得应用模型的值。

参数：None。

Java 调用序列：ApplicationStructureValue applicationStructureValue =
aoSession.getApplicationStructureValue ();

返回：返回名称：applicationStructureValue

返回类型：ApplicationStructureValue

应用模型的值。

可能的异常详见 13.3.18 节如下函数：

- AO_ACCESS_DENIED;

——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

(11) AOSESSION_GETBASESTRUCTURE

目的：通过返回具有接口 `BaseStructure` 的对象，从当前会话获取 ASAM ODS 基模型。通过该接口可以获得关于基模型的完整信息；它包含 ASAM ODS 指定的所有可能的基本元素和所有可能的基本属性。

参数：None.

Java 调用序列： `BaseStructure baseStructure = aoSession.getBaseStructure ( ) ;`

返回：返回名称： `baseStructure`

返回类型： `BaseStructure`

基础模型.

可能的异常详见 13.3.18 节如下函数：

——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

(12) AOSESSION_GETCONTEXT

目的：从会话中获取上下文变量。可以指定模式字符串来选择组。

参数： `varPattern (Type = Pattern)`

The name or the search pattern for the context variable (s) .

Java 调用序列： `NameValueIterator nvIterator = aoSession.getContext (varPattern) ;`

返回：返回名称： `nvIterator`

返回类型： `NameValueIterator`

上下文变量列表.

示例：Java 语言

```
import org.asam.ods.AoSession;
import org.asam.ods.NameValue;
import org.asam.ods.NameValueIterator;
// Create a new session with aoFactory.
AoSession session;
session = aoFactory.newSession ("" ) ;
```

```

// Session successfully created ?
if (session != null) {
// Create holder for return values.
NameValueIterator nvIte;
NameValue[] nvSeq;
// Ask for all variables with any name pattern.
nvIte = session.getContext ("*");
// Get max. 40 variables
nvSeq = nvIte.nextN (40);
...
// Print the result variable list to standard output.
for (int i=0; i<nvSeq.length; i++) {
System.out.println (nvSeq[i].name + " = \"\" +
nvSeq[i].value.u.stringVal () + "\"\"");
}
...
// Close the session.
session.close ();
}

```

可能的异常详见 13.3.18 节如下函数：

- AO_BAD_PARAMETER;
- AO_CONNECTION_LOST;
- AO_IMPLEMENTATION_PROBLEM;
- AO_NOT_FOUND;
- AO_NOT_IMPLEMENTED;
- AO_NO_MEMORY;
- AO_SESSION_NOT_ACTIVE。

(13) AOSESSION_GETCONTEXTBYNAME

目的：用名字从会话中获得上下文变量。

参数：varName (Type = Name)

请求的上下文变量名称。

Java 调用序列：NameValue contextVariable = aoSession.getContextByName (varName);

返回：返回名称：contextVariable

返回类型：NameValue

请求的上下文变量。

示例：Java 语言

```

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 ( ) ;
}

```

可能的异常详见 13.3.18 节如下函数：

- AO_BAD_PARAMETER;
- AO_CONNECTION_LOST;
- AO_IMPLEMENTATION_PROBLEM;
- AO_NOT_FOUND;
- AO_NOT_IMPLEMENTED;
- AO_NO_MEMORY;
- AO_SESSION_NOT_ACTIVE。

(14) AOSESSION_GETDESCRIPTION

目的：获取 ASAM ODS 会话的描述。会话的描述与 ASAM ODS 工厂的描述相同。如果描述不可用，则返回一个空字符串，并且不抛出异常。服务器从 **AoEnvironment** 实例的基本属性“description”中加载描述。

参数：None.

Java 调用序列：T_STRING description = aoSession.getDescription () ;

返回：返回名称：description

返回类型：T_STRING

ASAM ODS 会话的描述.

示例：Java 语言

```

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

```

可能的异常详见 13.3.18 节如下函数：

- AO_IMPLEMENTATION_PROBLEM;
- AO_NOT_IMPLEMENTED;

——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE;
 ——AO_CONNECTION_LOST。

(15) AOSESSION_GETENUMERATIONATTRIBUTES

目的：获取包含应用程序模型中属于枚举类型的所有元素的所有属性的列表。此列表的每个条目都提供了关于具有枚举数据类型的应用程序属性的名称的信息，此应用程序属性的 id 应属于枚举的名称，该值可以作为应用程序属性的值。该方法还返回基于应用程序的属性（其中枚举已经由 ASAM ODS 指定）。在应用程序模型不显示任何枚举类型属性的（不寻常的）情况下，将返回一个空列表。

参数：None.

Java 调用序列：EnumerationAttributeStructureSequence allEnumAtts[] =

aoSession.getEnumerationAttributes ();

返回：返回名称：allEnumAtts[]

返回类型：EnumerationAttributeStructureSequence

具有属性名称、元素 ID 和枚举名称的列表.

可能的异常详见 13.3.18 节如下函数：

——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

(16) AOSESSION_GETENUMERATIONSTRUCTURE

目的：获取应用程序模型中使用的所有枚举。此应用程序模型中可用的所有枚举的列表将返回给客户机。此列表包含每个枚举的名称及其所有项（包括项索引和项名称）。

参数：None.

Java 调用序列：EnumerationStructureSequence allEnums[] =

aoSession.getEnumerationStructure ();

返回：返回名称：allEnums[]

返回类型：EnumerationStructureSequence

枚举及其所有项.

可能的异常详见 13.3.18 节如下函数：

——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;

——AO_SESSION_NOT_ACTIVE。

(17) AOSESSION_GETLOCKMODE

目的：获取当前锁定模式。锁定模式告诉服务器要锁定哪些对象以应对即将发生的更改。可以调用应用元素、元素的实例或子元素。

参数：None.

Java 调用序列：LockMode lockMode = aoSession.getLockMode () ;

返回：返回名称：lockMode

返回类型：T_SHORT

当前锁定模式。锁定模式常量在接口锁定模式中定义。

接口定义语言 IDL 不允许设置枚举值，因此常量定义必须在接口中完成。

可能的异常详见 13.3.18 节如下函数：

——AO_ACCESS_DENIED;
 ——AO_BAD_PARAMETER;
 ——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

(18) AOSESSION_GETNAME

目的：获取 ASAM ODS 会话的名称。会话的名称与 ASAM ODS 工厂的名称相同。如果名称不可用，则返回一个空字符串，并且不抛出异常。服务器从基本属性“name”加载名称。

参数：None.

Java 调用序列：Name sessionName = aoSession.getName () ;

返回：返回名称：sessionName

返回类型：Name

ASAM ODS 会话的名称.

示例：Java 语言

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

可能的异常详见 13.3.18 节如下函数：

——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE;
 ——AO_CONNECTION_LOST。

(19) AOSESSION_GETTYPE

目的：获取 ASAM ODS 会话的类型。会话的类型与 ASAM ODS 工厂的类型相同。如果类型不可用，则返回一个空字符串，并且不抛出异常。服务器从基本属性“applicati”加载类型。

参数：None.

Java 调用序列：T_STRING sessionType = aoSession.getType () ;

返回：返回名称：sessionType

返回类型：T_STRING

ASAM ODS 会话的类型.

示例：Java 语言

String type;

type = aoSession.getType () ;

可能的异常详见 13.3.18 节如下函数：

——AO_IMPLEMENTATION_PROBLEM;

——AO_NOT_IMPLEMENTED;

——AO_NO_MEMORY;

——AO_SESSION_NOT_ACTIVE;

——AO_CONNECTION_LOST。

(20) AOSESSION_GETUSER

目的：获取 ASAM ODS 会话的用户。此方法返回已登录到此会话的用户的实例。这是 application 元素的实例，该元素派生自基本元素 AoUser，其名称由上下文变量名给出。

参数：None.

Java 调用序列：InstanceElement myUser = aoSession.getUser () ;

返回：返回名称：myUser

返回类型：InstanceElement

登录到 ASAM ODS 会话的用户实例.

可能的异常详见 13.3.18 节如下函数：

——AO_IMPLEMENTATION_PROBLEM;

——AO_NOT_IMPLEMENTED;

——AO_NO_MEMORY;

——AO_CONNECTION_LOST。

(21) AOSESSION_LISTCONTEXT

目的：列出会话中的上下文变量的名称。可以指定模式字符串来选择变量组。

参数：varPattern (Type = Pattern)

上下文变量名称或搜索模式.

Java 调用序列: `Namelterator nameliterator = aoSession.listContext (varPattern) ;`

返回: 返回名称: `nameliterator`

返回类型: `Namelterator`

上下文变量名列表.

可能的异常详见 13.3.18 节如下函数:

——`AO_BAD_PARAMETER;`
 ——`AO_CONNECTION_LOST;`
 ——`AO_IMPLEMENTATION_PROBLEM;`
 ——`AO_NOT_IMPLEMENTED;`
 ——`AO_NOT_FOUND;`
 ——`AO_NO_MEMORY;`
 ——`AO_SESSION_NOT_ACTIVE.`

(22) AOSESSION_REMOVECONTEXT

目的: 从会话中删除上下文变量。可以指定模式字符串来删除变量组。

参数: `varPattern (Type = Pattern)`

要删除的上下文变量名称或搜索模式.

Java 调用序列: `aoSession.removeContext (varPattern) ;`

返回: `None.`

示例: Java 语言

```
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 () ;
}
```

可能的异常详见 13.3.18 节如下函数：

——AO_BAD_PARAMETER;
 ——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NOT_FOUND;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

(23) AOSESSION_SETCONTEXT

目的：设置/修改已知的上下文变量或向会话添加新的上下文变量。

参数：contextVariable (Type = NameValue)

上下文变量。

Java 调用序列：aoSession.setContext (contextVariable) ;

返回：None.

示例：Java 语言

```
import org.asam.ods.AoSession;
import org.asam.ods.types.TS_Union;
import org.asam.ods.types.TS_Value;
import org.asam.ods.types.NameValue;
// Create a new session with aoFactory.
AoSession session;
session = aoFactory.newSession ("");
// Session successfully created ?
if (session != null) {
    // Define variable name and value.
    String varName = "TestVar";
    float varValue = 4711.3;
    /* Build the name-value pair.
    short[] flag = {0};
    TS_Union union = new TS_Union ( ) ;
    union.floatVal (varValue) ;
    TS_Value tsVal = new TS_Value (union,flag) ;
    NameValue nvPair = new NameValue (varName,tsVal)
    // Create a float test variable named TestVar.
    session.SetContext (nvPair) ;
    ...
```



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

可能的异常详见 13.3.18 节如下函数：

```
——AO_BAD_PARAMETER;
——AO_CONNECTION_LOST;
——AO_IMPLEMENTATION_PROBLEM;
——AO_NOT_IMPLEMENTED;
——AO_NO_MEMORY;
——AO_SESSION_NOT_ACTIVE。
```

(24) AOSESSION_SETCONTEXTSTRING

目的：设置/修改已知的上下文变量或向会话添加新的上下文变量。对于常用的字符串变量类型，这是一种方便的方法。它使用 `setContextinternally`。

参数：varName (Type = Name)

上下文变量的名称。

value (Type = T_STRING)

上下文变量的值。

Java 调用序列：aoSession.setContextString (varName,value) ;

返回：None.

示例：Java 语言

```
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 ( ) ;
}
```

可能的异常详见 13.3.18 节如下函数：

```
——AO_BAD_PARAMETER;
```

——AO_CONNECTION_LOST;
——AO_IMPLEMENTATION_PROBLEM;
——AO_NOT_IMPLEMENTED;
——AO_NO_MEMORY;
——AO_SESSION_NOT_ACTIVE。

(25) AOSESSION_SETCURRENTINITIALRIGHTS

目的：每个新创建的实例都将初始权限设置为<acl>。此方法覆盖应用初始权限的默认方法。初始权限仅对 `currentsession` 有效。

参数：`irlEntries` (Type = `InitialRightSequence`)

The current initial rights.

`set` (Type = `T_BOOLEAN`)

设置 (0) 移除 (1) 当前初始化权限。以前的初始化权限将丢失。如果参数集为 0 (移除) 参数 `irlEntries` 将被忽略。

Java 调用序列：`aoSession.setCurrentInitialRights (irlEntries,set)` ;

返回：None.

可能的异常详见 13.3.18 节如下函数：

——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。

(26) AOSESSION_SETLOCKMODE

目的：设置新的锁定模式。锁定模式告诉服务器要锁定哪些对象以应对即将发生的更改。可以调用应用元素、元素的实例或子元素。

参数：`lockMode` (Type = `T_SHORT`)

新的锁定模式。锁定模式常量在接口锁定模式中定义。

接口定义语言 IDL 不允许设置枚举值，因此常量定义必须在接口中完成。

Java 调用序列：`aoSession.setLockMode (lockMode)` ;

返回：None.

可能的异常详见 13.3.18 节如下函数：

——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。

(27) AOSESSION_SETPASSWORD

目的：更改使用<username>定义的密码。用户需要提供<oldPassword>，超级用户可以不用提供<oldPassword>更改用户密码。如果没有用户名，则将更改当前会话用户的密码。密码通常在用户实例的属性中加密。创建一个新用户可以通过创建一个新实例来完成，密码必须由超级用户设置。

参数：username (Type = T_STRING)

将更改密码的用户的名称。如果没有用户名，当前用户的密码将被更改如果用户名与当前用户不同，则当前用户必须是超级用户。

oldPassword (Type = T_STRING)

用户的密码。正常的用户必须提供用户密码。超级用户可以不用提供密码更改密码。

newPassword (Type = T_STRING)

用户的新密码。

Java 调用序列：aoSession.setPassword (username,oldPassword,newPassword)；

返回：None.

可能的异常详见 13.3.18 节如下函数：

——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。

(28) AOSESSION_STARTTRANSACTION

目的：在物理存储系统（例如数据库系统）上启动事务。只有当事物启动时，才允许创建或修改实例或测量数据。这些更改在事务提交时变为永久性的，或者在事务中止时丢。如果会话关闭，事务将自动提交。如果事务已处于活动状态，则不抛出异常。

参数：None.

Java 调用序列: `aoSession.startTransaction ()` ;

返回: None.

示例: Java 语言

```
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 () ;
}
```

可能的异常详见 13.3.18 节如下函数:

- AO_ACCESS_DENIED;
- AO_CONNECTION_LOST;
- AO_IMPLEMENTATION_PROBLEM;
- AO_NOT_IMPLEMENTED;
- AO_NO_MEMORY;
- AO_SESSION_NOT_ACTIVE;
- AO_TRANSACTION_ALREADY_ACTIVE。

13. 2. 3 APPLELEMACCESS

(1) APPLELEMACCESS_DELETEINSTANCES

目的: 从语义元素中删除实例。以便继承超类中给定 ID 的应用元素。这个方法可用于删除同一应用元素的多个实例。删除实例子类的应用元素实例。

参数: `aid (Type = T_LONGLONG)`

应用元素的 ID.

`instIds (Type = S_LONGLONG)`

实例 Id 的序列。在 RPC-API 中这个信息存储在结构化 `PutValReq` 和请求 `AOP_PutValReq` 的 `ElemId` 和 `nvSeq`。

Java 调用序列: `applElemAccess.deleteInstances (aid,instIds)`;

返回: None.

可能的异常详见 13.3.18 节如下函数:

——AO_BAD_PARAMETER;
 ——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

(2) APPLELEMACCESS_GETATTRIBUTERIGHTS

目的: 检索给定应用属性<aid>/<attr_name>的访问控制列表信息。

参数: aid (Type = T_LONGLONG)

应用元素的 ID.

attrName (Type = T_STRING)

属性名.

Java 调用序列: ACL[] aclEntries = applElemAccess.getAttributeRights (aid,attrName) ;

返回: 返回名称: aclEntries

返回类型: ACLSequence

给定应用属性的访问控制列表。

可能的异常详见 13.3.18 节如下函数:

——AO_BAD_PARAMETER;
 ——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

(3) APPLELEMACCESS_GETELEMENTINITIALRIGHTS

目的: 请求应用元素<aid>的检索访问控制列表信息。

参数: aid (Type = T_LONGLONG)

应用元素 Id 。

Java 调用序列: InitialRight[] initialRights = applElemAccess.getElementInitialRights (aid) ;

返回: 返回名称: initialRights

返回类型: InitialRightSequence

给定应用元素初始权限的访问控制列表。

可能的异常详见 13.3.18 节:

- AO_BAD_PARAMETER;
- AO_CONNECTION_LOST;
- AO_IMPLEMENTATION_PROBLEM;
- AO_NOT_IMPLEMENTED;
- AO_NO_MEMORY;
- AO_SESSION_NOT_ACTIVE。

(4) APPLELEMACCESS_GETELEMENTRIGHTS

目的：检索请求应用元素<aid>的访问控制列表信息。

参数：aid (Type = T_LONGLONG)

应用元素 Id。

Java 调用序列：ACL[] aclEntries = applElemAccess.getElementRights (aid)。

返回：返回名称：aclEntries

返回类型：ACLSequence

给定应用元素的范围控制列表实体。

可能的异常详见 13.3.18 节如下函数：

- AO_BAD_PARAMETER;
- AO_CONNECTION_LOST;
- AO_IMPLEMENTATION_PROBLEM;
- AO_NOT_IMPLEMENTED;
- AO_NO_MEMORY;
- AO_SESSION_NOT_ACTIVE。

(5) APPLELEMACCESS_GETINITIALRIGHTREFERENCE

目的：得到用于检索初始权的所有的属性名（引用）。

参数：aid (Type = T_LONGLONG)

应用元素 Id。

Java 调用序列：Name[] refNameList = applElemAccess.getInitialRightReference (aid)。

返回：返回名称：refNameList

返回类型：NameSequence

用于引用决定初始权的名字。

可能的异常详见 13.3.18 节如下函数：

- AO_BAD_PARAMETER;
- AO_CONNECTION_LOST;
- AO_IMPLEMENTATION_PROBLEM;
- AO_NOT_IMPLEMENTED;
- AO_NO_MEMORY;

——AO_SESSION_NOT_ACTIVE。

(6) APPLELEMACCESS_GETINSTANCEINITIALRIGHTS

目的：请求实例<aid>/<iid>的检索初始权范围控制列表。

参数：aid (Type = T_LONGLONG)

应用元素 Id。

iid (Type = T_LONGLONG)

实例 Id。

Java 调用序列：InitialRight[] initialRights = applElemAccess.getInstanceInitialRights (aid,iid)。

返回：返回名称：initialRights

返回类型：InitialRightSequence

初始权给定实例的访问控制列表实体。

可能的异常详见 13.3.18 节如下函数：

——AO_BAD_PARAMETER;
 ——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

(7) APPLELEMACCESS_GETINSTANCERIGHTS

目的：请求实例<aid>/<iid>的检索访问控制列表信息。

参数：aid (Type = T_LONGLONG)

应用元素 Id。

iid (Type = T_LONGLONG)

实例 Id。

Java 调用序列：ACL[] aclEntries = applElemAccess.getInstanceRights (aid,iid)。

返回：返回名称：aclEntries

返回类型：ACLSequence

给定实例的范围控制列表实体。

可能的异常详见 13.3.18 节如下函数：

——AO_BAD_PARAMETER;
 ——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

(8) APPLELEMACCESS_GETINSTANCES

目的：执行查询。所请求的不同应用元素的数量由 QueryStructure 的 anuSeq 中给出的查询定义精确定义。定义中还给出了每个应用元素的属性数 匹配实例（属性）的数量不是由查询定义的，因此，只为属性值定义一个迭代器。如果满足以下条件，则局部列实例的值是结果集的一部分：:仅请求从'id'、'generation_parameters'、'flags'、和 'values' 本地列导出属性（主要请求其他属性，不用提交数值），本地列的所有值都属于源自 aomeasu 量的 AoMeasurement 应用元素导出的一个实例。结果集不超过服务器内存限制。ASAM ODS 服务器有内存限制；只要结果集打破了这个限制,将出现异常。其余部分不需要迭代器。如果请求的属性中包含 blob 类型的属性，返回结构的内容将取决于返回的实例的数量：假设返回一个实例的属性（可能因为'how_many'设置为 1 或服务器只有一个可用的实例），返回结果将包含 blob 自身。假设返回多个实例的属性，返回结果不包含 blobs；相反，它包含各个 blob 的头。

参数: aoq (Type = QueryStructure)

The query definition.

how_many (Type = T_LONG)

每个结果集中的最大数量，合法的形参如下：

how_many = 0 : report all instances found

how_many > 0 : report at most the given number of instances found

Java 调用序列： ElemResultSet[] elemResultSet = applElemAccess.getInstance
(aoq,how_many)。

返回：返回名称: elemResultSet

返回类型: ElemResultSetSequence

带请求属性值的结果集合。

示例：Java 语言

```
import org.asam.ods.AoException;
import org.asam.ods.AoFactory;
import org.asam.ods.AoSession;
import org.asam.ods.ApplicationAttribute;
import org.asam.ods.ApplicationElement;
import org.asam.ods.ApplicationStructure;
import org.asam.ods.ApplElemAccess;
import org.asam.ods.QueryStructure;
import org.asam.ods.AIDName;
import org.asam.ods.AIDNameUnitId;
import org.asam.ods.AIDNameValueUnitId;
import org.asam.ods.ElemId;
```



```

import org.asam.ods.ElemResultSet;
import org.asam.ods.SelOpcode;
import org.asam.ods.SelOperator;
import org.asam.ods.SelValue;
import org.asam.ods.TS_Value;
import org.asam.ods.TS_Union;
import org.asam.ods.T_LONGLONG;
public class UsingApplElemAccess {
    public static void showElemResultSet (ElemResultSet elemRes[]) {
        for (int i = 0; i < elemRes.length; i++) {
            System.out.println ("aid = " + elemRes[i].aid.low) ;
            for (int j = 0; j < elemRes[i].attrValues.length; j++) {
                System.out.println (elemRes[i].attrValues[j].attrValues.valName) ;
            }
            // Print the values.
        }
    }
}
// Java application entry.
public static void main (String[] args) {
    try {
        // Establish a session to the service (without session
        options) .
        AoSession aoSession = aoFactory.newSession ("" ) ;
        // Get the application model.
        ApplicationStructure as =
        aoSession.getApplicationStructure ( ) ;
        // Get the appl elem access object
        ApplElemAccess aea = aoSession.getApplElemAccess ( ) ;
        // Create the query structure for the method getInstances
        QueryStructure aoq;
        // The application attribute.
        ApplicationAttribute aaObj;
        // Result of the request
        ElemResultSet elemRes[];
        // Selectvalue of the Id
        T_LONGLONG iid;
        // Query on the application element AoTest
    }
}

```

```

ApplicationElement ae[] = as.getElementsByBaseType (
    "AoTest") ;
ApplicationElement aeObj = null;
if (ae.length > 0)
{
    aeObj = ae[0];
    /* Report attribute Id, Name of the instances of
    AoTest with
    * id < 10 and name like "ZYK*" and reference to
    AoSubTest with Id 1"
    */
    aoq = new QueryStructure ( ) ;
    aaObj = aeObj.getAttributeByBaseName ("id") ;
    aoq.anuSeq = new AIDNameUnitId [2];
    aoq.anuSeq[0] = new AIDNameUnitId ( ) ;
    aoq.anuSeq[0].attr = new AIDName ( ) ;
    // set Id of element
    aoq.anuSeq[0].attr.aid = aeObj.getId ( ) ;
    // Set Name of attribute
    aoq.anuSeq[0].attr.aaName = aaObj.getName ( ) ;
    // Build the query, two select values.
    aoq.condSeq = new SelValue[2];
    // First select on the Id
    aaObj = aeObj.getAttributeByBaseName ("id") ;
    aoq.condSeq[0] = new SelValue ( ) ;
    aoq.condSeq[0].attr = new AIDNameValueUnitId ( ) ;
    aoq.condSeq[0].attr.attr = new AIDName ( ) ;
    aoq.condSeq[0].attr.attr.aid = aeObj.getId ( ) ;
    aoq.condSeq[0].attr.attr.aaName = aaObj.getName ( ) ;
    aoq.condSeq[0].value = new TS_Value ( ) ;
    aoq.condSeq[0].value.u = new TS_Union ( ) ;
    iid = new T_LONGLONG ( ) ;
    iid.high = 0;
    iid.low = 10;
    aoq.condSeq[0].value.u.longlongVal ( iid ) ;
    aoq.condSeq[0].oper = SelOpcode.LT;
    // Second select on the Name

```

```

aaObj = aeObj.getAttributeByBaseName ("name") ;
aoq.anuSeq[1] = new AIDNameUnitId () ;
aoq.anuSeq[1].attr = new AIDName () ;
// set Id of element
aoq.anuSeq[1].attr.aid = aeObj.getId () ;
// Set name of attribute
aoq.anuSeq[1].attr.aaName = aaObj.getName () ;
// Build the query.
aoq.condSeq[1] = new SelValue () ;
aoq.condSeq[1].attr = new AIDNameValueUnitId () ;
aoq.condSeq[1].attr.attr = new AIDName () ;
aoq.condSeq[1].attr.attr.aid = aeObj.getId () ;
aoq.condSeq[1].attr.attr.aaName = aaObj.getName () ;
aoq.condSeq[1].value = new TS_Value () ;
aoq.condSeq[1].value.u = new TS_Union () ;
aoq.condSeq[1].value.u.stringVal ("ZYK*") ;
aoq.condSeq[1].oper = SelOpcode.EQ;
// Set the operator.
aoq.operSeq = new SelOperator[1];
aoq.operSeq[0] = SelOperator.OR;
// Set the reference selection
ae = as.getElementsByBaseType ("AoSubTest") ;
ApplicationElement aeSubObj = null;
if (ae.length > 0)
{
    aeSubObj = ae[0];
    aoq.relInst = new ElemId () ;
    aoq.relInst.aid = aeSubObj.getId () ;
    aoq.relInst.iid = new T_LONGLONG () ;
    aoq.relInst.iid.low = 1;
}
System.out.println ("Start call getInstances") ;
System.out.println ("attribute Id of the instances of
AoTest with id < 10 or name like \"ZYK*\" and
reference to AoSubTest with Id 1") ;
elemRes = aea.getInstances (aoq, 100) ;
showElemResultSet (elemRes) ;

```

```

    }
    aea = null;
    as = null;
    // Close the active session.
    aoSession.close ();
} catch (AoException aoException) {
    System.err.println ("\nUsingApplElemAccess, " +
        aoException.toString () + ":" +
        "\n " + aoException.reason);
}
// Exit the application.
System.exit (0);
}
}

```

可能的异常详见 13.3.18 节如下函数:

——AO_BAD_PARAMETER;
 ——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

(9) APPLELEMACCESS_GETINSTANCEEXT

目的: 执行查询。与此接口的 GETINSTANCES 方法相比, 此方法可用于功能更强大的查询, 具有连接定义和聚合函数。所请求的不同应用元素的数量由 QueryStructureExt 字段 anuSeq 中给出的查询定义精确定。定义中还给出了每个应用元素的属性数。匹配实例 (它的属性) 的数量不是由查询定义的, 因此, 只为属性值定义一个迭代器。返回的是结构 ResultSetExt 的序列。这个序列的长度总是 1。结构中的这个元素有两个字段。

- firstElems of type ElemResultSetExtSequence
- restElems of type ElemResultSetExtSeqIterator

firstElems: firstElems 是 ElemResultSetExt 的序列, 序列的长度对应于 QueryStructureExt 的 anuSeq 中请求的不同应用元素的数量。结构 ElemResultSetExt 具有字段 aid 和 values。aid 是应用元素的 Id, values 和 structureNameValuesUnitid 的序列是属性的值, 按顺序给出每个报告的属性条目。属性的名称在字段 valName 中给出, 单位的 Id 在 unitId 字段中给出, 这是从基本元素 AoUnit 派生的 applicationelement 实例的 Id。这些值从类型为 TS_ValueSeq 的域值中给出。每个值序列都有相同的长度, 这个长度是小于等于 how_many (方法的参数)。如果没有值, 则序列的长度为 0。Name 值 SeqUnitId 序列中不同序列中具

有相同索引的值属于相同的实例元素。

restElems:

restelems 是 **elemresultsetext** 其余序列的迭代器。

RestElems 是 **ElemResultSetExt**

这个迭代器允许从结果集中加载下 **n** 个值。元素的结果与字段 **firstelems** 的存储方式相同，下一个 **n** 是名称值 **seknit** 的字段值序列的长度。不应使用 **nextone** 方法，而应使用 **n=1** 的 **nextn** 方法。方法 **getcount** 返回找到的元素总数，该数目已在 **firstelems** 中返回。方法 **reset** 将 **iterator** 设置为结果集的开头，这是第一个元素的开头。

如果满足以下条件，则本地列实例的值是结果集的一部分：

仅请求从本地列的基本属性 **'id'**、**'generation_parameters'**、**'flags'** 和 **'values'** 派生的属性（只要请求其他属性，就不会报告这些值）。

本地列的所有值都只属于从 **AOMeasurement** 派生的应用元素的一个实例。

结果集不超过服务器内存限制。一个 **asam ods** 服务器有内存限制（而不是系统内存限制）；一旦结果集打破了这个限制，就会抛出异常。

其余部分不需要迭代器。

如果请求的属性中有 **blob** 类型的属性，则返回结构的内容将取决于返回的实例数：如果只返回一个实例的属性（可能是因为“多少”设置为 **1**，或者服务器上只有一个实例可用），则返回结构包含 **blob** 本身。如果返回多个实例的属性，则返回结构不包含 **blob**；而是包含各自 **blob** 的头。参数：**aoq**（**Type = QueryStructureExt**）

查询定义。**how_many**（**Type = T_LONG**）

每个结果集中的最大实例数。有效参数为：**how_many = 0** : report all instances found

how_many > 0 : report at most the given number of instances found

Java 调用序列：**ResultSetExt[] resultSet = applElemAccess.getInstancesExt (aoq,how_many) ;**

返回：返回名称：**resultSet**

返回类型：**ResultSetExtSequence**

具有请求的属性值的结果集。可能的异常详见 13.3.18 节如下函数：

- AO_BAD_PARAMETER;**
- AO_CONNECTION_LOST;**
- AO_IMPLEMENTATION_PROBLEM;**
- AO_NOT_IMPLEMENTED;**
- AO_NO_MEMORY;**
- AO_SESSION_NOT_ACTIVE.**

(10) **APPLELEMACCESS_GETREINST**

目的：获取相关实例（ID）。此方法返回一系列相关实例。

关系名称指定了 **applStructValue** 中给定的关系。**elemid** 和 **relname** 的帮助定义了唯一的 relation 和目标应用元素。参数：**elem**（**Type = ElemId**）

原始实例。在 `rpc-api` 中，此信息存储在 `getinstrefreq` 结构的 `elemid` 字段和请求 `aop_getinstref` 字段中。`relName` (`Type = Name`)

关系名称。在 `rpc-api` 中，此信息存储在结构 `getinstrefreq` 和请求 `aop_getinstref` 的字段 `refname` 中。Java 调用序列：`T_LONGLONG[] instIds = applElemAccess.getRelInst(elem, relName);`

返回：返回名称：`instIds`

返回类型：`S_LONGLONG`

相关实例的 ID 列表。可能的异常详见 13.3.18 节如下函数：

——`AO_BAD_PARAMETER;`
 ——`AO_CONNECTION_LOST;`
 ——`AO_IMPLEMENTATION_PROBLEM;`
 ——`AO_NOT_IMPLEMENTED;`
 ——`AO_NO_MEMORY;`
 ——`AO_SESSION_NOT_ACTIVE.`

(11) APPLELEMACCESS_GETVALUEMATRIX

目的：获取度量值或子矩阵的值矩阵。如果将使用特殊的子矩阵链接建立值矩阵，则使用接口度量。

通过调用此方法，返回值矩阵的值矩阵模式将从上下文变量“`value matrix_mode`”中获取。

有关更多信息，尤其是关于值矩阵的不同模式的信息，请参见方法

“`measurement`”和“`submatrix`”接口的 `getValueMatrixinMode()` 中的描述。

参数：`elem` (`Type = ElemId`)

元素 ID。`aid` 必须是测量或子矩阵的应用元素 ID。

Java 调用序列：`ValueMatrix vm = applElemAccess.getValueMatrix(elem);`

返回：返回名称：`vm`

返回类型：`ValueMatrix`

The value matrix.

可能的异常详见 13.3.18 节如下函数：

——`AO_BAD_PARAMETER;`
 ——`AO_CONNECTION_LOST;`
 ——`AO_IMPLEMENTATION_PROBLEM;`
 ——`AO_INVALID_BASETYPE;`
 ——`AO_NOT_IMPLEMENTED;`
 ——`AO_NO_MEMORY;`
 ——`AO_SESSION_NOT_ACTIVE.`

(12) APPLELEMACCESS_GETVALUEMATRIXINMODE

目的：

获取度量值或子矩阵的值矩阵。如果将使用特殊的子矩阵链接建立值矩阵,则使用接口度量。
ASAMODS 服务器的行为取决于引用的元素是子矩阵还是度量。有关服务器行为的详细信息,请
参阅接口“measurement”和“submatrix”的 `getValueMatrixinMode()` 方法中的描述。

参数: `elem` (Type = `ElemId`)

元素 ID。`aid` 必须是测量或子矩阵的应用元素 ID。`vmMode` (Type = `ValueMatrixMode`)
值矩阵的请求模式。Java 调用序列: `ValueMatrix vm = applElemAccess.getValueMatrix(elem,
vmMode);`

返回: 返回名称: `vm`

返回类型: `ValueMatrix`

价值矩阵。可能的异常详见 13.3.18 节如下函数:

——`AO_BAD_PARAMETER;`
——`AO_CONNECTION_LOST;`
——`AO_IMPLEMENTATION_PROBLEM;`
——`AO_INVALID_BASETYPE;`
——`AO_INVALID_VALUEMATRIX_STRUCTURE;`
——`AO_NOT_IMPLEMENTED;`
——`AO_NO_MEMORY;`
——`AO_SESSION_NOT_ACTIVE.`

(13) APPLELEMACCESS_INSERTINSTANCES

目的:

创建应用元素的实例。应用元素由输入结构的辅助指定。属性名由输入结构的名称指定。一个实例的值在输入结构的 `valueseq` 中指定。通过填充输入结构的 `valueseq`, 可以在一次调用中创建多个实例。`valueseq` 中的相同索引对应于一个实例的属性值。此方法返回 `seq` 在 `ID` 的影响下, 顺序与输入中指定的实例顺序相关。在继承的情况下, 该方法只支持每次调用的 `SameSubType` 实例。返回的 `ID` 是相关父类型实例的 `ID`。

客户机必须在一个方法调用中提供所有必需的属性和引用; 否则, 服务器无法使数据库中的对象持久化, 而不会有违反任何数据库约束的风险。参数: `val` (Type = `AIDNameValueSeqUnitIdSequence`)

属性序列及其值。在 `rpc-api` 中, 此信息存储在

结构 `putvalreq` 和 `requestaop putvalreq` 的字段 `elemid` 和 `nvseq`。Java 调用序列:

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

返回: 返回名称: `elemIds`

返回类型: `ElemIdSequence`

List with the Ids of the newly created instances.

可能的异常详见 13.3.18 节如下函数:

——`AO_BAD_PARAMETER;`
——`AO_CONNECTION_LOST;`

——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

(14) APPLELEMACCESS_SETATTRIBUTERIGHTS

目的：设置由<aid>和<attr_name>定义的某些应用程序属性的 ACL 信息。<usergroup_id>定义应该为其设置权限的用户组。<rights>定义要设置或清除的权限。如果参数<set>设置为'set'，则将设置<rights>中的权限，其他所有权限都将被清除。如果参数<set>设置为'add'，则将<rights>中的权限添加到现有权限中。如果参数<set>设置为'remove'，则将从现有权限中删除<rights>中的权限。对模型的限制：只允许 aoUserGroup 类型的一个应用元素。

参数：aid (Type = T_LONGLONG)

应用元素的 ID。attrName (Type = T_STRING)

属性的名称。usergroupId (Type = T_LONGLONG)

要为其设置权限的用户组。rights (Type = T_LONG)

用户组的新权限。set (Type = RightsSet)

如何处理新的权利。Java 调用序列：applElemAccess.setAttributeRights

(aid,attrName,usergroupId,rights,set) ;

返回：None。

可能的异常详见 13.3.18 节如下函数：

——AO_BAD_PARAMETER;
 ——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

(15) APPLELEMACCESS_SETELEMENTINITIALRIGHTS

目的：为<aid>定义的某些应用元素的初始权限设置访问控制列表信息。<usergroup_id>定义应该为其设置权限的用户组。<rights>定义要设置或清除的权限。如果参数<set>设置为'set'，则设置<rights>中的权限，清除所有其他权限。如果参数<set>设置为'add'，则将<rights>中的权限添加到现有权限中。如果参数<set>设置为'remove'，则将从现有权限中删除<rights>中的权限。对模型的限制：只允许类型为 AOUserGroup 的一个应用元素。

参数：aid (Type = T_LONGLONG)

应用元素的 ID。usergroupId (Type = T_LONGLONG)

要为其设置初始权限的用户组。rights (Type = T_LONG)

用户组的新初始权限。权限常量在接口 SecurityRights 中定义。接口定义语言 IDL 不允许设置

枚举值，因此必须在接口中完成常量定义。refAid (Type = T_LONGLONG)

将使用初始权限的引用应用元素的 ID。如果没有设置 refaid，则独立于 application 元素的每个新实例都将使用初始权限。set (Type = RightsSet)

如何处理新的初始权利。Java 调用序列：applElemAccess.setElementInitialRights (aid,usergroupId,rights,refAid,set) ;

返回：None.

可能的异常详见 13.3.18 节如下函数：

_AO_BAD_PARAMETER;
——AO_CONNECTION_LOST;
——AO_IMPLEMENTATION_PROBLEM;
——AO_NOT_IMPLEMENTED;
——AO_NO_MEMORY;
——AO_SESSION_NOT_ACTIVE.

(16) APPLELEMACCESS_SETELEMENTRIGHTS

目的：设置由<aid>定义的某个应用元素的 ACL 信息。<usergroup_id>定义应该为其设置权限的用户组。<rights>定义要设置或清除的权限。如果参数<set>设置为'set'，则设置<rights>中的权限，清除所有其他权限。如果参数<set>设置为'add'，则将在<rights>中的权限添加到现有权限中。如果参数<set>设置为'remove'，则将从现有权限中删除<rights>中的权限。对模型的限制：只允许类型为 AOUserGroup 的一个应用元素。

参数：aid (Type = T_LONGLONG)

应用元素的 ID。usergroupId (Type = T_LONGLONG)

要为其设置权限的用户组。rights (Type = T_LONG)

用户组的新权限。权限常量在接口 SecurityRights 中定义。接口定义语言 IDL 不允许设置枚举值，因此必须在接口中完成常量定义。set (Type = RightsSet)

如何处理新的权利。

Java 调用序列：applElemAccess.setElementRights (aid,usergroupId,rights,set) ;

返回：None.

可能的异常详见 13.3.18 节如下函数：

——AO_BAD_PARAMETER;
——AO_CONNECTION_LOST;
——AO_IMPLEMENTATION_PROBLEM;
——AO_NOT_IMPLEMENTED;
——AO_NO_MEMORY;
——AO_SESSION_NOT_ACTIVE.

(17) APPLELEMACCESS_SETINITIALRIGHTREFERENCE

目的：如果将使用（或不使用）此引用来检索初始权限，请在 `svcatr` 中设置标志 `<set>`。如果多个引用设置为 `true`，则使用所有权利的联合（或函数）。

参数：aid（Type = T_LONGLONG）

应用元素 ID。refName（Type = T_STRING）

引用的名称。set（Type = RightsSet）

如何处理给定的引用。Java 调用序列：`applElemAccess.setInitialRightReference`
（aid,refName,set）；

返回：None。

可能的异常详见 13.3.18 节如下函数：

- AO_BAD_PARAMETER;
- AO_CONNECTION_LOST;
- AO_IMPLEMENTATION_PROBLEM;
- AO_NOT_IMPLEMENTED;
- AO_NO_MEMORY;
- AO_SESSION_NOT_ACTIVE。

（18）APPLELEMACCESS_SETINSTANCEINITIALRIGHTS

目的：为应用元素 `id<a id>` 定义的某些实例的初始权限和 `id<iid>` 定义的实例序列设置访问控制列表信息。`<usergroup_id>` 定义应该为其设置权限的用户组。`<rights>` 定义要设置或清除的权限。如果参数 `<set>` 设置为 'set'，则设置 `<rights>` 中的权限，清除所有其他权限。如果参数 `<set>` 设置为 'add'，则将 `<rights>` 中的权限添加到现有权限中。如果参数 `<set>` 设置为 'remove'，则将从现有权限中删除 `<rights>` 中的权限。对模型的限制：只允许类型为 `AOUserGroup` 的一个应用元素。参数：aid（Type = T_LONGLONG）

应用元素的 ID。instIds（Type = S_LONGLONG）

实例 ID 的序列。usergroupId（Type = T_LONGLONG）

要为其设置初始权限的用户组。rights（Type = T_LONG）

用户组的新初始权限。refAid（Type = T_LONGLONG）

将使用初始权限的引用应用元素的 ID。如果没有设置 `refaid`，则独立于 `application` 元素的每个新实例都将使用初始权限。set（Type = RightsSet）

指定如何处理新的初始权限 Java 调用序列：`applElemAccess.setInstanceInitialRights`
（aid,instIds,usergroupId,rights,refAid,set）；

返回：None。

可能的异常详见 13.3.18 节如下函数：

- AO_BAD_PARAMETER;
- AO_CONNECTION_LOST;
- AO_IMPLEMENTATION_PROBLEM;
- AO_NOT_IMPLEMENTED;

——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

(19) APPLELEMACCESS_SETINSTANCERIGHTS

目的：在应用元素 `id<a id>` 定义的某个实例和 `id<iid>` 定义的实例序列上设置 `acl` 信息。

`<usergroup_id>` 定义应该为其设置权限的用户组。`<rights>` 定义要设置或清除的权限。如果参数 `<set>` 设置为 'set'，则设置 `<rights>` 中的权限，清除所有其他权限。如果参数 `<set>` 设置为 'add'，则将 `<rights>` 中的权限添加到现有权限中。如果参数 `<set>` 设置为 'remove'，则将从现有权限中删除 `<rights>` 中的权限。对模型的限制：只允许类型为 `AOUserGroup` 的一个应用元素。

参数：aid (Type = T_LONGLONG)

应用元素的 ID。instIds (Type = S_LONGLONG)

实例 ID 的序列。usergroupId (Type = T_LONGLONG)

要为其设置权限的用户组。rights (Type = T_LONG)

用户组的新权限。set (Type = RightsSet)

如何处理新的权利。Java 调用序列：`applElemAccess.setInstanceRights (aid,instIds,usergroupId,rights,set)`;

返回：None.

可能的异常详见 13.3.18 节如下函数：

——AO_BAD_PARAMETER;
 ——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

(20) APPLELEMACCESS_SETREINST

目的：设置实例引用。

参数：elem (Type = ElemId)

要添加相关实例的实例。在 `rpc-api` 中，此信息存储在结构 `setinstrefreq` 和请求 `aop_setinstref.relname (type=name)` 的字段 `elemid1` 中。关系的名称。在 `rpc-api` 中，此信息存储在结构 `setinstrefreq` 和请求 `aop_setinstref` 的字段 `refname` 中。instIds (Type = S_LONGLONG)

实例 ID 的序列。在 `rpc-api` 中，此信息存储在结构 `setinstrefreq` 和请求 `aop_setinstref` 的字段 `elemid2` 中。不能建立多个关系。type (Type = SetType)

修改、插入、更新或删除的类型。在 `rpc-api` 中，此信息存储在结构 `setinstrefreq` 和请求 `aop_setinstref` 的 `onoff` 字段中。Java 调用序列：`applElemAccess.setRelInst (elem,relName,instIds,type)`;

返回: None.

可能的异常详见 13.3.18 节如下函数:

——AO_BAD_PARAMETER;
 ——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

(21) APPLELEMACCESS_UPDATEINSTANCES

目的: 更新一个或多个实例的一个或多个属性。输入结构还必须包含 **id** 属性; 它将用于选择实例。

如果是继承的应用元素, 则必须包含父类型 **ID**。一个实例的值在输入结构的 **valueseq** 中指定。**valueseq** 中的相同索引对应于一个实例的属性值。

如果满足以下条件, **ASAM ODS** 服务器将更新本地列实例的值: 只提供从本地列的基本属性 '**id**'、'**generation_-parameters**'、'**flags**'和'**values**'派生的属性 (只要提供其他属性, 就会引发异常)。

本地列的所有值只属于从 **AOMeasurement** 派生的应用元素的一个实例。

给定值的数量不超过服务器内存限制。**asamods** 服务器有内存限制 (而不是系统内存限制); 一旦给定的值打破这个限制, 就会抛出异常。参数: **val** (**Type** = **AIDNameValueSeqUnitIdSequence**)

属性序列及其值。至少一个属性值

序列必须是 **ID** 为的序列。在 **rpc-api** 中, 此信息存储在

结构 **putvalreq** 和 **requestaop putvalreq** 的字段 **elemid** 和 **nvseq**。Java 调用序列:
applElemAccess.updateInstances (val) ;

返回: None.

可能的异常详见 13.3.18 节如下函数:

——AO_BAD_PARAMETER;
 ——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

13. 2. 4 APPLICATIONATTRIBUTE

(1) APPLICATIONATTRIBUTE_GETAPPLICATIONELEMENT

目的: 返回属性所属的应用元素。

参数: None.

Java 调用序列: `ApplicationElement applElem = applAttr.getApplicationElement ( )`。

返回: 返回名称: `applElem`

返回类型: `ApplicationElement`

属性的应用元素。可能的异常详见 13.3.18 节如下函数:

- `AO_BAD_PARAMETER`;
- `AO_CONNECTION_LOST`;
- `AO_IMPLEMENTATION_PROBLEM`;
- `AO_NOT_IMPLEMENTED`;
- `AO_NO_MEMORY`;
- `AO_SESSION_NOT_ACTIVE`。

(2) APPLICATIONATTRIBUTE_GETBASEATTRIBUTE

目的: 获取应用程序属性的基属性。

参数: `None`。

Java 调用序列: `BaseAttribute baseAttr = applAttr.getBaseAttribute ( )` ;

返回: 返回名称: `baseAttr`

返回类型: `BaseAttribute`

应用程序属性的基本属性。如果应用程序属性没有基属性, 则返回“`null`”。可能的异常详见 13.3.18 节如下函数:

- `AO_CONNECTION_LOST`;
- `AO_IMPLEMENTATION_PROBLEM`;
- `AO_NOT_IMPLEMENTED`;
- `AO_NO_MEMORY`;
- `AO_SESSION_NOT_ACTIVE`。

(3) APPLICATIONATTRIBUTE_GETDATATYPE

目的: 获取应用程序属性的数据类型。

参数: `None`。

Java 调用序列: `DataType aaDataType = applAttr.getDataType ( )` ;

返回: 返回名称: `aaDataType`

返回类型: `DataType`

应用程序属性的数据类型。可能的异常详见 13.3.18 节如下函数:

- `AO_CONNECTION_LOST`;
- `AO_IMPLEMENTATION_PROBLEM`;
- `AO_NOT_IMPLEMENTED`;
- `AO_NO_MEMORY`;
- `AO_SESSION_NOT_ACTIVE`。

(4) APPLICATIONATTRIBUTE_GETENUMERATIONDEFINITION

目的：获取枚举的定义。

参数：None.

Java 调用序列：EnumerationDefinition enumDef;

enumDef = applAttr.getEnumerationDefinition () ;

返回：返回名称：enumDef

返回类型：EnumerationDefinition

ASAM ODS 枚举。可能的异常详见 13.3.18 节如下函数：

——AO_BAD_PARAMETER;
——AO_CONNECTION_LOST;
——AO_IMPLEMENTATION_PROBLEM;
——AO_NOT_IMPLEMENTED;
——AO_NO_MEMORY;
——AO_SESSION_NOT_ACTIVE;
——AO_INVALID_DATATYPE。

(5) APPLICATIONATTRIBUTE_GETLENGTH

目的：获取应用程序属性值的最大允许长度。对于第 3 章中指定的物理存储，最大允许长度取决于 svcattr 中的 aflen 值的设置，该值受到限制，例如字符串类型数据库字段的最大长度。
asam ods 不支持无限字符串。

参数：None.

Java 调用序列：T_LONG aaLength = applAttr.getLength () ;

返回：返回名称：aaLength

返回类型：T_LONG

应用程序属性允许的最大长度。可能的异常详见 13.3.18 节如下函数：

——AO_CONNECTION_LOST;
——AO_IMPLEMENTATION_PROBLEM;
——AO_NOT_IMPLEMENTED;
——AO_NO_MEMORY;
——AO_SESSION_NOT_ACTIVE。

(6) APPLICATIONATTRIBUTE_GETNAME

目的：获取应用程序属性的名称。

参数：None.

Java 调用序列：Name aaName = applAttr.getName () ;

返回：返回名称：aaName

返回类型: Name

应用程序属性的名称。可能的异常详见 13.3.18 节如下函数:

——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE;

(7) APPLICATIONATTRIBUTE_GETRIGHTS

目的: 检索给定对象的访问控制列表信息。

参数: None.

Java 调用序列: `ACL[] aclEntries = applAttr.getRights ( ) ;`

返回: 返回名称: `aclEntries`

返回类型: `ACLSequence`

给定应用元素的访问控制列表项。可能的异常详见 13.3.18 节如下函数:

——AO_BAD_PARAMETER;
 ——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

(8) APPLICATIONATTRIBUTE_GETUNIT

目的: 获取应用程序属性的单元 ID。单位 ID 仅对当前服务器有效。

参数: None.

Java 调用序列: `T_LONGLONG aaUnit = applAttr.getUnit ( ) ;`

返回: 返回名称: `aaUnit`

返回类型: `T_LONGLONG`

应用程序属性的单位 ID。可能的异常详见 13.3.18 节如下函数:

——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

(9) APPLICATIONATTRIBUTE_HASUNIT

目的: 属性为单位。如果设置了此标志, 则从该属性派生的实例的所有属性都将具有一个单元。

参数: None.

Java 调用序列: T_BOOLEAN hasUnit = aaObj.hasUnit ();

返回: 返回名称: hasUnit

返回类型: T_BOOLEAN

属性有单位时的标志。可能的异常详见 13.3.18 节如下函数:

——AO_BAD_PARAMETER;
——AO_CONNECTION_LOST;
——AO_IMPLEMENTATION_PROBLEM;
——AO_NOT_IMPLEMENTED;
——AO_NO_MEMORY;
——AO_SESSION_NOT_ACTIVE。

(10) APPLICATIONATTRIBUTE_ISAUTOGENERATED

目的: 获取应用程序属性的自动生成标志。

参数: None.

Java 调用序列: T_BOOLEAN isAutogenerated = applAttr.isAutogenerated ();

返回: 返回名称: isAutogenerated

返回类型: T_BOOLEAN

应用程序属性的自动生成标志。可能的异常详见 13.3.18 节如下函数:

——AO_CONNECTION_LOST;
——AO_IMPLEMENTATION_PROBLEM;
——AO_NOT_IMPLEMENTED;
——AO_NO_MEMORY;
——AO_SESSION_NOT_ACTIVE。

(11) APPLICATIONATTRIBUTE_ISOBLIGATORY

目的: 获取应用程序属性的强制标志。

参数: None.

Java 调用序列: T_BOOLEAN aalsObligatory = applAttr.isObligatory ();

返回: 返回名称: aalsObligatory

返回类型: T_BOOLEAN

应用程序属性的强制标志。可能的异常详见 13.3.18 节如下函数:

——AO_CONNECTION_LOST;
——AO_IMPLEMENTATION_PROBLEM;
——AO_NOT_IMPLEMENTED;
——AO_NO_MEMORY;
——AO_SESSION_NOT_ACTIVE。

(12) APPLICATIONATTRIBUTE_ISUNIQUE

目的：获取应用程序属性的唯一标志。

参数：None.

Java 调用序列：T_BOOLEAN aalsUnique = applAttr.isUnique () ;

返回：返回名称：aalsUnique

返回类型：T_BOOLEAN

应用程序属性的唯一标志。可能的异常详见 13.3.18 节如下函数：

——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

(13) APPLICATIONATTRIBUTE_SETBASEATTRIBUTE

目的：设置应用程序属性的基本属性。这允许客户端声明附加到基属性的应用程序属性（新的或现有的）。应用程序属性将成为给定基属性的派生属性。

允许在事务外部修改应用程序属性，但建议激活事务。

基属性在应用元素中必须是唯一的，否则将引发异常 `ao_duplicate_base_` 属性。出于性能和灵活性的原因，应在首次提交新的应用程序属性之前使用此设置方法。

如果在第一次提交之前调用此方法，它将不会引发以下异常：AO_INVALID_DATATYPE

AO_MISSING_VALUE

AO_NOT_UNIQUE.

在第一次提交之后，可能存在应用程序属性的实例。这些情况可能导致以下问题：

AO_INVALID_DATATYPE：基本属性的数据类型与已声明属性的数据类型不同。

AO_MISSING_VALUE：已设置基属性的强制标志，但实例中有一个或多个空值。

AO_NOT_UNIQUE：已设置基属性的唯一标志，但实例的值不唯一。

应用程序属性的长度、名称和单位不受此调用的影响。参数：baseAttr (Type = BaseAttribute)

基本属性。Java 调用序列：applAttr.setBaseAttribute (baseAttr) ;

返回：None

可能的异常详见 13.3.18 节如下函数：

——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。

(14) APPLICATIONATTRIBUTE_SETDATATYPE

目的：设置应用程序属性的数据类型。

允许在事务外部修改应用程序属性，但建议激活事务。不允许设置表示基属性的应用程序属性的数据类型。尝试设置此类应用程序属性的数据类型将导致异常 `ao_is_base_` 属性。出于性能和灵活性的原因，应在首次提交新的 `applicationattribute` 之前使用此 `set` 方法。如果在第一次提交之前调用此方法，它将不会引发以下异常：`AO_INVALID_DATATYPE`；

在第一次提交之后，可能存在应用程序属性的实例。这些情况可能导致以下问题：

`AO_INVALID_DATATYPE`：基本属性的数据类型与已声明属性的数据类型不同。参数：
`aaDataType` (`Type = DataType`)

数据类型。Java 调用序列：`applAttr.setDataType(aaDataType)`；

返回：`None`。

可能的异常详见 13.3.18 节如下函数：

——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。

(15) APPLICATIONATTRIBUTE_SETENUMERATIONDEFINITION

目的：设置枚举的定义。此方法修改应用程序模型，只有超级用户才能使用此方法。如果属性的数据类型不是 `dt_enum`，则抛出异常--`ao_invalid_data type`；。

参数：`enumDef` (`Type = EnumerationDefinition`)

新的枚举定义。Java 调用序列：`applAttr.setEnumerationDefinition(enumDef)`；

返回：`None`。

可能的异常详见 13.3.18 节如下函数：

——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。

(16) APPLICATIONATTRIBUTE_SETISAUTOGENERATED

目的:

设置应用程序属性的自动生成标志。

允许在事务外部修改应用程序属性，但建议激活事务。出于性能和灵活性的原因，应在首次提交新的 `applicationattribute` 之前使用此 `set` 方法。参数: `isAutogenerated` (Type = T_BOOLEAN)
 自动生成标志。Java 调用序列: `applAttr.setIsAutogenerated (isAutogenerated)` ;

返回: None.

可能的异常详见 13.3.18 节如下函数:

——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。

(17) APPLICATIONATTRIBUTE_SETISOBLIGATORY

目的: 设置应用程序属性的强制标志。

允许在事务外部修改应用程序属性，但建议激活事务。允许在事务外部修改应用程序属性，但建议激活事务。不允许设置表示基本属性的应用程序属性的强制标志。尝试设置此类应用程序属性的强制标志将导致异常 `AO_IS_BASE_ATTRIBUTE` 属性。出于性能和灵活性的原因，应在首次提交新的 `applicationattribute` 之前使用此 `set-method`。如果在第一次提交之前调用此方法，它将不会引发以下异常: `AO_MISSING_VALUE`;

在第一次提交之后，可能存在应用程序属性的实例。这些实例可能导致以下问题:

`AO_MISSING_VALUE`: 已设置基属性的强制标志，但实例中有一个或多个空值。

参数: `aalsObligatory` (Type = T_BOOLEAN)

强制性标志。Java 调用序列: `applAttr.setIsObligatory (aalsObligatory)` ;

返回: None.

可能的异常详见 13.3.18 节如下函数:

——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。

(18) APPLICATIONATTRIBUTE_SETISUNIQUE

目的：设置应用程序属性的唯一标志。

允许在事务外部修改应用程序属性，但建议激活事务。

服务器将检查实例属性的值是否唯一。如果设置了此标志，并且使用方法 `setValue` 时 `attributeAre` 的值不唯一，则引发异常。如果已经存在包含唯一值的应用元素实例，并且应设置该标志，则此方法将引发异常。不允许设置表示基本属性的应用程序属性的唯一标志。尝试设置此类应用程序属性的唯一标志将导致异常 `AO_IS_BASE_ATTRIBUTE`。属性。如果 `unique` 标志设置为 `TRUE`，则强制标志也设置为 `TRUE`。在这种情况下，两个标志的先前值并不重要。将 `unique` 标志设置为 `FALSE` 不会影响强制标志。出于性能和灵活性的原因，应在首次提交新的 `applicationattribute` 之前使用此 `set` 方法。如果在第一次提交之前调用此方法，它将不会引发以下异常：`AO_MISSING_VALUE`;

`AO_NOT_UNIQUE`;

在第一次提交之后，可能存在应用程序属性的实例。这些实例可能导致以下问题：

`AO_MISSING_VALUE`：：已设置基属性的强制标志，但实例中有一个或多个空值。

`AO_NOT_UNIQUE`：：已设置基属性的唯一标志，但实例的值不唯一。

参数：`aalsUnique` (Type = `T_BOOLEAN`)

唯一标志。Java 调用序列：`applAttr.setIsUnique(aalsUnique)`;

返回：`None`。

可能的异常详见 13.3.18 节如下函数：

——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。

(19) APPLICATIONATTRIBUTE_SETLENGTH

目的：设置应用程序属性允许的最大长度。ASAM ODS 服务器接受的此长度信息的最大值取决于基础物理存储。对于第 3 章中指定的物理存储，最大允许长度将存储在 SVCATTR 中的 AFLEN 值中，并受字符串类型数据库字段的最大长度限制。如果指定的长度超过基础物理存储的能力，则服务器将抛出异常的无效长度。ASAM ODS 不支持无限字符串。允许在事务外部修改应用程序属性，但建议激活事务。

此方法对 ODS 数据库设计工具非常有用。不允许使用负长度值。此方法只向设计阶段的数据库服务器提供一个提示，用于调整数据项的大小。长度仅与数据类型为 DT_STRING, DS_STRING, DT_DATE, DS_DATE, DT_EXTERNALREFERENCE 和 DS_EXTERNALREFERENCE 的所有组件以及 DT_BLOB 的头部分的属性值相关。出于性能和灵活性的原因，应在首次提交新的应用程序属性之前使用此设置方法。

如果在第一次提交之前调用此方法，它将不会引发以下异常：AO_HAS_INSTANCES 在第一次提交之后，可能存在应用程序属性的实例。如果应用程序属性的实例不是空的，这些实例可能会导致异常 AO_HAS_INSTANCES。参数：aaLength (Type = T_LONG)

最大属性长度。Java 调用序列：applAttr.setLength (aaLength) ;

返回：None.

可能的异常详见 13.3.18 节如下函数：

——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.

(20) APPLICATIONATTRIBUTE_SETNAME

目的：设置应用程序属性的名称。

允许在事务外部修改应用程序属性，但建议激活事务。名称必须唯一。出于性能和灵活性的原因，应在首次提交新的 applicationattribute 之前使用此 set 方法。

应用程序属性的名称不能超过基础物理存储的最大名称长度。物理存储的当前规范将其限制为 30 个字符。参数：aaName (Type = Name)

应用程序属性名称。Java 调用序列：applAttr.setName (aaName) ;

返回：None.

可能的异常详见 13.3.18 节如下函数：

——AO_BAD_PARAMETER;

——AO_CONNECTION_LOST;
 ——AO_DUPLICATE_NAME;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

(21) APPLICATIONATTRIBUTE_SETRIGHTS

目的: 应为其设置权限的给定用户组。<rights>定义要设置或清除的权限。如果参数<set>设置为'set', 则设置<rights>中的权限, 清除所有其他权限。如果参数<set>设置为'add', 则将<rights>中的权限添加到现有权限中。如果参数<set>设置为'remove', 则将从现有权限中删除<rights>中的权限。

参数: usergroup (Type = InstanceElement)

将修改其权限的用户组。rights (Type = T_LONG)

用户组的新权限。权限常量在接口 SecurityRights 中定义。接口定义语言 IDL 不允许设置枚举值, 因此必须在接口中完成常量定义。set (Type = RightsSet)

如何处理新的权利。Java 调用序列: applAttr.setRights (usergroup, rights, set);

返回: None.

可能的异常详见 13.3.18 节如下函数:

——AO_BAD_PARAMETER;
 ——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE;
 ——AO_TRANSACTION_NOT_ACTIVE。

(22) APPLICATIONATTRIBUTE_SETUNIT

目的: 设置应用程序属性的单元 ID。

允许在事务外部修改应用程序属性, 但建议激活事务。单位 ID 仅对当前服务器有效。如果存在应用程序属性的实例, 则相应的值将自动转换为新单元。如果没有已知的转换, 则会引发异常。如果单位设置为零, 可以避免自动转换。之后, 必须提交事务。在下一步中, 可以在另一个事务中设置新单元。仅对以下数据类型执行自动转换: DT_BYTE

DT_COMPLEX
 DT_DCOMPLEX
 DT_DOUBLE
 DT_FLOAT
 DT_LONG

DT_LONGLONG

DT_SHORT

以及相应的序列数据类型。对于复杂数据类型，实部和虚部分别转换。如果设置了属性的单位，则该单位为常量。如果属性值具有其他单位，则该值将重新计算为应用程序属性的单位。如果应用程序属性中没有单位，那么属性值中的单位将被存储，并根据请求在实例中报告。出于性能和灵活性的原因，应在首次提交新的应用程序属性之前使用此设置方法。

如果在第一次提交之前调用此方法，它将不会引发以下异常：

AO_INCOMPATIBLE_UNITS

AO_MATH_ERROR

在第一次提交之后，可能存在应用程序属性的实例。这些情况可能导致以下问题：

AO_INCOMPATIBLE_UNITS：没有已知的转换规则来转换单位。

AO_MATH_ERROR：将值转换为新的单位会导致数据溢出或下溢，或者检测到被零除。

参数：aaUnit（Type = T_LONGLONG）

单位 ID。Java 调用序列：applAttr.setUnit（aaUnit）；

返回：None。

可能的异常详见 13.3.18 节如下函数：

——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。

(23) APPLICATIONATTRIBUTE_WITHUNIT

目的：指定属性是否具有单位。对方法 setUnit（）的调用将自动将 withUnit 标志设置为 true。

参数：withUnit（Type = T_BOOLEAN）

标志值：TRUE，如果属性将有一个单位，则为 FALSE。

Java 调用序列：aaObj.withUnit（TRUE）；

返回：None。

可能的异常详见 13.3.18 节如下函数：

——AO_BAD_PARAMETER；
 ——AO_CONNECTION_LOST；
 ——AO_IMPLEMENTATION_PROBLEM；
 ——AO_NOT_IMPLEMENTED；

——AO_NO_MEMORY;
——AO_SESSION_NOT_ACTIVE;
——AO_TRANSACTION_NOT_ACTIVE。

13.2.5 APPLICATIONELEMENT

(1) APPLICATIONELEMENT_CREATEATTRIBUTE

目的：在服务器上创建新的应用程序属性。

允许在事务外部修改应用元素，但建议激活事务。

可以通过 **application attribute** 接口的 **set** 方法更改新应用程序属性的属性。出于性能原因，建议在首次提交前设置应用程序属性的所有必需属性。这就避免了每个属性使用数据库 cross-checks。新应用程序属性的默认属性为：**BaseAttribute NULL**

DataType	DT_UNKNOWN
IsObligatory	0
IsUnique	0
Length	0
Name	"AUTOGEN"
Unit	NULL

如果已经存在应用元素的实例，则新属性的现有实例的值未定义。如果已经存在另一个名为

“**AUTOGEN**”的应用程序属性，则会出现异常 **AO_DUPLICATE_NAME**。参数：**None**。

Java 调用序列：**ApplicationAttribute applAttr = applElem.createAttribute () ;**

返回：返回名称：**applAttr**

返回类型：**ApplicationAttribute**

新的应用程序属性。可能的异常详见 13.3.18 节如下函数：

——AO_CONNECTION_LOST;
——AO_DUPLICATE_NAME;
——AO_IMPLEMENTATION_PROBLEM;
——AO_NOT_IMPLEMENTED;
——AO_NO_MEMORY;
——AO_SESSION_NOT_ACTIVE;
——AO_TRANSACTION_NOT_ACTIVE。

(2) APPLICATIONELEMENT_CREATEINSTANCE

目的：创建应用元素的实例。

允许在事务外部创建实例，但建议激活事务。当事务被提交时，实例将获得永久性。所有连接到应用元素的属性都会自动创建并连接到实例。属性的值可以通过接口 **instanceelement** 的方法 **setvalue** 设置。参数：**ieName (Type = Name)**

实例名称。Java 调用序列：**InstanceElement instElem = applElem.createInstance (ieName) ;**

返回：返回名称：instElem

返回类型：InstanceElement

新实例。可能的异常详见 13.3.18 节如下函数：

——AO_BAD_PARAMETER;
 ——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE;
 ——AO_TRANSACTION_NOT_ACTIVE。

(3) APPLICATIONELEMENT_CREATEINSTANCES

目的：创建包含实例的列表。属性以序列的名称给出。属性的值在值序列中给出。不同值序列中的索引与一个实例匹配。相关实例实例序列中的索引与值序列中具有相同索引的实例匹配。

参数：attributes (Type = NameValueSeqUnitSequence)

新 创 建 实 例 的 属 性 。 relatedInstances (Type = ApplicationRelationInstanceElementSeqSequence)

具有不同应用程序关系的相关实例的列表。Java 调用序列：InstanceElement[] instElems = applElem.createInstances (attributes, relatedInstances) ;

返回：返回名称：instElems

返回类型：InstanceElementSequence

The new instances.

可能的异常详见 13.3.18 节如下函数：

——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) APPLICATIONELEMENT_GETALLRELATEDELEMENTS

目的：获取连接到此应用元素的所有相关应用元素的列表。参数：None.

Java 调用序列：ApplicationElement[] applElems = applElem.getAllRelatedElements () ;

返回：返回名称：applElems

返回类型：ApplicationElementSequence

相关的应用元素。可能的异常详见 13.3.18 节如下函数：

- AO_CONNECTION_LOST;
- AO_IMPLEMENTATION_PROBLEM;
- AO_NOT_IMPLEMENTED;
- AO_NO_MEMORY;
- AO_SESSION_NOT_ACTIVE。

(5) APPLICATIONELEMENT_GETALLRELATIONS

目的：获取连接到此应用元素的所有应用程序关系的列表。与指向给定应用元素的其他应用元素连接的关系的逆关系不会返回。

参数：None.

Java 调用序列：ApplicationRelation[] applRels = applElem.getAllRelations () ;

返回：返回名称：applRels

返回类型：ApplicationRelationSequence

应用元素的应用关系。可能的异常详见 13.3.18 节如下函数：

- AO_CONNECTION_LOST;
- AO_IMPLEMENTATION_PROBLEM;
- AO_NOT_IMPLEMENTED;
- AO_NO_MEMORY;
- AO_SESSION_NOT_ACTIVE。

(6) APPLICATIONELEMENT_GETAPPLICATIONSTRUCTURE

目的：通过返回具有接口 applicationstructure 的对象，获取应用元素所属的应用程序模型。提供的应用程序模型与 InterfaceAsession 的 getApplicationStructure 方法提供的应用程序模型相同。此方法确保客户端软件能够在会话对象不可用时返回会话。

参数：None.

Java 调用序列：ApplicationStructure applStruct = applElem.getApplicationStructure () ;

返回：返回名称：applStruct

返回类型：ApplicationStructure

应用元素所属的应用程序模型。可能的异常详见 13.3.18 节如下函数：

- AO_BAD_PARAMETER;
- AO_CONNECTION_LOST;
- AO_IMPLEMENTATION_PROBLEM;
- AO_NOT_IMPLEMENTED;
- AO_NO_MEMORY;
- AO_SESSION_NOT_ACTIVE。

(7) APPLICATIONELEMENT_GETATTRIBUTEBYBASENAME

目的：获取从具有给定名称的基属性派生的应用元素的应用程序属性。属性的基名称不区分大小写（基模型本身不区分大小写；id, id, id, …被视为相同的项），不能包含通配符。

注意：基础模型是不区分大小写的，例如 id、id 和 id 都是相同的基础属性。参数：baName (Type = Name)

基本属性名称。Java 调用序列：ApplicationAttribute applAttr = applElem.getAttributeByBaseName (baName) ;

返回：返回名称：applAttr

返回类型：ApplicationAttribute

应用程序属性。可能的异常详见 13.3.18 节如下函数：

——AO_BAD_PARAMETER;
 ——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

(8) APPLICATIONELEMENT_GETATTRIBUTEBYNAME

目的：获取具有给定名称的应用元素的应用程序属性。名称区分大小写，不能包含通配符。

注1：应用程序模型（与基本模型相反）区分大小写；例如 ID、ID 和 ID 是三个不同的项。

注2：应用程序模型是区分大小写的，例如 id 和 id 是不同的应用程序属性，不要使用这种误导性的属性名称。

参数：aaName (Type = Name)

应用程序属性名称。Java 调用序列：ApplicationAttribute applAttr = applElem.getAttributeByName (aaName);

返回：返回名称：applAttr

返回类型：ApplicationAttribute

The application attribute.

可能的异常详见 13.3.18 节如下函数：

——AO_BAD_PARAMETER;
 ——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

(9) APPLICATIONELEMENT_GETATTRIBUTES

目的：获取应用元素的应用程序属性列表。不会返回引用属性。

参数: aaPattern (Type = Pattern)

应用程序属性名称的名称或搜索模式。Java 调用序列: ApplicationAttribute[] applAttrs =
applElem.getAttributes (aaPattern);

返回: 返回名称: applAttrs

返回类型: ApplicationAttributeSequence

The application attributes.

可能的异常详见 13.3.18 节如下函数:

——AO_BAD_PARAMETER;
——AO_CONNECTION_LOST;
——AO_IMPLEMENTATION_PROBLEM;
——AO_NOT_IMPLEMENTED;
——AO_NO_MEMORY;
——AO_SESSION_NOT_ACTIVE。

(10) APPLICATIONELEMENT_GETBASEELEMENT

目的: 获取应用元素的基元素。

参数: None.

Java 调用序列: BaseElement baseElem = applElem.getBaseElement ();

返回: 返回名称: baseElem

返回类型: BaseElement

The base element.

可能的异常详见 13.3.18 节如下函数:

——AO_CONNECTION_LOST;
——AO_IMPLEMENTATION_PROBLEM;
——AO_NOT_IMPLEMENTED;
——AO_NO_MEMORY;
——AO_SESSION_NOT_ACTIVE。

(11) APPLICATIONELEMENT_GETID

目的: 获取应用元素的 ID。

参数: None.

Java 调用序列: T_LONGLONG aeld = applElem.getId ();

返回: 返回名称: aeld

返回类型: T_LONGLONG

应用元素的 ID。可能的异常详见 13.3.18 节如下函数:

——AO_CONNECTION_LOST;
——AO_IMPLEMENTATION_PROBLEM;

——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

(12) APPLICATIONELEMENT_GETINITIALRIGHTRELATIONS

目的：获取用于检索实例以创建新创建实例的初始权限的所有关系。如果存在多个应用程序关系，则每个相关实例的初始权限将“或”添加到初始权限列表中。

参数：None.

Java 调用序列：ApplicationRelation[] applRels = applElem.getInitialRightRelations () ;

返回：返回名称：applRels

返回类型：ApplicationRelationSequence

具有将用于创建新创建实例的初始权限的应用程序关系的序列。可能的异常详见 13.3.18 节如下函数：

——AO_BAD_PARAMETER;
 ——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

(13) APPLICATIONELEMENT_GETINITIALRIGHTS

目的：检索给定对象的初始权限的访问控制列表信息。

参数：None.

Java 调用序列：InitialRight[] initialRights = applElem.getInitialRights () ;

返回：返回名称：initialRights

返回类型：InitialRightSequence

访问控制列表项具有给定应用元素的初始权限。可能的异常详见 13.3.18 节如下函数：

——AO_BAD_PARAMETER;
 ——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

(14) APPLICATIONELEMENT_GETINSTANCEBYID

目的：获取由给定 ID 指定的实例。如果该实例的 ID 不唯一，则引发异常。

参数：ield (Type = T_LONGLONG)

实例 ID。Java 调用序列：InstanceElement instElem = applElem.getInstanceById (ield) ;

返回：返回名称：instElem

返回类型：InstanceElement

The instance.

可能的异常详见 13.3.18 节如下函数：

——AO_BAD_PARAMETER;
——AO_CONNECTION_LOST;
——AO_IMPLEMENTATION_PROBLEM;
——AO_NOT_IMPLEMENTED;
——AO_NO_MEMORY;
——AO_SESSION_NOT_ACTIVE。

(15) APPLICATIONELEMENT_GETINSTANCEBYNAME

目的：获取由给定名称指定的实例。如果实例的名称不唯一，则引发异常。

对于具有唯一名称的实例，这是一个方便的方法。如果实例的名称重复，请改用 `getInstances` 方法，并将请求的名称指定为模式参数。参数：ieName (Type = Name)

实例名称。Java 调用序列：InstanceElement instElem = applElem.getInstanceByName (ieName);

返回：返回名称：instElem

返回类型：InstanceElement

The instance.

可能的异常详见 13.3.18 节如下函数：

——AO_BAD_PARAMETER;
——AO_CONNECTION_LOST;
——AO_DUPLICATE_NAME;
——AO_IMPLEMENTATION_PROBLEM;
——AO_NOT_IMPLEMENTED;
——AO_NO_MEMORY;
——AO_SESSION_NOT_ACTIVE。

(16) APPLICATIONELEMENT_GETINSTANCES

目的：获取名称与模式匹配的实例。模式区分大小写，可以包含通配符。

参数：iePattern (Type = Pattern)

实例名称的名称或搜索模式。Java 调用序列：InstanceElementIterator ieliterator = applElem.getInstances (iePattern);

返回：返回名称：ieliterator

返回类型：InstanceElementIterator

实例。可能的异常详见 13.3.18 节如下函数：

——AO_BAD_PARAMETER;
 ——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

(17) APPLICATIONELEMENT_GETNAME

目的：获取应用元素的名称。

参数：None.

Java 调用序列：Name aeName = applElem.getName () ;

返回：返回名称：aeName

返回类型：Name

应用元素的名称。可能的异常详见 13.3.18 节如下函数：

——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

(18) APPLICATIONELEMENT_GETRELATEDELEMENTSBYRELATIONSHIP

目的：通过指定的关系连接相关的应用元素。

参数：aeRelationship (Type = Relationship)

请 求 的 关 系 。 Java 调 用 序 列 ： ApplicationElement[] applElems
 =applElem.getRelatedElementsByRelationship (aeRelationship) ;

返回：返回名称：applElems

返回类型：ApplicationElementSequence

相关的应用元素。可能的异常详见 13.3.18 节如下函数：

——AO_BAD_PARAMETER;
 ——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_INVALID_RELATIONSHIP;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE;

(19) APPLICATIONELEMENT_GETRELATIONSBYBASENAME

目的：返回从具有给定关系名称的基关系派生的应用程序关系。

请注意，最多只能从基关系派生一个应用程序关系。但是，有一些定义良好的异常，其中此方法可

能返回多个应用程序关系。参数：baseRelName (Type = Name)

基关系的名称。Java 调用序列：ApplicationRelation[] applRels =

applElem.getRelationsByBaseName (baseRelName) ;

返回：返回名称：applRels

返回类型：ApplicationRelationSequence

应用程序关系序列；如果找不到关系，则返回空序列。可能的异常详见 13.3.18 节如下函数：

——AO_CONNECTION_LOST;

——AO_IMPLEMENTATION_PROBLEM;

——AO_NOT_IMPLEMENTED;

——AO_NO_MEMORY;

——AO_SESSION_NOT_ACTIVE。

(20) APPLICATIONELEMENT_GETRELATIONSBYTYPE

目的：获取从该应用元素连接的所请求类型的应用程序关系。不返回反向关系。参数：

aeRelationType (Type = RelationType)

请求的关系类型。Java 调用序列：ApplicationRelation[] applRels = applElem.getRelationsByType

(aeRelationType) ;

返回：返回名称：applRels

返回类型：ApplicationRelationSequence

The application relations.

可能的异常详见 13.3.18 节如下函数：

——AO_BAD_PARAMETER;

——AO_CONNECTION_LOST;

——AO_IMPLEMENTATION_PROBLEM;

——AO_INVALID_RELATION_TYPE;

——AO_NOT_IMPLEMENTED;

——AO_NO_MEMORY;

——AO_SESSION_NOT_ACTIVE。

(21) APPLICATIONELEMENT_GETRIGHTS

目的：检索给定对象的访问控制列表信息。

参数：None.

Java 调用序列：ACL[] aclEntries = applElem.getRights () ;

返回：返回名称：aclEntries

返回类型：ACLSequence

给定应用元素的访问控制列表项。可能的异常详见 13.3.18 节如下函数：

——AO_BAD_PARAMETER;
 ——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

(22) APPLICATIONELEMENT_GETSECURITYLEVEL

目的：获取应用元素的安全级别。安全级别指示是否对应用元素和实例进行安全检查，或者只对应应用程序属性、实例或完全不进行安全检查。

参数：None.

Java 调用序列：SecurityLevel secLevel = applElem.getSecurityLevel () ;

返回：返回名称：secLevel

返回类型：T_LONG

当前安全级别。安全级别常量在接口 security level 中定义。接口定义语言 IDL 不允许设置枚举值，因此必须在接口中完成常量定义。可能的异常详见 13.3.18 节如下函数：

——AO_BAD_PARAMETER;
 ——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

(23) APPLICATIONELEMENT_LISTALLRELATEDELEMENTS

目的：获取所有相关应用元素的名称。

参数：None.

Java 调用序列：Name[] applElemNames = applElem.listAllRelatedElements () ;

返回：返回名称：applElemNames

返回类型：NameSequence

相关应用元素的名称。可能的异常详见 13.3.18 节如下函数：

——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

(24) APPLICATIONELEMENT_LISTATTRIBUTES

目的：获取应用元素的应用程序属性名称。结果列表中没有返回包含对其他应用元素的引用的属性名。

参数：aaPattern (Type = Pattern)

应用程序属性名称的名称或搜索模式。Java 调用序列：Name[] applAttrNames = applElem.listAttributes (aaPattern) ;

返回：返回名称：applAttrNames

返回类型：NameSequence

应用程序属性的名称。可能的异常详见 13.3.18 节如下函数：

——AO_BAD_PARAMETER;
——AO_CONNECTION_LOST;
——AO_IMPLEMENTATION_PROBLEM;
——AO_NOT_IMPLEMENTED;
——AO_NO_MEMORY;
——AO_SESSION_NOT_ACTIVE。

(25) APPLICATIONELEMENT_LISTINSTANCES

目的：获取名称与模式匹配的实例的名称。模式区分大小写，可以包含通配符。

参数：aaPattern (Type = Pattern)

应用程序属性名称的名称或搜索模式。Java 调用序列：NameIterator ieNameIterator = applElem.listInstances (aaPattern) ;

返回：返回名称：ieNameIterator

返回类型：NameIterator

实例的名称。可能的异常详见 13.3.18 节如下函数：

——AO_BAD_PARAMETER;
——AO_CONNECTION_LOST;
——AO_IMPLEMENTATION_PROBLEM;
——AO_NOT_IMPLEMENTED;
——AO_NO_MEMORY;
——AO_SESSION_NOT_ACTIVE。

(26) APPLICATIONELEMENT_LISTRELATEDELEMENTSBYRELATIONSHIP

目的：获取通过指定关系连接的相关应用元素的名称。

参数：aeRelationship (Type = Relationship)

请求的关系。Java 调用序列：Name[] applElemNames = applElem.listRelatedElementsByRelationship (aeRelationship) ;

返回：返回名称：applElemNames

返回类型: **NameSequence**

相关应用元素的名称。可能的异常详见 13.3.18 节如下函数:

——**AO_BAD_PARAMETER**;
 ——**AO_CONNECTION_LOST**;
 ——**AO_IMPLEMENTATION_PROBLEM**;
 ——**AO_INVALID_RELATIONSHIP**;
 ——**AO_NOT_IMPLEMENTED**;
 ——**AO_NO_MEMORY**;
 ——**AO_SESSION_NOT_ACTIVE**。

(27) **APPLICATIONELEMENT_REMOVEATTRIBUTE**

目的: 从应用元素中移除应用程序属性。如果存在应用元素的实例, 则现有实例的属性将从应用程序更改为实例属性。

允许在事务外部修改应用元素, 但建议激活事务。参数: **applAttr** (Type = **ApplicationAttribute**)

要删除的应用程序属性。Java 调用序列: **applElem.removeAttribute (applAttr)** ;

返回: **None**.

可能的异常详见 13.3.18 节如下函数:

——**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**。

(28) **APPLICATIONELEMENT_REMOVEINSTANCE**

目的: 从应用元素中移除实例。

允许删除事务外部的实例, 但建议激活事务。提交事务时, 将从服务器中删除该实例。如果设置了递归标志, 则实例的所有子代也将被删除。仅当没有对此实例的引用 (关系) 时才允许删除实例。如果设置了递归标志, 则不允许引用其中一个子项, 并将导致异常。参数: **ield** (Type = **T_LONGLONG**)

实例 ID。

recursive (Type = **T_BOOLEAN**)

递归标志。Java 调用序列: **applElem.removeInstance (ield,recursive)** ;

返回: **None**.

可能的异常详见 13.3.18 节如下函数:

——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。

(29) APPLICATIONELEMENT_SETBASEELEMENT

目的：设置应用元素的基元素。

允许在事务外部修改应用元素，但建议激活事务。对当前基本元素的赋值将被覆盖。如果存在应用

元素的实例或对应用元素的引用，则会引发异常。参数：baseElem (Type = BaseElement)

基本元素。Java 调用序列：applElem.setBaseElement (baseElem) ;

返回：None。

可能的异常详见 13.3.18 节如下函数：

——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。

(30) APPLICATIONELEMENT_SETINITIALRIGHTRELATION

目的：为给定的应用元素设置，哪个关系将用于确定新创建实例的初始权限。

参数：applRel (Type = ApplicationRelation)

用于确定初始权利的应用关系。应用程序关系的关系范围必须为[1:1]，否则服务器找不到唯一的实例来检索初始权限。set (Type = T_BOOLEAN)

设置或删除初始权限的关系。如果此参数为真，则将设置关系，否则将删除关系。Java 调用序列：

applElem.setInitialRightRelation (applRel,set) ;

返回：None。

可能的异常详见 13.3.18 节如下函数：

——AO_BAD_PARAMETER;

——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE;
 ——AO_TRANSACTION_NOT_ACTIVE。

(31) APPLICATIONELEMENT_SETINITIALRIGHTS

目的：应为其设置初始权限的给定用户组。<rights>定义要设置或清除的权限。如果参数<set>设置为'set'，则设置<rights>中的权限，清除所有其他权限。如果参数<set>设置为'add'，则将<rights>中的权限添加到现有权限中。如果参数<set>设置为'remove'，则将从现有权限中删除<rights>中的权限。

参数：usergroup (Type = InstanceElement)

将修改其初始权限的用户组。rights (Type = T_LONG)

用户组的新初始权限。权限常量在接口 SecurityRights 中定义。接口定义语言 IDL 不允许设置枚举值，因此必须在接口中完成常量定义。refAid (Type = T_LONGLONG)

将使用初始权限的引用应用元素的 ID。如果没有设置 refaid，则独立于应用元素的每个新实例都将使用初始权限。set (Type = RightsSet)

如何处理新的初始权利。Java 调用序列：applElem.setInitialRights (usergroup, rights, refAid, set);
 返回：None。

可能的异常详见 13.3.18 节如下函数：

——AO_BAD_PARAMETER;
 ——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE;
 ——AO_TRANSACTION_NOT_ACTIVE。

(32) APPLICATIONELEMENT_SETNAME

目的：设置应用元素的名称。

允许在事务外部修改应用元素，但建议激活事务。应用元素的名称必须唯一。

应用元素的名称不能超过基础物理存储的最大名称长度。物理存储的当前规范将其限制为 30 个字符。参数：aeName (Type = Name)

应用元素名称。Java 调用序列：applElem.setName (aeName);

返回：None。

可能的异常详见 13.3.18 节如下函数：

——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。

(33) APPLICATIONELEMENT_SETRIGHTS

目的: 应为其设置权限的给定用户组。<rights>定义要设置或清除的权限。如果参数<set>设置为'set', 则设置<rights>中的权限, 清除所有其他权限。如果参数<set>设置为'add', 则将<rights>中的权限添加到现有权限中。如果参数<set>设置为'remove', 则将从现有权限中删除<rights>中的权限。

参数: usergroup (Type = InstanceElement)

将修改其权限的用户组。rights (Type = T_LONG)

用户组的新权限。权限常量在接口中定义

安全性。接口定义语言 IDL 不允许设置

因此, 必须在接口中进行常量定义的枚举。set (Type = RightsSet)

如何处理新的权利。Java 调用序列: applElem.setRights (usergroup, rights, set);

返回: None.

可能的异常详见 13.3.18 节如下函数:

——AO_BAD_PARAMETER;
 ——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE;
 ——AO_TRANSACTION_NOT_ACTIVE。

(34) APPLICATIONELEMENT_SETSECURITYLEVEL

目的: 设置给定应用元素的安全级别。如果添加了安全级别, 客户端将负责现有对象的访问控制列表条目。

参数: secLevel (Type = T_LONG)

新的安全级别。安全级别常量在接口 SecurityLevel 中定义。接口定义语言 IDL 不允许设置枚举值, 因此必须在接口中完成常量定义。set (Type = RightsSet)

如何处理新的安全级别。Java 调用序列: applElem.setSecurityLevel (secLevel, set);

返回: None.

可能的异常详见 13.3.18 节如下函数:

——AO_BAD_PARAMETER;
 ——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE;
 ——AO_TRANSACTION_NOT_ACTIVE。

13.2.6 APPLICATIONRELATION

(1) APPLICATIONRELATION_GETBASERELATION

目的: 获取应用程序关系的基关系。

参数: None.

Java 调用序列: `BaseRelation baseRel = applRel.getBaseRelation ()`;

返回: 返回名称: `baseRel`

返回类型: `BaseRelation`

应用关系的基础关系。如果应用程序关系没有基关系, 则返回 “null”。可能的异常详见 13.3.18 节如下函数:

——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

(2) APPLICATIONRELATION_GETELEM1

目的: 获取应用关系的第一个应用元素。

参数: None.

Java 调用序列: `ApplicationElement applElem = applRel.getElem1 ()`;

返回: 返回名称: `applElem`

返回类型: `ApplicationElement`

应用关系的第一个应用元素。可能的异常详见 13.3.18 节如下函数:

——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;

——AO_SESSION_NOT_ACTIVE。

(3) APPLICATIONRELATION_GETELEM2

目的：获取应用关系的第二个应用元素。

参数：None。

Java 调用序列：ApplicationElement applElem = applRel.getElem2 () ；

返回：返回名称：applElem

返回类型：ApplicationElement

应用关系的第二个应用元素。可能的异常详见 13.3.18 节如下函数：

——AO_CONNECTION_LOST；

——AO_IMPLEMENTATION_PROBLEM；

——AO_NOT_IMPLEMENTED；

——AO_NO_MEMORY；

——AO_SESSION_NOT_ACTIVE。

(4) APPLICATIONRELATION_GETINVERSERELATIONNAME

目的：获取应用程序关系的逆名称。应用程序关系的逆名称是从其他应用元素看到的关系的名称。

参数：None。

Java 调用序列：Name arInvName = applRel.getInverseRelationName () ；

返回：返回名称：arInvName

返回类型：Name

应用程序关系的逆名称。可能的异常详见 13.3.18 节如下函数：

——AO_CONNECTION_LOST；

——AO_IMPLEMENTATION_PROBLEM；

——AO_NOT_IMPLEMENTED；

——AO_NO_MEMORY；

——AO_SESSION_NOT_ACTIVE。

(5) APPLICATIONRELATION_GETINVERSERELATIONRANGE

目的：得到应用关系的逆关系范围。

参数：None。

Java 调用序列：RelationRange arRange = applRel.getInverseRelationRange () ；

返回：返回名称：arRange

返回类型：RelationRange

应用关系的反关系范围。可能的异常详见 13.3.18 节如下函数：

——AO_CONNECTION_LOST；

——AO_IMPLEMENTATION_PROBLEM；

——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

(6) APPLICATIONRELATION_GETINVERSERELATIONSHIP

目的：得到应用关系的逆关系。

参数：None.

Java 调用序列：Relationship relationship = applRel.getInverseRelationship () ;

返回：返回名称：relationship

返回类型：Relationship

应用关系的逆关系。可能的异常详见 13.3.18 节如下函数：

——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

(7) APPLICATIONRELATION_GETRELATIONNAME

目的：获取应用程序关系的名称。

参数：None.

Java 调用序列：Name arName = applRel.getRelationName () ;

返回：返回名称：arName

返回类型：Name

应用程序关系的名称。可能的异常详见 13.3.18 节如下函数：

——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

(8) APPLICATIONRELATION_GETRELATIONRANGE

目的：获取应用关系的关系范围。参数：None.

Java 调用序列：RelationRange arRange = applRel.getRelationRange () ;

返回：返回名称：arRange

返回类型：RelationRange

应用关系的关系范围。可能的异常详见 13.3.18 节如下函数：

——AO_CONNECTION_LOST;

——AO_IMPLEMENTATION_PROBLEM;
——AO_NOT_IMPLEMENTED;
——AO_NO_MEMORY;
——AO_SESSION_NOT_ACTIVE。

(9) APPLICATIONRELATION_GETRELATIONSHIP

目的：获取应用程序关系的关系。

参数：None.

Java 调用序列：Relationship relationship = applRel.getRelationship () ;

返回：返回名称：relationship

返回类型：Relationship

应用程序关系的关系。可能的异常详见 13.3.18 节如下函数：

——AO_CONNECTION_LOST;
——AO_IMPLEMENTATION_PROBLEM;
——AO_NOT_IMPLEMENTED;
——AO_NO_MEMORY;
——AO_SESSION_NOT_ACTIVE。

(10) APPLICATIONRELATION_GETRELATIONTYPE

目的：获取应用程序关系的关系类型。

参数：None.

Java 调用序列：RelationType arType = applRel.getRelationType () ;

返回：返回名称：arType

返回类型：RelationType

应用程序关系的关系类型。可能的异常详见 13.3.18 节如下函数：

——AO_CONNECTION_LOST;
——AO_IMPLEMENTATION_PROBLEM;
——AO_NOT_IMPLEMENTED;
——AO_NO_MEMORY;
——AO_SESSION_NOT_ACTIVE。

(11) APPLICATIONRELATION_SETBASERELATION

目的：

设置应用程序关系的基础关系。

允许在事务外修改应用关系，但建议激活事务。从基础关系复制关系类型和关系范围。以前的值会

丢失。参数：baseRel (Type = BaseRelation)

基本关系。Java 调用序列：applRel.setBaseRelation (baseRel) ;

返回: None.

可能的异常详见 13.3.18 节如下函数:

——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。

(12) APPLICATIONRELATION_SETELEM1

目的: 设置应用关系的第一个应用元素。

允许在事务外修改应用关系, 但建议激活事务。参数: applElem (Type = ApplicationElement)

应用元素。Java 调用序列: applRel.setElem1 (applElem);

返回: None.

可能的异常详见 13.3.18 节如下函数:

——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。

(13) APPLICATIONRELATION_SETELEM2

目的: 设置应用关系的第二个应用元素。

允许在事务外修改应用关系, 但建议激活事务。参数: applElem (Type = ApplicationElement)

应用元素。Java 调用序列: applRel.setElem2 (applElem);

返回: None.

可能的异常详见 13.3.18 节如下函数:

——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。

(14) APPLICATIONRELATION_SETINVERSERELATIONNAME

目的：设置应用程序关系的逆名称。

允许在事务外修改应用关系，但建议激活事务。参数：arInvName (Type = Name)

反向应用程序关系名称。Java 调用序列：applRel.setInverseRelationName (arInvName)；

返回：None。

可能的异常详见 13.3.18 节如下函数：

——AO_BAD_PARAMETER;
——AO_CONNECTION_LOST;
——AO_IMPLEMENTATION_PROBLEM;
——AO_NOT_IMPLEMENTED;
——AO_NO_MEMORY;
——AO_SESSION_NOT_ACTIVE;
——AO_TRANSACTION_NOT_ACTIVE。

(15) APPLICATIONRELATION_SETINVERSERELATIONRANGE

目的：设置应用关系的反关系范围。

允许在事务外修改应用关系，但建议激活事务。

只有在未定义基础关系的情况下，才允许设置关系类型。参数：arRelationRange (Type = RelationRange)

反关系范围。Java 调用序列：applRel.setInverseRelationRange (arRelationRange)；

返回：None。

可能的异常详见 13.3.18 节如下函数：

——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。

(16) APPLICATIONRELATION_SETRELATIONNAME

目的：设置应用程序关系的名称。

允许在事务外修改应用关系，但建议激活事务。

应用程序关系的名称不能超过基础物理存储的最大名称长度。物理存储的当前规范将其限制为 30 个字符。参数：arName (Type = Name)

应用程序关系名称。Java 调用序列：applRel.setRelationName (arName) ;

返回：None.

可能的异常详见 13.3.18 节如下函数：

——AO_BAD_PARAMETER;
 ——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE.

(17) APPLICATIONRELATION_SETRELATIONRANGE

目的：设置应用程序关系的关系范围。

允许在事务外修改应用关系，但建议激活事务。只有在未定义基础关系的情况下，才允许设置关系类型。参数：arRelationRange (Type = RelationRange)

The relation range.

Java 调用序列：applRel.setRelationRange (arRelationRange) ;

返回：None.

可能的异常详见 13.3.18 节如下函数：

——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.

(18) APPLICATIONRELATION_SETRELATIONTYPE

目的：设置应用程序关系的关系类型。

允许在事务外修改应用关系，但建议激活事务。当设置关系类型时，将自动设置关系。只有在未定义基础关系的情况下，才允许设置关系类型。只有在没有给定基关系时才允许修改

关系类型，否则将引发异常 `ao_has_base_relation`。不允许设置关系类型 `FATHER_CHILD`，此关系类型为基础模型保留。可以设置的两种关系类型是 `INFO` 和 `INHERITANCE`。关系类型设置后自动设置，关系范围的最大值确定关系元素之间的关系，见表 206 所述。

表206 从关系类型和范围生成的关系

关系类型	最大关系范围	最大逆向关系范围	关系
INFO	1	1	exception AO_INVALID_RELATION_TYPE
INFO	1	MANY	INFO_TO
INFO	MANY	1	INFO_FROM
INFO	MANY	MANY	INFO_REL
INHERITANCE	1	1	Exception AO_INVALID_RELATION_TYPE
INHERITANCE	1	MANY	SUPERTYPE
INHERITANCE	MANY	1	SUBTYPE
INHERITANCE	MANY	MANY	exception AO_INVALID_RELATION_TYPE

参数： `arRelationType` (Type = `RelationType`)

关系类型。

Java 调用序列： `applRel.setRelationType (arRelationType)` ；

返回： `None`。

可能的异常详见 13.3.18 节如下函数：

- `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`。

13. 2. 7 APPLICATIONSTRUCTURE

`ApplicationStructure` 接口允许访问服务器当前使用的 ASAM ODS 应用模型(需要注意结构和模型之间表述方面的差异)，`BaseStructure` 与 ASAM ODS 基础模型的关系也类似。

(1) APPLICATIONSTRUCTURE_CHECK

目的：检查应用模型是否符合 ASAM ODS。发现的第一个错误由异常报告。执行以下检查：每个

应用元素必须从有效的基元素派生。应用程序属性可以从基属性派生。一个应用元素内不允许从同一基属性派生多个应用程序属性。允许应用程序属性不是从任何基属性派生的。所有应用元素必须至少具有强制属性。每个应用元素必须由唯一的 ASAM 路径标识。不允许“浮动”应用元素。必须存在基本模型所需的所有关系。参数：None.

Java 调用序列：applStruct.check（）；

返回：None.

可能的异常详见 13.3.18 节如下函数：

——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。

(2) APPLICATIONSTRUCTURE_CREATEELEMENT

目的：在应用程序模型中创建新的应用元素。

允许在事务外部修改应用程序模型，但建议激活事务。新应用元素是否为顶级元素的信息取自指定的基元素。应用元素的 ID 是自动设置的。将自动创建必需的基本属性。选项属性必须由调用程序创建。应用程序属性接口方法可用于修改属性。参数：baseElem（Type = BaseElement）

从中派生应用元素的基本元素。Java 调用序列：ApplicationElement applElem = applStruct.createElement（baseElem）；

返回：返回名称：applElem

返回类型：ApplicationElement

示例：新的应用元素。Java 语言

```
import org.asam.ods.AoSession;
import org.asam.ods.ApplicationStructure;
import org.asam.ods.ApplicationElement;
import org.asam.ods.BaseStructure;
import org.asam.ods.BaseElement;
// Create a new session with aoFactory.
AoSession session;
```

```

    session = aoFactory.newSession("");
    // Session successfully created ?
    if (session != null) {
        // Get application model.
        ApplicationStructure as =
            session.getApplicationStructure();
        // Get base model.
        BaseStructure bs = session.getBaseStructure();
        // Get a base element by type.
        BaseElement be = bs.getElementByType("AoSubmatrix");
        // Start the transaction
        session.startTransaction();
        // Create a new application element.
        ApplicationElement ae = as.createElement(be);
        ...
        // Commit the transaction
        session.commitTransaction();
        // Close the session.
        session.close();
    }

```

可能的异常详见 13.3.18 节如下函数：

- AO_BAD_PARAMETER;
- AO_CONNECTION_LOST;
- AO_IMPLEMENTATION_PROBLEM;
- AO_NOT_IMPLEMENTED;
- AO_NO_MEMORY;
- AO_SESSION_NOT_ACTIVE;
- AO_TRANSACTION_NOT_ACTIVE。

(3) APPLICATIONSTRUCTURE_CREATEENUMERATIONDEFINITION

目的：创建新的枚举定义。此方法修改应用程序模型，仅允许超级用户使用。

参数：enumName (Type = T_STRING)

枚举的名称 Java 调用序列：EnumerationDefinition newEnum;

```
newEnum = applStruct.createEnumerationDefinition (enumName);
```

返回：返回名称：newEnum

返回类型：EnumerationDefinition

新创建的枚举可能的异常详见 13.3.18 节如下函数：

- 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) APPLICATIONSTRUCTURE_CREATEINSTANCERELATIONS

目的：创建实例列表之间的关系。两个列表中的实例数必须相同。每个列表中实例的应用元素必须相同。应用元素必须与应用程序关系的应用元素匹配。实例列表中的索引定义相关实例。

参数：applRel (Type = ApplicationRelation)

应用关系。*elemList1* (Type = InstanceElementSequence)

包含将为其创建关系的一个应用元素的实例的列表。*elemList2* (Type = InstanceElementSequence)

包含相关实例的列表。Java 调用序列：applStruct.createInstanceRelations (applRel, elemList1, elemList2) ;

返回：none

可能的异常详见 13.3.18 节如下函数：

- 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。

(5) APPLICATIONSTRUCTURE_CREATERELATION

目的：创建新关系。

允许在事务外部修改应用程序模型，但建议激活事务。关系是应用程序模型的一部分。

可以使用应用程序关系接口方法修改关系。新应用程序关系的默认属性为：BaseRelation
NULL

Element1	NULL
Element2	NULL
Range	-2, -2

Name	NULL
Type	INFO

如果在指定关系名称之前设置了元素 1 或元素 2，则应用程序关系的名称将设置为“autogen”。参数：None.

Java 调用序列：ApplicationRelation applRel = applStruct.createRelation () ;

返回：返回名称：applRel

返回类型：ApplicationRelation

新的应用程序关系。示例：Java 语言

```
import org.asam.ods.AoSession;
import org.asam.ods.ApplicationStructure;
import org.asam.ods.ApplicationRelation;
// Create a new session with aoFactory.
AoSession session;
session = aoFactory.newSession ("" ) ;
// Session successfully created ?
if (session != null) {
    session.StartTransaction ( ) ;
    // Get application model.
    ApplicationStructure as =
    session.getApplicationStructure ( ) ;
    // Create a new application relation.
    ApplicationRelation ar = as.createRelation ( ) ;
    ...
    session.commitTransaction ( ) ;
    // Close the session.
    session.close ( ) ;
}
```

可能的异常详见 13.3.18 节如下函数：

- AO_CONNECTION_LOST;
- AO_IMPLEMENTATION_PROBLEM;
- AO_NOT_IMPLEMENTED;
- AO_NO_MEMORY;
- AO_SESSION_NOT_ACTIVE;
- AO_TRANSACTION_NOT_ACTIVE。

(6) APPLICATIONSTRUCTURE_GETELEMENTBYID

目的：获取具有请求 ID 的应用元素。

参数: aeld (Type = T_LONGLONG)

请求的应用元素的 ID。Java 调用序列: `ApplicationElement applElem = applStruct.getElementById (aeld);`

返回: 返回名称: `applElem`

返回类型: `ApplicationElement`

请求的应用元素。示例: Java 语言

```
import org.asam.ods.AoSession;
import org.asam.ods.ApplicationStructure;
import org.asam.ods.ApplicationElement;
// Create a new session with aoFactory.
AoSession session;
session = aoFactory.newSession ("");
// Session successfully created ?
if (session != null) {
// Get application model.
ApplicationStructure as =
session.getApplicationStructure ();
// Get an application element by id.
ApplicationElement ae = as.getElementById (69);
...
// Close the session.
session.close ();
}
```

可能的异常详见 13.3.18 节如下函数:

- AO_BAD_PARAMETER;
- AO_CONNECTION_LOST;
- AO_IMPLEMENTATION_PROBLEM;
- AO_NOT_IMPLEMENTED;
- AO_NO_MEMORY;
- AO_SESSION_NOT_ACTIVE。

(7) APPLICATIONSTRUCTURE_GETELEMENTBYNAME

目的: 获取具有请求名称的应用元素。

参数: aeName (Type = Name)

请求的应用元素的名称。Java 调用序列: `ApplicationElement applElem = applStruct.getElementByName (aeName);`

返回: 返回名称: `applElem`

返回类型: **ApplicationElement**

请求的应用元素。示例: **Java** 语言

```
import org.asam.ods.AoSession;
import org.asam.ods.ApplicationStructure;
import org.asam.ods.ApplicationElement;
// Create a new session with aoFactory.
AoSession session;

session = aoFactory.newSession("");
// Session successfully created ?
if (session != null) {
// Get application model.
ApplicationStructure as =
session.getApplicationStructure();
// Get an application element by name.
ApplicationElement ae = as.getElementByName("myEngine");
...
// Close the session.
session.close();
}
```

可能的异常详见 13.3.18 节如下函数:

- AO_BAD_PARAMETER;
- AO_CONNECTION_LOST;
- AO_IMPLEMENTATION_PROBLEM;
- AO_NOT_IMPLEMENTED;
- AO_NO_MEMORY;
- AO_SESSION_NOT_ACTIVE。

(8) APPLICATIONSTRUCTURE_GETELEMENTS

目的: 获取名称与模式匹配的应用元素。模式区分大小写, 可以包含通配符。参数: **aePattern** (Type = Pattern)

请求的应用元素的名称或搜索模式。Java 调用序列: **ApplicationElement[] applElems = applStruct.getElements (aePattern);**

返回: 返回名称: **applElems**

返回类型: **ApplicationElementSequence**

请求的应用元素。示例: **Java** 语言

```
import org.asam.ods.AoSession;
import org.asam.ods.ApplicationStructure;
```

```

import org.asam.ods.ApplicationElement;
// Create a new session with aoFactory.
AoSession session;
session = aoFactory.newSession("");
// Session successfully created ?
if (session != null) {
// Get application model.
ApplicationStructure as =
session.getApplicationStructure();
// Get a list of application elements.
ApplicationElementSeq aeSeq[] = as.getElements("");
...
// Close the session.
session.close();
}

```

可能的异常详见 13.3.18 节如下函数:

- AO_BAD_PARAMETER;
- AO_CONNECTION_LOST;
- AO_IMPLEMENTATION_PROBLEM;
- AO_NOT_IMPLEMENTED;
- AO_NO_MEMORY;
- AO_SESSION_NOT_ACTIVE。

(9) APPLICATIONSTRUCTURE_GETELEMENTSBYBASETYPE

目的: 获取从指定的基元素派生的应用元素的名称。参数: aeType (Type = BaseType)

请求的基本元素类型。基元素类型可以是模式。Java 调用序列: ApplicationElement[] applElems =
 applStruct.getElementsByBaseType (aeType);

返回: 返回名称: applElems

返回类型: ApplicationElementSequence

请求的应用元素名称。示例: Java 语言

```

import org.asam.ods.AoSession;
import org.asam.ods.ApplicationStructure;
import org.asam.ods.ApplicationElement;
// Create a new session with aoFactory.
AoSession session;
session = aoFactory.newSession("");
// Session successfully created ?

```

```

    if (session != null) {
        // Get application model.
        ApplicationStructure as =
            session.getApplicationStructure ();
        // Get a list of application elements by type.
        ApplicationElementSeq aeSeq[] =
            as.getElementsByBaseType ("AoTest");
        ...
        // Close the session.
        session.close ();
    }

```

可能的异常详见 13.3.18 节如下函数：

——AO_BAD_PARAMETER;
 ——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_INVALID_BASETYPE;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

(10) APPLICATIONSTRUCTURE_GETENUMERATIONDEFINITION

目的：获取指定的枚举定义。

参数：enumName (Type = T_STRING)

请求的枚举的名称。Java 调用序列：EnumerationDefinition enumDef;

```
enumDef = applStruct.getEnumerationDefinition (enumName);
```

返回：返回名称：enumDef

返回类型：EnumerationDefinition

枚举定义。可能的异常详见 13.3.18 节如下函数：

——AO_BAD_PARAMETER;
 ——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

(11) APPLICATIONSTRUCTURE_GETINSTANCEBYASAMPATH

目的：获取由 ASAM 路径指定的实例。

参数: `asamPath` (Type = Name)

所请求实例的 ASAM 路径。Java 调用序列: `InstanceElement instElem = applStruct.getInstanceByAsamPath (asamPath);`

返回: 返回名称: `instElem`

返回类型: `InstanceElement`

The requested instance.

示例: Java 语言

```
InstanceElement ie;
Name asamPath;

ie = as.getInstanceByAsamPath (asamPath);
```

可能的异常详见 13.3.18 节如下函数:

- AO_BAD_PARAMETER;
- AO_CONNECTION_LOST;
- AO_IMPLEMENTATION_PROBLEM;
- AO_INVALID_ASAM_PATH;
- AO_NOT_IMPLEMENTED;
- AO_NO_MEMORY;
- AO_SESSION_NOT_ACTIVE。

(12) APPLICATIONSTRUCTURE_GETINSTANCESBYID

目的: 获取元素 ID 指定的实例。

参数: `ields` (Type = ElemIdSequence)

具有元素 ID 的序列。Java 调用序列: `InstanceElement[] instElems = applStruct.getInstancesById (ields);`

返回: 返回名称: `instElems`

返回类型: `InstanceElementSequence`

请求的实例序列。可能的异常详见 13.3.18 节如下函数:

- AO_BAD_PARAMETER;
- AO_CONNECTION_LOST;
- AO_IMPLEMENTATION_PROBLEM;
- AO_INVALID_ASAM_PATH;
- AO_NOT_IMPLEMENTED;
- AO_NO_MEMORY;
- AO_SESSION_NOT_ACTIVE。

(12) APPLICATIONSTRUCTURE_GETRELATIONS

目的: 返回两个应用元素之间的关系。

参数: applElem1 (Type = ApplicationElement)

第一个应用元素。

applElem2 (Type = ApplicationElement)

第二个应用元素。

Java 调用序列: ApplicationRelation[] applRels = applStruct.getRelations(applElem1,applElem2);

返回: 返回名称: applRels

返回类型: ApplicationRelationSequence

示例: 指定应用元素之间的关系。Java 语言

```
ApplicationRelationSeq arSeq[];
ApplicationElement ae1, ae2;
arSeq = as.getRelations (ae1,ae2) ;
```

可能的异常详见 13.3.18 节如下函数:

- AO_BAD_PARAMETER;
- AO_CONNECTION_LOST;
- AO_IMPLEMENTATION_PROBLEM;
- AO_NOT_IMPLEMENTED;
- AO_NO_MEMORY;
- AO_SESSION_NOT_ACTIVE。

(13) APPLICATIONSTRUCTURE_GETSESSION

目的: 获取在其中创建应用程序模型的当前客户端会话。参数: None.

Java 调用序列: AoSession session = applStruct.getSession ();

返回: 返回名称: session

返回类型: AoSession

The current client session.

可能的异常详见 13.3.18 节如下函数:

- AO_BAD_PARAMETER;
- AO_CONNECTION_LOST;
- AO_IMPLEMENTATION_PROBLEM;
- AO_NOT_IMPLEMENTED;
- AO_NO_MEMORY;
- AO_SESSION_NOT_ACTIVE。

(14) APPLICATIONSTRUCTURE_GETTOPLEVELELEMENTS

目的: 获取与基类型匹配的基元素所固有的顶级应用元素。如果给定的基类型不是顶级基元素, 则引发异常。顶级应用元素是没有父元素的应用元素。

参数: aeType (Type = BaseType)

请求的基类型。基元素类型可以是模式。Java 调用序列：`ApplicationElement[] applElems = applStruct.getTopLevelElements (aeType) ;`

返回：返回名称：`applElems`

返回类型：`ApplicationElementSequence`

顶级应用元素。示例：Java 语言

```
import org.asam.ods.AoSession;
import org.asam.ods.ApplicationStructure;
import org.asam.ods.ApplicationElement;
// Create a new session with aoFactory.
AoSession session;

session = aoFactory.newSession ("") ;
// Session successfully created ?
if (session != null) {
// Get application model.
ApplicationStructure as =
session.getApplicationStructure () ;
// Get a list of top level application elements.
ApplicationElementSeq aeSeq[] =
as.getTopLevelElements ("AoAny") ;
...
// Close the session.
session.close () ;
}
```

可能的异常详见 13.3.18 节如下函数：

- `AO_BAD_PARAMETER`;
- `AO_CONNECTION_LOST`;
- `AO_IMPLEMENTATION_PROBLEM`;
- `AO_INVALID_Basetype`;
- `AO_NOT_IMPLEMENTED`;
- `AO_NO_MEMORY`;
- `AO_SESSION_NOT_ACTIVE`。

(15) APPLICATIONSTRUCTURE_LISTELEMENTS

目的：获取与模式匹配的应用元素的名称。模式区分大小写，可以包含通配符。

参数：`aePattern (Type = Pattern)`

请求的基本元素的名称或搜索模式。Java 调用序列：`Name[] applElemNames = applStruct.listElements (aePattern) ;`

返回：返回名称：applElemNames

返回类型：NameSequence

应用元素的名称。示例：Java 语言

```
import org.asam.ods.AoSession;
import org.asam.ods.ApplicationStructure;
// Create a new session with aoFactory.
AoSession session;

session = aoFactory.newSession("");
// Session successfully created ?
if (session != null) {
// Get application model.
ApplicationStructure as =
session.getApplicationStructure();
// Get a list of application element names.
String aeNameList[] = as.listElements("*");
...
// Close the session.
session.close();
}
```

可能的异常详见 13.3.18 节如下函数：

- AO_BAD_PARAMETER;
- AO_CONNECTION_LOST;
- AO_IMPLEMENTATION_PROBLEM;
- AO_NOT_IMPLEMENTED;
- AO_NO_MEMORY;
- AO_SESSION_NOT_ACTIVE。

(16) APPLICATIONSTRUCTURE_LISTELEMENTSBYBASETYPE

目的：获取从给定基类型派生的应用元素的名称。

参数：aeType (Type = BaseType)

请求的基类型。基元素类型可以是模式。Java 调用序列：Name[] applElemNames =
applStruct.listElementsByBaseType (aeType);

返回：返回名称：applElemNames

返回类型：NameSequence

应用元素的名称。

示例：Java 语言

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

```

import org.asam.ods.ApplicationStructure;
// Create a new session with aoFactory.
AoSession session;
session = aoFactory.newSession("");
// Session successfully created ?
if (session != null) {
// Get application model.
ApplicationStructure as =
session.getApplicationStructure();
// Get a list of application element names.
String aeNameList[] = as.listElementsByBaseType("AoTest");
...
// Close the session.
session.close();
}

```

可能的异常详见 13.3.18 节如下函数:

- AO_BAD_PARAMETER;
- AO_CONNECTION_LOST;
- AO_IMPLEMENTATION_PROBLEM;
- AO_INVALID_BASETYPE;
- AO_NOT_IMPLEMENTED;
- AO_NO_MEMORY;
- AO_SESSION_NOT_ACTIVE。

(17) APPLICATIONSTRUCTURE_LISTENUMERATIONS

目的: 获取所有枚举名称的列表。

参数: None.

Java 调用序列: Name[] enumNames;

```
enumNames = applStruct.listEnumerations();
```

返回: 返回名称: enumNames

返回类型: NameSequence

列出所有枚举名称。可能的异常详见 13.3.18 节如下函数:

- AO_BAD_PARAMETER;
- AO_CONNECTION_LOST;
- AO_IMPLEMENTATION_PROBLEM;
- AO_NOT_IMPLEMENTED;
- AO_NO_MEMORY;

——AO_SESSION_NOT_ACTIVE。

(18) APPLICATIONSTRUCTURE_LISTTOPLEVELEMENTS

目的：获取从给定基类型派生的顶级应用元素的名称。如果给定的基类型不是顶级基元素，则引发异常。顶级应用元素是没有父元素的应用元素。

参数：aeType (Type = BaseType)

请求的基类型。Java 调用序列：Name[] applElemNames = applStruct.listTopLevelElements (aeType) ;

返回：返回名称：applElemNames

返回类型：NameSequence

示例：应用元素的名称。Java 语言

```
import org.asam.ods.AoSession;
import org.asam.ods.ApplicationStructure;
// Create a new session with aoFactory.
AoSession session;
session = aoFactory.newSession ("");
// Session successfully created ?
if (session != null) {
// Get application model.
ApplicationStructure as =
session.getApplicationStructure ();
// Get a list of top level application element names.
String aeNameList[] = as.listTopLevelElements ("*");
...
// Close the session.
session.close ();
}
```

可能的异常详见 13.3.18 节如下函数：

——AO_BAD_PARAMETER;
 ——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_INVALID_BASETYPE;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

(19) APPLICATIONSTRUCTURE_REMOVEELEMENT

目的：从应用程序模型中移除应用元素。

允许在事务外部修改应用程序模型，但建议激活事务。-只允许：-如果应用元素为空（没有实例）。

-与其他应用元素没有关系。参数：applElem（Type = ApplicationElement）

要删除的应用元素。Java 调用序列：applStruct.removeElement（applElem）；

返回：None.

示例：Java 语言

```
import org.asam.ods.AoSession;
import org.asam.ods.ApplicationStructure;
import org.asam.ods.ApplicationElement;
import org.asam.ods.BaseStructure;
import org.asam.ods.BaseElement;
// Create a new session with aoFactory.
AoSession session;
session = aoFactory.newSession("");
// Session successfully created ?
if (session != null) {
// Get application model.
ApplicationStructure as =
session.getApplicationStructure();
// Get base model.
BaseStructure bs = session.getBaseStructure();
// Get a base element by type.
BaseElement be = bs.getElementByType("AoSubmatrix");
// Create a new application element.
ApplicationElement ae = as.createElement(be);
...
// Remove an application element from application model.
as.removeElement(ae);
...
// Close the session.
session.close();
}
```

可能的异常详见 13.3.18 节如下函数：

——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。

(20) APPLICATIONSTRUCTURE_REMOVEENUMERATIONDEFINITION

目的：删除枚举定义。服务器检查某个属性是否仍在使用枚举。此方法修改应用程序模型，仅允许超级用户使用。

参数：enumName (Type = T_STRING)

要删除的枚举的名称。Java 调用序列：applStruct.removeEnumerationDefinition (enumName)；

返回：None。

可能的异常详见 13.3.18 节如下函数：

——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。

(21) APPLICATIONSTRUCTURE_REMOVERELATION

目的：此方法从应用程序模型中删除应用程序关系。

允许在事务外部修改应用程序模型，但建议激活事务。关系的元素仍然是应用程序模型的一部分。

如果存在关系的实例，则也会删除这些实例。参数：applRel (Type = ApplicationRelation)

要删除的应用程序关系。Java 调用序列：applStruct.removeRelation (applRel)；

返回：None。

示例：Java 语言

```
import org.asam.ods.AoSession;
import org.asam.ods.ApplicationStructure;
import org.asam.ods.ApplicationRelation;
// Create a new session with aoFactory.
AoSession session;
session = aoFactory.newSession("");
// Session successfully created ?
```

```

if (session != null) {
    // Get application model.
    ApplicationStructure as =
        session.getApplicationStructure ();
    // Create a new application relation.
    ApplicationRelation ar = as.createRelation ();
    ...
    // Remove an application relation from an application
    model.
    as.removeRelation (ar);
    ...
    // Close the session.
    session.close ();
}

```

可能的异常详见 13.3.18 节如下函数：

- 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。

13.2.8 BASEATTRIBUTE

(1) BASEATTRIBUTE_GETBASEELEMENT

目的：返回 **attribute** 所属的基本元素..

参数：None.

Java 调用序列：BaseElement baseElem = baseAttr.getBaseElement ();

返回：返回名称：baseElem

返回类型：BaseElement

属性的基元素。

可能的异常详见 13.3.18 节如下函数：

- AO_BAD_PARAMETER;
- AO_CONNECTION_LOST;
- AO_IMPLEMENTATION_PROBLEM;

——AO_NOT_IMPLEMENTED;
——AO_NO_MEMORY;
——AO_SESSION_NOT_ACTIVE。

(2) BASEATTRIBUTE_GETDATATYPE

目的：获取基属性的数据类型。

参数：None.

Java 调用序列：DataType baDataType = baseAttr.getDataType () ;

返回：返回名称：baDataType

返回类型：DataType

基于属性的数据类型。可能的异常详见 13.3.18 节如下函数：

——AO_CONNECTION_LOST;
——AO_IMPLEMENTATION_PROBLEM;
——AO_NOT_IMPLEMENTED;
——AO_NO_MEMORY;
——AO_SESSION_NOT_ACTIVE。

(3) BASEATTRIBUTE_GETENUMERATIONDEFINITION

目的：获取基属性枚举的定义。

参数：None.

Java 调用序列：EnumerationDefinition enumDef = baseAttr.getEnumerationDefinition () ;

返回：返回名称：enumDef

返回类型：EnumerationDefinition

基属性的枚举定义。可能的异常详见 13.3.18 节如下函数：

——AO_CONNECTION_LOST;
——AO_IMPLEMENTATION_PROBLEM;
——AO_NOT_IMPLEMENTED;
——AO_NO_MEMORY;
——AO_SESSION_NOT_ACTIVE;
——AO_INVALID_DATATYPE。

(4) BASEATTRIBUTE_GETNAME

目的：获取基属性的名称。

参数：None.

Java 调用序列：Name baName = baseAttr.getName () ;

返回：返回名称：baName

返回类型：Name

基属性的名称。可能的异常详见 13.3.18 节如下函数：

——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

(5) BASEATTRIBUTE_ISOBLIGATORY

目的：获取基属性的强制标志。

参数：None.

Java 调用序列：T_BOOLEAN balsObligatory = baseAttr.isObligatory () ;

返回：返回名称：balsObligatory

返回类型：T_BOOLEAN

The obligatory flag of the base attribute.

可能的异常详见 13.3.18 节如下函数：

——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

(6) BASEATTRIBUTE_ISUNIQUE

目的：获取基属性的唯一标志。

参数：None.

Java 调用序列：T_BOOLEAN balsUnique = baseAttr.isUnique () ;

返回：返回名称：balsUnique

返回类型：T_BOOLEAN

基属性的唯一标志。可能的异常详见 13.3.18 节如下函数：

——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

13.2.9 BASEELEMENT

(1) BASEELEMENT_GETALLRELATIONS

目的：获取基本元素的所有已知关系。

参数: None.

Java 调用序列: `BaseRelation[] baseRels = baseElem.getAllRelations ( ) ;`

返回: 返回名称: `baseRels`

返回类型: `BaseRelationSequence`

基本元素的所有已知关系。可能的异常详见 13.3.18 节如下函数:

- `AO_CONNECTION_LOST`;
- `AO_IMPLEMENTATION_PROBLEM`;
- `AO_NOT_IMPLEMENTED`;
- `AO_NO_MEMORY`;
- `AO_SESSION_NOT_ACTIVE`。

(2) BASEELEMENT_GETATTRIBUTES

目的: 获取基元素的属性。

参数: `baPattern (Type = Pattern)`

请求的基本属性的名称或搜索模式。Java 调用序列: `BaseAttribute[] baseAttrs = baseElem.getAttributes (baPattern) ;`

返回: 返回名称: `baseAttrs`

返回类型: `BaseAttributeSequence`

基元素的请求属性。可能的异常详见 13.3.18 节如下函数:

- `AO_BAD_PARAMETER`;
- `AO_CONNECTION_LOST`;
- `AO_IMPLEMENTATION_PROBLEM`;
- `AO_NOT_IMPLEMENTED`;
- `AO_NO_MEMORY`;
- `AO_SESSION_NOT_ACTIVE`。

(3) BASEELEMENT_GETRELATEDELEMENTSBYRELATIONSHIP

目的: 获取由关系定义的基元素的相关元素。

参数: `brRelationship (Type = Relationship)`

请求的关系。

Java 调用序列: `BaseElement[] baseElems = baseElem.getRelatedElementsByRelationship (brRelationship) ;`

返回: 返回名称: `baseElems`

返回类型: `BaseElementSequence`

基元素的相关元素。可能的异常详见 13.3.18 节如下函数:

- `AO_BAD_PARAMETER`;
- `AO_CONNECTION_LOST`;

——AO_IMPLEMENTATION_PROBLEM;
 ——AO_INVALID_RELATIONSHIP;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

(4) BASEELEMENT_GETRELATIONSBYTYPE

目的：获取所请求关系类型的基元素关系。

参数：brRelationType (Type = RelationType)

The requested relation type.

Java 调用序列：BaseRelation[] baseRels = baseElem.getRelationsByType (brRelationType) ;

返回：返回名称：baseRels

返回类型：BaseRelationSequence

所请求类型的基元素关系。可能的异常详见 13.3.18 节如下函数：

——AO_BAD_PARAMETER;
 ——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_INVALID_RELATION_TYPE;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

(5) BASEELEMENT_GETTYPE

目的：获取基元素的类型。基本元素的类型与基本元素的名称相同。基本元素的类型是字符串。

参数：None.

Java 调用序列：BaseType beType = baseElem.getType () ;

返回：返回名称：beType

返回类型：BaseType

基本元素的类型。可能的异常详见 13.3.18 节如下函数：

——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

(6) BASEELEMENT_ISTOPLEVEL

目的：获取基元素是否为顶级元素。顶层元素是没有父元素的元素。

参数: None.

Java 调用序列: `T_BOOLEAN belsTopLevel = baseElem.isTopLevel ()`;

返回: 返回名称: `belsTopLevel`

返回类型: `T_BOOLEAN`

布尔值, 基元素是否为顶级元素。可能的异常详见 13.3.18 节如下函数:

——`AO_CONNECTION_LOST`;
——`AO_IMPLEMENTATION_PROBLEM`;
——`AO_NOT_IMPLEMENTED`;
——`AO_NO_MEMORY`;
——`AO_SESSION_NOT_ACTIVE`。

(7) BASEELEMENT_LISTATTRIBUTES

目的: 获取基本元素的属性名称。参数: `baPattern (Type = Pattern)`

请求的基本属性名称的名称或搜索模式。Java 调用序列: `Name[] baseElemNames = baseElem.listAttributes (baPattern)`;

返回: 返回名称: `baseElemNames`

返回类型: `NameSequence`

基元素的请求属性名。可能的异常详见 13.3.18 节如下函数:

——`AO_BAD_PARAMETER`;
——`AO_CONNECTION_LOST`;
——`AO_IMPLEMENTATION_PROBLEM`;
——`AO_NOT_IMPLEMENTED`;
——`AO_NO_MEMORY`;
——`AO_SESSION_NOT_ACTIVE`。

(8) BASEELEMENT_LISTRELATEDELEMENTSBYRELATIONSHIP

目的: 获取由关系定义的基本元素的相关元素名称。

参数: `brRelationship (Type = Relationship)`

请求的关系。Java 调用序列: `BaseType[] baseTypes = baseElem.listRelatedElementsByRelationship (brRelationship)`;

返回: 返回名称: `baseTypes`

返回类型: `BaseTypeSequence`

基本元素的相关元素名称。可能的异常详见 13.3.18 节如下函数:

——`AO_BAD_PARAMETER`;
——`AO_CONNECTION_LOST`;
——`AO_IMPLEMENTATION_PROBLEM`;
——`AO_INVALID_RELATIONSHIP`;

——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

13.2.10 BASERELATION

(1) BASERELATION_GETELEM1

目的：获取基关系的第一个基元素。

参数：None.

Java 调用序列：BaseElement baseElem = baseRel.getElem1 () ;

返回：返回名称：baseElem

返回类型：BaseElement

基关系的第一个基元素。可能的异常详见 13.3.18 节如下函数：

——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

(2) BASERELATION_GETELEM2

目的：获取基关系的第二个基元素。

参数：None.

Java 调用序列：BaseElement baseElem = baseRel.getElem2 () ;

返回：返回名称：baseElem

返回类型：BaseElement

基关系的第二个基元素。可能的异常详见 13.3.18 节如下函数：

——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

(3) BASERELATION_GETINVERSERELATIONNAME

目的：获取反基关系的关系名称。

参数：None.

Java 调用序列：Name invRelName = baseRel.getInverseRelationName () ;

返回：返回名称：invRelName

返回类型：Name

The relation name of the inverse base relation.

可能的异常详见 13.3.18 节如下函数：

——AO_BAD_PARAMETER;
——AO_CONNECTION_LOST;
——AO_IMPLEMENTATION_PROBLEM;
——AO_NOT_IMPLEMENTED;
——AO_NO_MEMORY;
——AO_SESSION_NOT_ACTIVE。

(4) BASERELATION_GETINVERSERELATIONRANGE

目的：得到基关系的逆关系范围。

参数：None.

Java 调用序列：RelationRange brRange = baseRel.getInverseRelationRange () ;

返回：返回名称：brRange

返回类型：RelationRange

The inverse relation range of the base relation.

可能的异常详见 13.3.18 节如下函数：

——AO_CONNECTION_LOST;
——AO_IMPLEMENTATION_PROBLEM;
——AO_NOT_IMPLEMENTED;
——AO_NO_MEMORY;
——AO_SESSION_NOT_ACTIVE。

(5) BASERELATION_GETINVERSERELATIONSHIP

目的：得到基关系的逆关系。

参数：None.

Java 调用序列：Relationship relationship = baseRel.getInverseRelationship () ;

返回：返回名称：relationship

返回类型：Relationship

基关系的逆关系。可能的异常详见 13.3.18 节如下函数：

——AO_CONNECTION_LOST;
——AO_IMPLEMENTATION_PROBLEM;
——AO_NOT_IMPLEMENTED;
——AO_NO_MEMORY;
——AO_SESSION_NOT_ACTIVE。

(6) BASERELATION_GETRELATIONNAME

目的：获取基关系的关系名称。

参数：None.

Java 调用序列：Name brName = baseRel.getRelationName () ;

返回：返回名称：brName

返回类型：Name

The relation name of the base relation.

可能的异常详见 13.3.18 节如下函数：

——AO_CONNECTION_LOST;

——AO_IMPLEMENTATION_PROBLEM;

——AO_NOT_IMPLEMENTED;

——AO_NO_MEMORY;

——AO_SESSION_NOT_ACTIVE。

(7) BASERELATION_GETRELATIONRANGE

目的：获取基础关系的关系范围。

参数：None.

Java 调用序列：RelationRange brRange = baseRel.getRelationRange () ;

返回：返回名称：brRange

返回类型：RelationRange

基础关系的关系范围。可能的异常详见 13.3.18 节如下函数：

——AO_CONNECTION_LOST;

——AO_IMPLEMENTATION_PROBLEM;

——AO_NOT_IMPLEMENTED;

——AO_NO_MEMORY;

——AO_SESSION_NOT_ACTIVE。

(8) BASERELATION_GETRELATIONSHIP

目的：获取基础关系的关系。

参数：None.

Java 调用序列：Relationship relationship = baseRel.getRelationship () ;

返回：返回名称：relationship

返回类型：Relationship

The relationship of the base relation.

可能的异常详见 13.3.18 节如下函数：

——AO_CONNECTION_LOST;

——AO_IMPLEMENTATION_PROBLEM;

——AO_NOT_IMPLEMENTED;

——AO_NO_MEMORY;
——AO_SESSION_NOT_ACTIVE。

(9) BASERELATION_GETRELATIONTYPE

目的：获取基关系的关系类型。参数：None.

Java 调用序列：RelationType brType = baseRel.getRelationType () ;

返回：返回名称：brType

返回类型：RelationType

The relation type of the base relation.

可能的异常详见 13.3.18 节如下函数：

——AO_CONNECTION_LOST;
——AO_IMPLEMENTATION_PROBLEM;
——AO_NOT_IMPLEMENTED;
——AO_NO_MEMORY;
——AO_SESSION_NOT_ACTIVE。

13.2.11 BASESTRUCTURE

BaseStructure 接口允许访问服务器使用的当前 ASAM ODS 基础模型（需要注意结构和模型之间表述方面的差异），ApplicationStructure 与 ASAM ODS 应用模型间的关系也类似。

(1) BASESTRUCTURE_GETELEMENTBYTYPE

目的：获取与请求的类型匹配的基元素。基元素的类型与基元素的名称相同。

参数：beType (Type = BaseType)

请求的基本元素的名称。Java 调用序列：BaseElement baseElem = baseStruct.getElementByType (beType) ;

返回：返回名称：baseElem

返回类型：BaseElement

请求的基本元素。示例：Java 语言

```
import org.asam.ods.AoSession;
import org.asam.ods.BaseStructure;
// Create a new session with aoFactory.
AoSession session;
session = aoFactory.newSession ("");
// Session successfully created ?
if (session != null) {
// Get base model.
BaseStructure bs = session.getBaseStructure ();
// Get a the base element of type AoMeasurement.
```



```

BaseElement be = bs.getElementByType ("AoMeasurement");
...
// Close the session.
session.close ();
}

```

可能的异常详见 13.3.18 节如下函数：

——AO_BAD_PARAMETER;
 ——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_INVALID_Basetype;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

(2) BASESTRUCTURE_GETELEMENTS

目的：获取与模式匹配的基本元素。模式区分大小写，可以包含通配符。

参数：bePattern (Type = Pattern)

请求的基本元素的名称或搜索模式。Java 调用序列：BaseElement[] baseElems =
 baseStruct.getElements (bePattern);

返回：返回名称：baseElems

返回类型：BaseElementSequence

The requested base elements.

示例：Java 语言

```

import org.asam.ods.AoSession;
import org.asam.ods.BaseStructure;
// Create a new session with aoFactory.
AoSession session;
session = aoFactory.newSession ("");
// Session successfully created ?
if (session != null) {
// Get base model.
BaseStructure bs = session.getBaseStructure ();
// Get a list of base elements.
BaseElement beList[] = bs.getElements ("*");
...
// Close the session.
session.close ();
}

```

}

可能的异常详见 13.3.18 节如下函数：

——AO_BAD_PARAMETER;
 ——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

(3) BASESTRUCTURE_GETRELATION

目的：得到两个基元素之间的基关系。基本关系是从第一个元素（参数 **elem1**）到第二个元素（参数 **elem2**）之间的关系，而不是给出反比关系。

参数：**elem1**（Type = BaseElement）

关系开始的基元素。**elem2**（Type = BaseElement）

关系指向的基本元素。

Java 调用序列：**BaseRelation baseRel = baseStruct.getRelation (elem1,elem2) ;**

返回：返回名称：**baseRel**

返回类型：**BaseRelation**

The base relation between the two base elements.

示例：Java 语言

```
import org.asam.ods.AoSession;
import org.asam.ods.BaseStructure;
// Create a new session with aoFactory.
AoSession session;
session = aoFactory.newSession("");
// Session successfully created ?
if (session != null) {
// Get base model.
BaseStructure bs = session.getBaseStructure();
// Get a the base elements of type AoMeasurement and
AoQuantity.
BaseElement be1 = bs.getElementByType("AoMeasurement");
BaseElement be2 = bs.getElementByType("AoQuantity");
// Get the relation between the elements.
BaseRelation br = bs.getRelation (be1,be2);
...
// Close the session.
```

```
session.close();
}
```

可能的异常详见 13.3.18 节如下函数：

——AO_BAD_PARAMETER;
 ——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

(4) BASESTRUCTURE_GETTOPLEVELEMENTS

目的：获取与模式匹配的顶级基本元素。模式区分大小写，可以包含通配符。顶层基本元素是没有父元素的基本元素。参数：bePattern (Type = Pattern)

请求的顶级基本元素的名称或搜索模式。Java 调用序列：BaseElement[] baseElems = baseStruct.getTopLevelElements (bePattern);

返回：返回名称：baseElems

返回类型：BaseElementSequence

示例：请求的顶级基本元素。Java 语言

```
import org.asam.ods.AoSession;
import org.asam.ods.BaseStructure;
// Create a new session with aoFactory.
AoSession session;
session = aoFactory.newSession("");
// Session successfully created ?
if (session != null) {
// Get base model.
BaseStructure bs = session.getBaseStructure();
// Get a list of top level base elements.
BaseElement beList[] = bs.getTopLevelElements("*");
...
// Close the session.
session.close();
}
```

可能的异常详见 13.3.18 节如下函数：

——AO_BAD_PARAMETER;
 ——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;

——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

(5) BASESTRUCTURE_GETVERSION

目的：获取基础模型的版本。基础模型的版本是 ASAM ODS 基础模型的版本（例如：“ASAM30”）。

参数：None。

Java 调用序列：T_STRING version = baseStruct.getVersion（）；

返回：返回名称：version

返回类型：T_STRING

The version of the ASAM ODS base model.

示例：Java 语言

```
import org.asam.ods.AoSession;
import org.asam.ods.BaseStructure;
// Create a new session with aoFactory.
AoSession session;
session = aoFactory.newSession("");
// Session successfully created ?
if (session != null) {
    // Get base model.
    BaseStructure bs = session.getBaseStructure();
    // Get current base model version.
    String bsVersion = bs.getVersion();
    ...
    // Close the session.
    session.close();
}
```

可能的异常详见 13.3.18 节如下函数：

——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

(6) BASESTRUCTURE_LISTELEMENTS

目的：获取与模式匹配的基本元素名称。模式区分大小写，可以包含通配符。

参数：bePattern（Type = Pattern）

请求的基本元素名称的名称或搜索模式。Java 调用序列：`BaseType[] baseTypes = baseStruct.listElements (bePattern) ;`

返回：返回名称：`baseTypes`

返回类型：`BaseTypeSequence`

The requested base element names.

示例：Java 语言

```
import org.asam.ods.AoSession;
import org.asam.ods.BaseStructure;
// Create a new session with aoFactory.
AoSession session;
session = aoFactory.newSession ("");
// Session successfully created ?
if (session != null) {
// Get base model.
BaseStructure bs = session.getBaseStructure ();
// Get a list of base element names.
String beNameList[] = bs.ListElements ("*");
...
// Close the session.
session.close ();
}
```

可能的异常详见 13.3.18 节如下函数：

- `AO_BAD_PARAMETER`;
- `AO_CONNECTION_LOST`;
- `AO_IMPLEMENTATION_PROBLEM`;
- `AO_NOT_IMPLEMENTED`;
- `AO_NO_MEMORY`;
- `AO_SESSION_NOT_ACTIVE`。

(7) BASESTRUCTURE_LISTTOPLEVELELEMENTS

目的：获取与模式匹配的顶级基本元素名称。模式区分大小写，可以包含通配符。顶层基本元素是没有父元素的基本元素。

参数：`bePattern` (`Type = Pattern`)

请求的顶级基本元素名称的名称或搜索模式。Java 调用序列：`BaseType[] baseTypes = baseStruct.listTopLevelElements (bePattern) ;`

返回：返回名称：`baseTypes`

返回类型：`BaseTypeSequence`

示例：请求的顶级基本元素名称。Java 语言

```
import org.asam.ods.AoSession;
import org.asam.ods.BaseStructure;
// Create a new session with aoFactory.
AoSession session;
session = aoFactory.newSession("");
// Session successfully created ?
if (session != null) {
// Get base model.
BaseStructure bs = session.getBaseStructure();
// Get a list of top level base element names.
String beNameList[] = bs.ListTopLevelElements("");
...
// Close the session.
session.close();
}
```

可能的异常详见 13.3.18 节如下函数：

- AO_BAD_PARAMETER;
- AO_CONNECTION_LOST;
- AO_IMPLEMENTATION_PROBLEM;
- AO_NOT_IMPLEMENTED;
- AO_NO_MEMORY;
- AO_SESSION_NOT_ACTIVE。

13.2.12 BLOB

(1) BLOB_APPEND

目的：将字节序列附加到二进制大对象。

允许在事务外部修改二进制大型对象，但建议激活事务。参数：value (Type = S_BYTE)

字节序列。Java 调用序列：blob.append(value)；

返回：None。

可能的异常详见 13.3.18 节如下函数：

- AO_BAD_PARAMETER;
- AO_CONNECTION_LOST;
- AO_IMPLEMENTATION_PROBLEM;
- AO_NOT_IMPLEMENTED;
- AO_NO_MEMORY;
- AO_SESSION_NOT_ACTIVE。

(2) BLOB_COMPARE

目的：比较二进制大对象的内容。不比较标题。

参数：aBlob (Type = T_BLOB)

要比较的 blob。Java 调用序列：T_BOOLEAN contentEqual = blob.compare (aBlob) ;

返回：返回名称：contentEqual

返回类型：T_BOOLEAN

标记比较的 blob 是否相等。可能的异常详见 13.3.18 节如下函数：

——AO_BAD_PARAMETER;
 ——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

(3) BLOB_DESTROY

目的：销毁服务器上的对象。此方法用于告诉服务器客户端不再使用 Blob 对象。销毁方法后访问此对象将导致异常。

参数：None.

Java 调用序列：blob.destroy ()

返回：None.

可能的异常详见 13.3.18 节如下函数：

——AO_BAD_PARAMETER;
 ——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

(4) BLOB_GET

目的：获取二进制大对象的一部分。

参数：offset (Type = T_LONG)

blob 中数据的起始位置。length (Type = T_LONG)

从 blob 请求的字节数。Java 调用序列：S_BYTE byteStream = blob.get (offset,length) ;

返回：返回名称：byteStream

返回类型：S_BYTE

BLOB 数据的请求部分。可能的异常详见 13.3.18 节如下函数：

——AO_BAD_PARAMETER;

- AO_CONNECTION_LOST;
- AO_IMPLEMENTATION_PROBLEM;
- AO_NOT_IMPLEMENTED;
- AO_NO_MEMORY;
- AO_SESSION_NOT_ACTIVE。

(5) BLOB_GETHEADER

目的：获取二进制大对象的头。

参数：None.

Java 调用序列：T_STRING header = blob.getHeader () ;

返回：返回名称：header

返回类型：T_STRING

The blob header.

可能的异常详见 13.3.18 节如下函数：

- AO_CONNECTION_LOST;
- AO_IMPLEMENTATION_PROBLEM;
- AO_NOT_IMPLEMENTED;
- AO_NO_MEMORY;
- AO_SESSION_NOT_ACTIVE。

(6) BLOB_GETLENGTH

目的：获取二进制大对象的长度，而不加载它。

参数：None.

Java 调用序列：T_LONG length = blob.getLength () ;

返回：返回名称：length

返回类型：T_LONG

斑点长度。可能的异常详见 13.3.18 节如下函数：

- AO_CONNECTION_LOST;
- AO_IMPLEMENTATION_PROBLEM;
- AO_NOT_IMPLEMENTED;
- AO_NO_MEMORY;
- AO_SESSION_NOT_ACTIVE。

(7) BLOB_SET

目的：清除二进制大对象并设置新数据。

允许在事务外部修改二进制大型对象，但建议激活事务。参数：value (Type = S_BYTE)

新的 blob 数据。Java 调用序列：blob.set (value) ;

返回: None.

可能的异常详见 13.3.18 节如下函数:

——AO_BAD_PARAMETER;
 ——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

(8) BLOB_SETHEDER

目的: 设置二进制大对象的头。

允许在事务外部修改二进制大型对象, 但建议激活事务。参数: header (Type = T_STRING)

Blob 头。Java 调用序列: blob.setHeader (header);

返回: None.

可能的异常详见 13.3.18 节如下函数:

——AO_BAD_PARAMETER;
 ——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

13.2.13 COLUMN

(1) COLUMN_DESTROY

目的: 销毁服务器上的对象。此方法用于告诉服务器客户端不再使用此对象。销毁方法后访问此对象将导致异常。

参数: None.

Java 调用序列: column.destroy ()

返回: None.

可能的异常详见 13.3.18 节如下函数:

——AO_BAD_PARAMETER;
 ——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

(2) COLUMN_GETDATATYPE

目的：获取列的数据类型。

此方法始终返回此本地列所属的测量数量的数据类型，与此本地列所属值矩阵的值矩阵模式此本地列所属值矩阵的值矩阵模式参数：None.

Java 调用序列：DataType dataType = column.getDataType ();

返回：返回名称：dataType

返回类型：DataType

列的数据类型。可能的异常详见 13.3.18 节如下函数：

——AO_CONNECTION_LOST;
——AO_IMPLEMENTATION_PROBLEM;
——AO_NOT_IMPLEMENTED;
——AO_NO_MEMORY;
——AO_SESSION_NOT_ACTIVE。

(3) COLUMN_GETFORMULA

目的：获取列的公式。由于 asam ods 中当前没有公式的规范，因此不应使用此方法；如果使用此方法，它将返回空字符串。参数：None.

Java 调用序列：T_STRING formula = column.getFormula ();

返回：返回名称：formula

返回类型：T_STRING

列的公式。可能的异常详见 13.3.18 节如下函数：

——AO_CONNECTION_LOST;
——AO_IMPLEMENTATION_PROBLEM;
——AO_NOT_IMPLEMENTED;
——AO_NO_MEMORY;
——AO_SESSION_NOT_ACTIVE。

(4) COLUMN_GETGENERATIONPARAMETERS

目的：获取列的生成参数。

如果不存在生成参数，则返回一个空序列，其类型为 double，长度为 0。如果列所属值矩阵的值矩阵模式为“CALCULATED”，则返回一个空序列，其类型为 double，长度为 0。隐式列的生成参数的数据类型是在相应的测量数量上给定的数据类型。原始数据生成参数的数据类型始终为 T ou double。参数：None.

Java 调用序列：TS_UNION genPar = column.getGenerationParameters ();

返回：返回名称：genPar

返回类型：TS_UNION

The generation parameters.

可能的异常详见 13.3.18 节如下函数：

——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

(5) COLUMN_GETNAME

目的：获取列的名称。

参数：None.

Java 调用序列：Name columnName = column.getName () ;

返回：返回名称：columnName

返回类型：Name

列的名称。可能的异常详见 13.3.18 节如下函数：

——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

(6) COLUMN_GETRAWDATATYPE

目的：获取原始值的数据类型（它是从基属性“raw_data type”派生的属性值）。

如果本地列不包含原始值，则返回相应测量数量的数据类型。此外，如果列所属值矩阵的值矩阵模式为“CALCULATED”，则返回测量数量的数据类型。参数：None.

Java 调用序列：DataType dt = column.getRawDataType () ;

返回：返回名称：dt

返回类型：DataType

原始值的数据类型。可能的异常详见 13.3.18 节如下函数：

——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

(7) COLUMN_GETSEQUENCEREPRESENTATION

目的：获取列的序列表示形式。

当列所属的值矩阵的值矩阵模式为“CALCULATED”时，序列表示总是显式的。参数：None.

Java 调用序列: `T_LONG seqRep = column.getSequenceRepresentation ( ) ;`

返回: 返回名称: `seqRep`

返回类型: `T_LONG`

列的序列表示形式。这是根据基本模型的数字顺序表示的整数值。可能的异常详见 13.3.18 节如下

函数:

——`AO_CONNECTION_LOST`;
——`AO_IMPLEMENTATION_PROBLEM`;
——`AO_NOT_IMPLEMENTED`;
——`AO_NO_MEMORY`;
——`AO_SESSION_NOT_ACTIVE`。

(8) COLUMN_GETSOURCEMQ

目的: 获取源测量数量。

参数: `None`。

Java 调用序列: `InstanceElement instElem = column.getSourceMQ ( ) ;`

返回: 返回名称: `instElem`

返回类型: `InstanceElement`

The source measurement quantity.

可能的异常详见 13.3.18 节如下函数:

——`AO_CONNECTION_LOST`;
——`AO_IMPLEMENTATION_PROBLEM`;
——`AO_NOT_IMPLEMENTED`;
——`AO_NO_MEMORY`;
——`AO_SESSION_NOT_ACTIVE`。

(9) COLUMN_GETUNIT

目的: 获取列的单位。

该方法根据当前设置返回值的单位。最初在会话中, 这是存储中给定的单元。为特定列调用 `setUnit ( )` 后, 将为此列返回 `setUnit ( )` 指定的单位。因此, ODS 服务器将根据以下策略检索返回的单元: 1) 如果之前 `setUnit (..)` 已经为特定列设置了一个单元, 则返回该单元, 否则

2) 如果相关的度量数量有一个单位(即, 它有一个从引用基类型 `AOUnit` 实例的基关系“Unit”派生的关系), 则返回该单位的名称, 否则 3) 如果相关的度量数量与某个数量有关系(即从基本关系“数量”派生的关系引用的是基本类型 `AOQuantity` 的实例), 并且此数量具有默认单位(即, 它具有从引用基本类型 `AOUnit` 的实例的基本关系“默认单元”派生的关系), 则该单位返回的名称, 否则

4) 返回空字符串。

参数: None.

Java 调用序列: T_STRING unit = column.getUnit () ;

返回: 返回名称: unit

返回类型: T_STRING

The unit of the column.

可能的异常详见 13.3.18 节如下函数:

——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

(10) COLUMN_ISINDEPENDENT

目的: 该列是独立列吗

参数: None.

Java 调用序列: T_BOOLEAN independent = column.isIndependent () ;

返回: 返回名称: independent

返回类型: T_BOOLEAN

列的独立标志。可能的异常详见 13.3.18 节如下函数:

——AO_BAD_PARAMETER;
 ——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

(11) COLUMN_ISSCALING

目的: 该列是缩放列吗

参数: None.

Java 调用序列: T_BOOLEAN scaling = column.isScaling () ;

返回: 返回名称: scaling

返回类型: T_BOOLEAN

Tells if the column is a scaling column.

可能的异常详见 13.3.18 节如下函数:

——AO_BAD_PARAMETER;
 ——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;

- AO_NOT_IMPLEMENTED;
- AO_NO_MEMORY;
- AO_SESSION_NOT_ACTIVE。

(12) COLUMN_SETFORMULA

目的: 设置列的公式。由于目前 ASAM ODS 中没有公式的规范, 因此不应使用此方法。参数: formula

(Type = T_STRING)

公式。Java 调用序列: column.setFormula (formula) ;

返回: None.

可能的异常详见 13.3.18 节如下函数:

- AO_BAD_PARAMETER;
- AO_CONNECTION_LOST;
- AO_IMPLEMENTATION_PROBLEM;
- AO_NOT_IMPLEMENTED;
- AO_NO_MEMORY;
- AO_SESSION_NOT_ACTIVE。

(13) COLUMN_SETGENERATIONPARAMETERS

目的: 设置生成参数。

隐式列生成参数的数据类型必须是在相应的度量值数量处给定的数据类型。原始数据的生成参数的

数据类型必须始终为 t_double。参数: genPar (Type=TS_UNION)

生成参数。Java 调用序列: column.setGenerationParameter (genPar) ;

返回: None.

可能的异常详见 13.3.18 节如下函数:

- AO_BAD_PARAMETER;
- AO_CONNECTION_LOST;
- AO_IMPLEMENTATION_PROBLEM;
- AO_NOT_IMPLEMENTED;
- AO_NO_MEMORY;
- AO_SESSION_NOT_ACTIVE;
- AO_TRANSACTION_NOT_ACTIVE。

(14) COLUMN_SETINDEPENDENT

目的: 将列设置为独立列。

参数: independent (Type = T_BOOLEAN)

独立标志的新值。Java 调用序列: column.setIndependent (independent) ;

返回: None.

可能的异常详见 13.3.18 节如下函数：

——AO_BAD_PARAMETER;
 ——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE;
 ——AO_TRANSACTION_NOT_ACTIVE。

(15) COLUMN_SETSCALING

目的：将列设置为缩放列。

参数：scaling (Type = T_BOOLEAN)

缩放标志的新值。Java 调用序列：column.setScaling (scaling) ;

返回：None.

可能的异常详见 13.3.18 节如下函数：

——AO_BAD_PARAMETER;
 ——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE;
 ——AO_TRANSACTION_NOT_ACTIVE。

(16) COLUMN_SETSEQUENCEREPRESENTATION

目的：设置新列的序列表示形式。

参数：seqRep (Type=T_LONG)

序列表示。这是根据基本模型的枚举序列表示枚举的整数值。Java 调用序列：

column.setSequenceRepresentation (seqRep) ;

返回：None.

可能的异常详见 13.3.18 节如下函数：

——AO_BAD_PARAMETER;
 ——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE;
 ——AO_TRANSACTION_NOT_ACTIVE。

(17) COLUMN_SETUNIT

目的：设置列的单位。这只是获取列值的临时单位。这个单元不存储在物理存储器中；但是它在当前会话结束或下一个 `setUnit()` 请求之前一直有效。

如果此列所属的值矩阵的值矩阵模式为“STORAGE”，则此方法不会影响从值矩阵返回的值。如果此方法提供空字符串，则后续请求列值的单位将重置为存储在物理存储中的单位（与相应的测量数量相关）。如果需要永久修改单位，客户需要更改相应测量数量的关系。参数：
`unit (Type = T_STRING)`

物理单位。Java 调用序列：`column.setUnit (unit) ;`

返回：None.

可能的异常详见 13.3.18 节如下函数：

——AO_BAD_PARAMETER;
——AO_CONNECTION_LOST;
——AO_IMPLEMENTATION_PROBLEM;
——AO_NOT_IMPLEMENTED;
——AO_NO_MEMORY;
——AO_SESSION_NOT_ACTIVE.

13.2.14 ELEMRESULTSETTEXTSEQITERATOR**(1) ELEMRESULTSETTEXTSEQITERATOR_DESTROY**

目的：销毁迭代器并释放关联的内存。

参数：None.

Java 调用序列：`ersIter.destroy () ;`

返回：None.

可能的异常详见 13.3.18 节如下函数：

——AO_CONNECTION_LOST;
——AO_IMPLEMENTATION_PROBLEM;
——AO_NOT_IMPLEMENTED;
——AO_NO_MEMORY;
——AO_SESSION_NOT_ACTIVE.

(2) ELEMRESULTSETTEXTSEQITERATOR_GETCOUNT

目的：获取迭代器可访问的元素总数。

参数：None.

Java 调用序列：`T_LONG cnt = ersIter.getCount () ;`

返回：返回名称：`count`

返回类型：`T_LONG`

迭代器可访问的元素数。可能的异常详见 13.3.18 节如下函数：

——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

(3) ELEMRESULTSETTEXTSEQITERATOR_NEXTN

目的：从序列中获取下 n 个元素。参数：how_many (Type = T_LONG)

请求的元素数。Java 调用序列：T_LONG how_many = 10;

ElemResultSetExt[] elemResults = ersIter.nextN (how_many)

返回：返回名称：elemResults

返回类型：ElemResultSetExtSequence

元素结果集序列中的下一个 n 属性值。可能的异常详见 13.3.18 节如下函数：

——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

(4) ELEMRESULTSETTEXTSEQITERATOR_NEXTONE

目的：从序列中获取下一个元素。

参数：None.

Java 调用序列：ElemResultSetExt elemResult = ersIter.nextOne ()

返回：返回名称：elemResult

返回类型：ElemResultSetExt

元素结果集序列中的下一个属性值。可能的异常详见 13.3.18 节如下函数：

——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

(5) ELEMRESULTSETTEXTSEQITERATOR_RESET

目的：将元素序列中的指针重置为第一个元素。

参数：None.

Java 调用序列：ersIter.reset ();

返回: None.

可能的异常详见 13.3.18 节如下函数:

- AO_CONNECTION_LOST;
- AO_IMPLEMENTATION_PROBLEM;
- AO_NOT_IMPLEMENTED;
- AO_NO_MEMORY;
- AO_SESSION_NOT_ACTIVE。

13.2.15 ENUMERATIONDEFINITION

(1) ENUMERATIONDEFINITION_ADDITEM

目的: 向枚举中添加新项。此方法修改应用程序模型, 仅允许超级用户使用。

项的名称不能超过基础物理存储的最大名称长度。物理存储的当前规范将枚举项的长度限制为最多 128 个字符。参数: itemName (Type = T_STRING)

新项的名称。Java 调用序列: enumDef.addItem (itemName);

返回: None.

可能的异常详见 13.3.18 节如下函数:

- 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。

(2) ENUMERATIONDEFINITION_GETINDEX

目的: 获取枚举的索引。

参数: None.

Java 调用序列: T_LONG enumIndex = enumDef.getIndex ();

返回: 返回名称: enumIndex

返回类型: T_LONG

枚举的索引。可能的异常详见 13.3.18 节如下函数:

- AO_BAD_PARAMETER;
- AO_CONNECTION_LOST;
- AO_IMPLEMENTATION_PROBLEM;

——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

(3) ENUMERATIONDEFINITION_GETITEM

目的：获取项的值。

参数：itemName (Type = T_STRING)

项的名称。Java 调用序列：T_LONG item = enumDef.getItem (itemName) ;

返回：返回名称：item

返回类型：T_LONG

项目的编号。

可能的异常详见 13.3.18 节如下函数：

——AO_BAD_PARAMETER;
 ——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE;
 ——AO_NOT_FOUND。

(4) ENUMERATIONDEFINITION_GETITEMNAME

目的：获取项的名称。

参数：item (Type = T_LONG)

项的值。Java 调用序列：T_STRING itemName = enumDef.getItemName (item) ;

返回：返回名称：itemName

返回类型：T_STRING

项的名称。可能的异常详见 13.3.18 节如下函数：

——AO_BAD_PARAMETER;
 ——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE;
 ——AO_NOT_FOUND。

(5) ENUMERATIONDEFINITION_GETNAME

目的：获取枚举的名称。

参数: None.

Java 调用序列: `T_STRING enumName = enumDef.getName ()`;

返回: 返回名称: `enumName`

返回类型: `T_STRING`

枚举的名称。可能的异常详见 13.3.18 节如下函数:

- `AO_BAD_PARAMETER`;
- `AO_CONNECTION_LOST`;
- `AO_IMPLEMENTATION_PROBLEM`;
- `AO_NOT_IMPLEMENTED`;
- `AO_NO_MEMORY`;
- `AO_SESSION_NOT_ACTIVE`。

(6) `ENUMERATIONDEFINITION_LISTITEMNAMES`

目的: 列出枚举的可能名称。列表的排序顺序是项的值。第一项的值为 0 (零)。

参数: None.

Java 调用序列: `Name[] nameList = enumDef.listItemNames ()`;

返回: 返回名称: `nameList`

返回类型: `NameSequence`

列出枚举项的所有可能名称。可能的异常详见 13.3.18 节如下函数:

- `AO_BAD_PARAMETER`;
- `AO_CONNECTION_LOST`;
- `AO_IMPLEMENTATION_PROBLEM`;
- `AO_NOT_IMPLEMENTED`;
- `AO_NO_MEMORY`;
- `AO_SESSION_NOT_ACTIVE`。

(7) `ENUMERATIONDEFINITION_RENAMEITEM`

目的: 重命名枚举项。此方法修改应用程序模型, 仅允许超级用户使用。参数: `oldItemName (Type = T_STRING)`

The existing name of the item.

`newItemName (Type = T_STRING)`

the new name of the item.

Java 调用序列: `enumDef.renameItem (oldItemName, newItemName)`;

返回: None.

可能的异常详见 13.3.18 节如下函数:

- `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。

(8) ENUMERATIONDEFINITION_SETNAME

目的：设置枚举的名称。此方法修改应用程序模型，仅允许超级用户使用。

枚举定义的名称不能超过基础物理存储的最大名称长度。当前物理存储的规范将枚举名称的长度限制为最多 30 个字符。参数：enumName (Type = T_STRING)

枚举的名称。Java 调用序列：enumDef.setName (enumName)；

返回：None.

可能的异常详见 13.3.18 节如下函数：

——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。

13.2.16 INSTANCEELEMENT

(1) INSTANCEELEMENT_ADDINSTANCEATTRIBUTE

目的：向实例添加实例属性。实例属性在客户机上构建为名称/值/单位元组。此方法必须将数据从客户端复制到服务器。属性的名称必须唯一。

允许在事务外部添加实例属性，但建议激活事务。注意：由于物理存储的当前规范将实例属性的名称长度限制为最多 30 个字符，因此客户端必须注意不要超过此限制。参数：instAttr (类型=名称值单位)

要添加的实例属性。Java 调用序列：instElem.addInstanceAttribute (instAttr)；

返回：None.

可能的异常详见 13.3.18 节如下函数：

——AO_BAD_PARAMETER;
 ——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;

——AO_NO_MEMORY;
——AO_SESSION_NOT_ACTIVE。

(2) INSTANCEELEMENT_COMPARE

目的：比较实例。比较了应用元素的 ID 和实例的 ID。首先比较应用元素的 ID。

参数：compleObj（类型=instanceElement）

要与之比较的实例。Java 调用序列：T_LONGLONG diff = ieObj.compare（compleObj）；

返回：返回名称：diff

返回类型：T_LONGLONG

身份证的区别。意思是：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.

可能的异常详见 13.3.18 节如下函数：

——AO_BAD_PARAMETER;
——AO_CONNECTION_LOST;
——AO_IMPLEMENTATION_PROBLEM;
——AO_NOT_IMPLEMENTED;
——AO_NO_MEMORY;
——AO_SESSION_NOT_ACTIVE。

(3) INSTANCEELEMENT_CREATERELATEDINSTANCES

目的：使用与实际实例相关的实例创建列表。属性以序列的名称给出。属性的值在值序列中给出。

不同值序列中的索引与一个实例匹配。相关实例实例序列中的索引与值序列中具有相同索引的实例匹配。

注意：由于物理存储的当前规范将实例属性的名称长度限制为最多 30 个字符，因此如果客户端使用此方法调用指定实例属性（应用程序模型未提供的属性），则必须注意不要超过此限制。

参数：applRel（类型=应用关系）将创建相关实例的应用程序关系。attributes（类型=NameValueSeqUnitSequence）

新 创 建 实 例 的 属 性 。 relatedInstances（类型 = ApplicationRelationInstanceElementSeqSequence）

具有不同应用程序关系的相关实例的列表。Java 调用序列：InstanceElement[] instElems = ieObj.createRelatedInstances（applRel, attributes,relatedInstances）；

返回：返回名称：instElems

返回类型：InstanceElementSequence

包含新创建实例的列表。可能的异常详见 13.3.18 节如下函数：

——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) INSTANCEELEMENT_CREATERELATION

目的：在当前实例和给定实例之间创建关系。检查关系的应用元素是否与实例的应用元素匹配。

允许在事务外部修改实例，但建议激活事务。参数：relation（类型= ApplicationRelation）

应用关系。instElem（类型= InstanceElement）

实例。Java 调用序列：instElem.createRelation（relation,instElem）；

返回：None。

可能的异常详见 13.3.18 节如下函数：

——AO_BAD_PARAMETER;
 ——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_INVALID_RELATION;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

(5) INSTANCEELEMENT_DEEPCOPY

目的：提供易于使用和有效的复制机制，例如服务器内部的层次结构（例如，使用所有测试复制项目或使用所有度量复制测试）。深度复制只跟随子引用，而不跟随信息引用。

示例：复制 aoMeasurement 类型的元素不包括复制 aoMeasurementQuantity 类型的引用元素。AOMeasurement 类型的复制实例将引用与或原始相同的度量值。想要复制度量数量的应用程序也必须自己执行此操作（包括设置适当的引用），例如，通过另一个调用来调用 Shallowcopy；在这种情况下，不需要 DeepCopy，因为 AoMeasurementQuantity 没有子级。

参数：newName（类型=T_字符串）

新实例的名称。如果要创建新版本，则此参数可以为副本使用相同的名称。在这种情况下，必须提供新版本。newVersion（类型=T_字符串）

新实例的版本。如果提供新名称，此参数可能为空。Java 调用序列：InstanceElement newInstElem = instElem.deepCopy（newName,newVersion）；

返回：返回名称：newInstElem

返回类型：InstanceElement

The reference to the copied instance hierarchy.

可能的异常详见 13.3.18 节如下函数：

- AO_BAD_PARAMETER;
- AO_CONNECTION_LOST;
- AO_IMPLEMENTATION_PROBLEM;
- AO_NOT_IMPLEMENTED;
- AO_NO_MEMORY;
- AO_SESSION_NOT_ACTIVE;
- AO_TRANSACTION_NOT_ACTIVE。

(6) INSTANCEELEMENT_DESTROY

目的：销毁服务器上的对象。此方法用于告诉服务器客户端不再使用此对象。销毁方法后访问此对象将导致异常。参数：None。

Java 调用序列：ieObj.destroy ()

返回：None。

可能的异常详见 13.3.18 节如下函数：

- AO_BAD_PARAMETER;
- AO_CONNECTION_LOST;
- AO_IMPLEMENTATION_PROBLEM;
- AO_NOT_IMPLEMENTED;
- AO_NO_MEMORY;
- AO_SESSION_NOT_ACTIVE。

(7) INSTANCEELEMENT_GETAPPLICATIONELEMENT

目的：获取实例的应用元素。

参数：None。

Java 调用序列：ApplicationElement applElem = instElem.getApplicationElement () ;

返回：返回名称：applElem

返回类型：ApplicationElement

从中派生实例的应用元素。可能的异常详见 13.3.18 节如下函数：

- AO_CONNECTION_LOST;
- AO_IMPLEMENTATION_PROBLEM;
- AO_NOT_IMPLEMENTED;
- AO_NO_MEMORY;
- AO_SESSION_NOT_ACTIVE。

(8) INSTANCEELEMENT_GETASAMPATH

目的：获取 ASAM-Path 实例的路径

参数：None。

Java 调用序列: `Name asamPath = instElem.getAsamPath ( ) ;`

返回: 返回名称: `asamPath`

返回类型: `Name`

The ASAM path to the instance.

可能的异常详见 13.3.18 节如下函数:

——`AO_CONNECTION_LOST`;
 ——`AO_IMPLEMENTATION_PROBLEM`;
 ——`AO_NOT_IMPLEMENTED`;
 ——`AO_NO_MEMORY`;
 ——`AO_SESSION_NOT_ACTIVE`。

(9) INSTANCEELEMENT_GETID

目的: 获取实例的 ID。

参数: `None`。

Java 调用序列: `T_LONGLONG ield = instElem.getId ( ) ;`

返回: 返回名称: `ield`

返回类型: `T_LONGLONG`

实例的 ID。可能的异常详见 13.3.18 节如下函数:

——`AO_CONNECTION_LOST`;
 ——`AO_IMPLEMENTATION_PROBLEM`;
 ——`AO_NOT_IMPLEMENTED`;
 ——`AO_NO_MEMORY`;
 ——`AO_SESSION_NOT_ACTIVE`。

(10) INSTANCEELEMENT_GETINITIALRIGHTS

目的: 检索给定对象的初始权限的访问控制列表信息。

参数: `None`。

Java 调用序列: `InitialRight[] initialRights = instElem.getInitialRights ( ) ;`

返回: 返回名称: `initialRights`

返回类型: `InitialRightSequence`

访问控制列表项具有给定应用元素的初始权限。可能的异常详见 13.3.18 节如下函数:

——`AO_BAD_PARAMETER`;
 ——`AO_CONNECTION_LOST`;
 ——`AO_IMPLEMENTATION_PROBLEM`;
 ——`AO_NOT_IMPLEMENTED`;
 ——`AO_NO_MEMORY`;
 ——`AO_SESSION_NOT_ACTIVE`。

(11) INSTANCEELEMENT_GETNAME

目的：获取实例的名称。

参数：None.

Java 调用序列：Name ieName = instElem.getName () ;

返回：返回名称：ieName

返回类型：Name

实例的名称。可能的异常详见 13.3.18 节如下函数：

——AO_CONNECTION_LOST;
——AO_IMPLEMENTATION_PROBLEM;
——AO_NOT_IMPLEMENTED;
——AO_NO_MEMORY;
——AO_SESSION_NOT_ACTIVE。

(12) INSTANCEELEMENT_GETRELATEDINSTANCES

目的：获取相关实例。应用程序关系和相关实例的名称指定列出的实例。模式区分大小写，可以包含通配符。

参数：applRel（类型= ApplicationRelation）

应用关系。iePattern（类型= Pattern）

相关实例名称的名称或搜索模式。Java 调用序列：InstanceElementIterator ieliterator =
instElem.getRelatedInstances (applRel,iePattern) ;

返回：返回名称：ieliterator

返回类型：InstanceElementIterator

相关实例。可能的异常详见 13.3.18 节如下函数：

——AO_BAD_PARAMETER;
——AO_CONNECTION_LOST;
——AO_IMPLEMENTATION_PROBLEM;
——AO_INVALID_RELATION;
——AO_NOT_IMPLEMENTED;
——AO_NO_MEMORY;
——AO_SESSION_NOT_ACTIVE。

(13) INSTANCEELEMENT_GETRELATEDINSTANCESBYRELATIONSHIP

目的：获取相关实例的列表。关系和相关实例的名称指定列出的实例。模式区分大小写，可以包含通配符。

参数：ieRelationship（类型=关系）

请求的关系。iePattern（类型=模式）

相关实例名称的名称或搜索模式。

Java 调用序列: `InstanceElementIterator ieliterator = instElem.getRelatedInstancesByRelationship (ieRelationship, iePattern);`

返回: 返回名称: `ieliterator`

返回类型: `InstanceElementIterator`

The related instances.

可能的异常详见 13.3.18 节如下函数:

——`AO_BAD_PARAMETER;`
 ——`AO_CONNECTION_LOST;`
 ——`AO_IMPLEMENTATION_PROBLEM;`
 ——`AO_INVALID_RELATIONSHIP;`
 ——`AO_NOT_IMPLEMENTED;`
 ——`AO_NO_MEMORY;`
 ——`AO_SESSION_NOT_ACTIVE.`

(14) INSTANCEELEMENT_GETRIGHTS

目的: 检索给定对象的访问控制列表信息。

参数: `None`.

Java 调用序列: `ACL[] aclEntries = instElem.getRights ();`

返回: 返回名称: `aclEntries`

返回类型: `ACLSequence`

给定应用元素的访问控制列表项。可能的异常详见 13.3.18 节如下函数:

——`AO_BAD_PARAMETER;`
 ——`AO_CONNECTION_LOST;`
 ——`AO_IMPLEMENTATION_PROBLEM;`
 ——`AO_NOT_IMPLEMENTED;`
 ——`AO_NO_MEMORY;`
 ——`AO_SESSION_NOT_ACTIVE.`

(15) INSTANCEELEMENT_GETVALUE

目的: 获取实例的给定属性的属性值 (名称、值和单位)。此方法不会返回关系属性的值, 请使用 `getRelatedInstances` 方法。

参数: `attrName` (类型=名称)

请求的属性的名称。

Java 调用序列: `NameValueUnit nameValueUnit = instElem.getValue (attrName);`

返回: 返回名称: `nameValueUnit`

返回类型: `NameValueUnit`

属性值。可能的异常详见 13.3.18 节如下函数：

——AO_BAD_PARAMETER;
 ——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

(16) INSTANCEELEMENT_GETVALUEBYBASENAME

目的：获取从实例的给定基属性派生的属性的属性值（值和单位）。基名称不区分大小写，不能包含通配符。

参数：baseAttrName（类型=名称）

所请求属性的基名称。Java 调用序列：NameValueUnit nameValueUnit =
 instElem.getValueByBaseName（baseAttrName）；

返回：返回名称：nameValueUnit

返回类型：NameValueUnit

属性值。可能的异常详见 13.3.18 节如下函数：

——AO_BAD_PARAMETER;
 ——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

(17) INSTANCEELEMENT_GETVALUEINUNIT

目的：获取实例的给定属性的属性值（名称、值和单位）。参数：attr（类型=名称单位）请求的属性的名称和属性值的单位。Java 调用序列：NameValueUnit nameValueUnit =
 instElem.getValueInUnit（attr）；

返回：返回名称：nameValueUnit

返回类型：NameValueUnit

属性值，值转换为请求的单位。可能的异常详见 13.3.18 节如下函数：

——AO_BAD_PARAMETER;
 ——AO_CONNECTION_LOST;
 ——AO_INCOMPATIBLE_UNITS;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;

——AO_SESSION_NOT_ACTIVE。

(18) INSTANCEELEMENT_GETVALUESEQ

目的：获取由应用程序或实例属性的名称指定的值序列。名称序列可以对实例的所有属性使用模式(*)。这意味着应用程序属性以及实例属性将被传递。参数：attrNames (类型=名称序列) 要报告的属性的名称。Java 调用序列：Name attrNames[];

...

NameValueUnit values[] = ieObj.getValueSeq (attrNames) ;

返回：返回名称：values

返回类型：NameValueUnitSequence

The sequence of the attribute values.

可能的异常详见 13.3.18 节如下函数：

——AO_BAD_PARAMETER;
 ——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

(19) INSTANCEELEMENT_LISTATTRIBUTES

目的：从实例中获取属性名。未列出为关系保留的属性。

参数：iaPattern (类型=模式)

属性名称的名称或搜索模式。aType (类型= AttrType)

请求的属性类型。

Java 调用序列：Name[] ieNames = instElem.listAttributes (iaPattern,aType) ;

返回：返回名称：ieNames

返回类型：NameSequence

属性的名称。可能的异常详见 13.3.18 节如下函数：

——AO_BAD_PARAMETER;
 ——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_INVALID_ATTRIBUTE_TYPE;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

(20) INSTANCEELEMENT_LISTRELATEDINSTANCES

目的：获取相关实例的名称。应用程序关系和相关实例的名称指定列出的名称。模式区分大小写，可以包含通配符。

参数：ieRelation（类型=应用关系）

应用关系。iePattern（类型=模式）

相关实例名称的名称或搜索模式。Java 调用序列：NameIterator ieNameIterator = instElem.listRelatedInstances（ieRelation,iePattern）；

返回：返回名称：ieNameIterator

返回类型：NameIterator

相关实例的名称。可能的异常详见 13.3.18 节如下函数：

——AO_BAD_PARAMETER；
 ——AO_CONNECTION_LOST；
 ——AO_IMPLEMENTATION_PROBLEM；
 ——AO_NOT_IMPLEMENTED；
 ——AO_NO_MEMORY；
 ——AO_SESSION_NOT_ACTIVE。

（21）INSTANCEELEMENT_LISTRELATEDINSTANCESBYRELATIONSHIP

目的：获取相关实例的名称。关系和相关实例的名称指定列出的名称。模式区分大小写，可以包含通配符。

参数：ieRelationship（Type = Relationship）

请求的关系。iePattern（Type = Pattern）

相关实例名称的名称或搜索模式。Java 调用序列：NameIterator ieNameIterator = instElem.listRelatedInstancesByRelationship（ieRelationship,iePattern）；

返回：返回名称：ieNameIterator

返回类型：NameIterator

相关实例的名称。可能的异常详见 13.3.18 节如下函数：

——AO_BAD_PARAMETER；
 ——AO_CONNECTION_LOST；
 ——AO_IMPLEMENTATION_PROBLEM；
 ——AO_INVALID_RELATIONSHIP；
 ——AO_NOT_IMPLEMENTED；
 ——AO_NO_MEMORY；
 ——AO_SESSION_NOT_ACTIVE。

（22）INSTANCEELEMENT_REMOVEINSTANCEATTRIBUTE

目的：删除实例属性。

允许在事务外部删除实例属性，但建议激活事务。无法删除应用程序属性。参数：attrName（Type

= Name)

要删除的属性的名称。Java 调用序列：instElem.removeInstanceAttribute (attrName) ;

返回：None.

可能的异常详见 13.3.18 节如下函数：

——AO_BAD_PARAMETER;
 ——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_FOUND;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

(23) INSTANCEELEMENT_REMOVERELATION

目的：删除当前实例与给定实例之间的关系。如果不是所有关系都应删除，则必须指定 N:M 关系的实例。

允许在事务外部修改实例，但建议激活事务。参数：applRel (Type = ApplicationRelation)

要删除的关系。instElem_nm (Type = InstanceElement)

从 n:m 关系中进行特定删除的实例。Java 调用序列：instElem.removeRelation (applRel,instElem_nm) ;

返回：None.

可能的异常详见 13.3.18 节如下函数：

——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。

(24) INSTANCEELEMENT_RENAMEINSTANCEATTRIBUTE

目的：重命名实例属性。无法重命名应用程序属性。允许在事务外部重命名实例属性，但建议激活事务。注意：由于物理存储的当前规范将实例属性的名称长度限制为最多 30 个字符，因此客户端必须注意不要超过此限制。参数：oldName (Type = Name)

旧实例属性名。newName (Type = Name)

新实例属性名称。Java 调用序列：instElem.renameInstanceAttribute (oldName,newName) ;

返回：None.

可能的异常详见 13.3.18 节如下函数：

——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。

(25) INSTANCEELEMENT_SETINITIALRIGHTS

目的：应为其设置初始权限的给定用户组。<rights>定义要设置或清除的权限。如果参数<set>设置为'set'，则设置<rights>中的权限，清除所有其他权限。如果参数<set>设置为'add'，则将在<rights>中的权限添加到现有权限中。如果参数<set>设置为'remove'，则将从现有权限中删除<rights>中的权限。参数： *usergroup* (*Type = InstanceElement*)

将修改其初始权限的用户组。 *rights* (*Type = T_LONG*)

用户组的新初始权限。权限常量在接口 *SecurityRights* 中定义。接口定义语言 IDL 不允许设置枚举值；因此必须在接口中完成常量定义。 *refAid* (*Type = T_LONGLONG*)

将使用初始权限的引用应用元素的 ID。如果设置了 *norefaid*，则独立于 *application* 元素的每个新实例都将使用初始权限。 *set* (*Type = RightsSet*)

如何处理新的初始权利。Java 调用序列： *instElem.setInitialRights* (*usergroup, rights, refAid, set*) ;
 返回： *None*。

可能的异常详见 13.3.18 节如下函数：

——AO_BAD_PARAMETER;
 ——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE;
 ——AO_TRANSACTION_NOT_ACTIVE。

(26) INSTANCEELEMENT_SETNAME

目的：设置实例的名称。

允许在事务外部修改实例的名称，但建议激活事务。参数： *iaName* (*Type = Name*)

实例属性名称。Java 调用序列： *instElem.setName* (*iaName*) ;

返回： *None*。

可能的异常详见 13.3.18 节如下函数：

——AO_BAD_PARAMETER;
 ——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

(27) INSTANCEELEMENT_SETRIGHTS

目的：应为其设置权限的给定用户组。<rights>定义要设置或清除的权限。如果参数<set>设置为'set'，则设置<rights>中的权限，清除所有其他权限。如果参数<set>设置为'add'，则将<rights>中的权限添加到现有权限中。如果参数<set>设置为'remove'，则将从现有权限中删除<rights>中的权限。

参数：usergroup (Type = InstanceElement)

将修改其权限的用户组。rights (Type = T_LONG)

用户组的新权限。权限常量在接口 SecurityRights 中定义。接口定义语言 IDL 不允许设置枚举值，因此必须在接口中完成常量定义。set (Type = RightsSet)

如何处理新的权利。

Java 调用序列：instElem.setRights (usergroup, rights, set) ;

返回：None.

可能的异常详见 13.3.18 节如下函数：

——AO_BAD_PARAMETER;
 ——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE;
 ——AO_TRANSACTION_NOT_ACTIVE。

(28) INSTANCEELEMENT_SETVALUE

目的：设置应用程序属性或现有实例属性的值。此方法不会创建新的实例属性；为此，必须使用此接口的方法 addInstanceAttribute ()。

允许在事务外部修改属性值，但建议激活事务。属性的名称由 NameValueUnit 元组的名称指定。

如果设置了应用程序属性标志“唯一”，则检查新值的唯一性。此方法不能用于设置关系属性的值，请使用此接口的方法 createRelation 代替。参数：value (Type = NameValueUnit)

要在实例中设置的值。Java 调用序列：instElem.setValue (value) ;

返回：None.

可能的异常详见 13.3.18 节如下函数：

——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。

(29) INSTANCEELEMENT_SETVALUESEQ

目的：设置应用程序属性的值序列。此方法无法设置实例属性的值。

允许在事务外部修改属性值，但建议激活事务。属性的名称由 NameValueUnit 元组的名称指定。如果设置了应用程序属性标志“唯一”，则检查新值的唯一性。注意：由于物理存储的当前规范将实例属性的名称长度限制为最多 30 个字符，因此如果客户端使用此方法调用指定实例属性（应用程序模型未提供的属性），则必须注意不要超过此限制。

参数：values（Type = NameValueUnitSequence）

要在实例上设置的值的序列。Java 调用序列：instElem.setValueSeq（values）；

返回：None。

可能的异常详见 13.3.18 节如下函数：

——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。

(30) INSTANCEELEMENT_SHALLOWCOPY

目的：为服务器内的实例提供易于使用和有效的复制机制。

新实例获取原始实例中可用的所有属性值和信息关系的副本。新实例与原始实例具有相同的父实例，

但它没有对原始实例的任何子实例的引用。参数：newName（Type = T_STRING）

新实例的名称。如果要创建新版本，此参数可以是

空以对副本使用相同的名称。在这种情况下，必须提供新版本。newVersion（Type = T_STRING）

新实例的版本。如果提供新名称，此参数可能为空。Java 调用序列：InstanceElement newInstElem

= instElem.shallowCopy（newName,newVersion）；

返回：返回名称：newInstElem

返回类型: InstanceElement

对复制实例的引用。可能的异常详见 13.3.18 节如下函数:

——AO_BAD_PARAMETER;
 ——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE;
 ——AO_TRANSACTION_NOT_ACTIVE。

(31) INSTANCEELEMENT_UPCASTMEASUREMENT

目的: 将实例强制转换为度量值。有些面向对象的语言不允许这种转换。

参数: None.

Java 调用序列: Measurement measurement = instElem.upcastMeasurement ();

返回: 返回名称: measurement

返回类型: Measurement

类型度量的实例。可能的异常详见 13.3.18 节如下函数:

——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE;
 ——AO_INVALID_Basetype。

(32) INSTANCEELEMENT_UPCASTSUBMATRIX

目的: 将实例强制转换为子矩阵。有些面向对象的语言不允许这种转换。

参数: None.

Java 调用序列: SubMatrix subMatrix = instElem.upcastSubMatrix ();

返回: 返回名称: subMatrix

返回类型: SubMatrix

Submatrix 类型的实例。可能的异常详见 13.3.18 节如下函数:

——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE;
 ——AO_INVALID_Basetype。

13.2.17 INSTANCEELEMENTITERATOR

(1) INSTANCEELEMENTITERATOR_DESTROY

目的：销毁迭代器并释放关联的内存。

参数：None.

Java 调用序列：instElemIterator.destroy () ;

返回：None.

可能的异常详见 13.3.18 节如下函数：

——AO_CONNECTION_LOST;
——AO_IMPLEMENTATION_PROBLEM;
——AO_NOT_IMPLEMENTED;
——AO_NO_MEMORY;
——AO_SESSION_NOT_ACTIVE。

(2) INSTANCEELEMENTITERATOR_GETCOUNT

目的：获取迭代器可访问的元素总数。

参数：None.

Java 调用序列：T_LONG instanceCount = instElemIterator.getCount () ;

返回：返回名称：instanceCount

返回类型：T_LONG

迭代器可访问的元素数。可能的异常详见 13.3.18 节如下函数：

——AO_CONNECTION_LOST;
——AO_IMPLEMENTATION_PROBLEM;
——AO_NOT_IMPLEMENTED;
——AO_NO_MEMORY;
——AO_SESSION_NOT_ACTIVE。

(3) INSTANCEELEMENTITERATOR_NEXTN

目的：从序列中获取下 n 个元素。参数：how_many (类型=T_long)

请求的元素数。Java 调用序列：InstanceElement[] instElems = instElemIterator.nextN
(how_many) ;

返回：返回名称：instElems

返回类型：InstanceElementSequence

实例序列中接下来的 n 个实例。可能的异常详见 13.3.18 节如下函数：

——AO_BAD_PARAMETER;
——AO_CONNECTION_LOST;
——AO_IMPLEMENTATION_PROBLEM;
——AO_NOT_IMPLEMENTED;

——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

(4) INSTANCEELEMENTITERATOR_NEXTONE

目的：从序列中获取下一个元素。

参数：None.

Java 调用序列：InstanceElement instElem = instElemIterator.nextOne ();

返回：返回名称：instElem

返回类型：InstanceElement

实例序列中的下一个实例。可能的异常详见 13.3.18 节如下函数：

——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

(5) INSTANCEELEMENTITERATOR_RESET

目的：将元素序列中的指针重置为第一个元素。参数：None.

Java 调用序列：instElemIterator.reset ();

返回：None.

可能的异常详见 13.3.18 节如下函数：

——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

13.2.18 MEASUREMENT

Measurement接口扩展了接口 InstanceElement, 因此 InstanceElement 接口的所有方法也适用它。

(1) MEASUREMENT_CREATEMATLINK

目的：创建子矩阵链接。Submitrix 链接仅在当前会话中有效。当会话关闭时，子矩阵链接将被破坏。

参数：None.

Java 调用序列：SMatLink smLink = measurement.createSMatLink ();

返回：返回名称：smLink

返回类型：SMatLink

The new submatrix link.

可能的异常详见 13.3.18 节如下函数：

- AO_CONNECTION_LOST;
- AO_IMPLEMENTATION_PROBLEM;
- AO_NOT_IMPLEMENTED;
- AO_NO_MEMORY;
- AO_SESSION_NOT_ACTIVE。

(2) MEASUREMENT_GETSMATLINKS

目的：获取子矩阵链接的列表。

参数：None.

Java 调用序列：SMatLink[] smLinks = measurement.getSMatLinks () ;

返回：返回名称：smLinks

返回类型：SMatLinkSequence

The available submatrix links.

可能的异常详见 13.3.18 节如下函数：

- AO_CONNECTION_LOST;
- AO_IMPLEMENTATION_PROBLEM;
- AO_NOT_IMPLEMENTED;
- AO_NO_MEMORY;
- AO_SESSION_NOT_ACTIVE。

(3) MEASUREMENT_GETVALUEMATRIX

目的：获取测量值矩阵。

通过调用此方法，返回值矩阵的值矩阵模式将取自上下文变量“value matrix_mode”。有关值矩阵不同模式的详细信息，请参阅此接口的 `getValueMatrixinMode ( )` 方法中的说明。参数：

None.

Java 调用序列：ValueMatrix vm = measurement.getValueMatrix () ;

返回：返回名称：vm

返回类型：ValueMatrix

The value matrix.

可能的异常详见 13.3.18 节如下函数：

- AO_CONNECTION_LOST;
- AO_IMPLEMENTATION_PROBLEM;
- AO_INVALID_VALUEMATRIX_STRUCTURE;
- AO_NOT_IMPLEMENTED;
- AO_NO_MEMORY;
- AO_SESSION_NOT_ACTIVE。

(4) MEASUREMENT_GETVALUEMATRIXINMODE

目的：获取所请求模式下的测量值矩阵。

服务器检查独立列，如果测量中有多个子矩阵，则每个子矩阵中必须正好有一个具有相同测量数量的独立本地列。如果不是这样，则抛出异常 **AO_INVALID_VALUEMATRIX_STRUCTURE**。一般情况下，服务器会抛出异常 **AO_INVALID_VALUEMATRIX_STRUCTURE** 结构，以防由于测量数据而无法创建值矩阵：-如果没有独立的本地列

-如果在不同的子矩阵中有不同的独立局部列

-如果存在具有多个独立局部列的子矩阵。

服务器根据独立本地列的值（按升序）对值进行排序。如果一个测量数量在同一个测量点上有两个不同的值，则由服务器决定返回的值。间隙由 **TS_ValueSeq** (**flagvalue = 0**) 中的适当标志标记。如果请求的模式为“**CALCULATED**”，服务器将返回隐式和原始本地列的计算值。隐式局部列将使用生成参数扩展为显式值序列。存储原始值的本地列将使用原始值和生成参数重新计算。如果请求的模式是“**STORAGE**”，服务器将返回本地列的值，如同它们在物理存储中一样。只有当值矩阵模式为“**STORAGE**”时，才允许将值写入值矩阵。参数：

vmMode（类型=值矩阵模式）

值矩阵的请求模式。Java 调用序列：**ValueMatrix vm = measurement.getValueMatrixInMode (vmMode) ;**

返回：返回名称：**vm**

返回类型：**ValueMatrix**

价值矩阵。可能的异常详见 13.3.18 节如下函数：

——**AO_CONNECTION_LOST**;
 ——**AO_IMPLEMENTATION_PROBLEM**;
 ——**AO_INVALID_VALUEMATRIX_STRUCTURE**;
 ——**AO_NOT_IMPLEMENTED**;
 ——**AO_NO_MEMORY**;
 ——**AO_SESSION_NOT_ACTIVE**。

(5) MEASUREMENT_REMOVESMATLINK

目的：拆下一个子矩阵链接。

参数：**smLink**（类型= **SMatLink**）

要删除的子矩阵链接。

Java 调用序列：**measurement.removeSMatLink (smLink) ;**

返回：**None**。

可能的异常详见 13.3.18 节如下函数：

——**AO_BAD_PARAMETER**;

- AO_CONNECTION_LOST;
- AO_IMPLEMENTATION_PROBLEM;
- AO_NOT_FOUND;
- AO_NOT_IMPLEMENTED;
- AO_NO_MEMORY;
- AO_SESSION_NOT_ACTIVE。

13.2.19 NAMEITERATOR

(1) NAMEITERATOR_DESTROY

目的：销毁迭代器并释放关联的内存。

参数：None.

Java 调用序列：nameiterator.destroy () ;

返回：None.

可能的异常详见 13.3.18 节如下函数：

- AO_CONNECTION_LOST;
- AO_IMPLEMENTATION_PROBLEM;
- AO_NOT_IMPLEMENTED;
- AO_NO_MEMORY;
- AO_SESSION_NOT_ACTIVE。

(2) NAMEITERATOR_GETCOUNT

目的：获取迭代器可访问的元素总数。参数：None.

Java 调用序列：T_LONG nameCount = nameiterator.getCount () ;

返回：返回名称：nameCount

返回类型：T_LONG

迭代器可访问的元素数。可能的异常详见 13.3.18 节如下函数：

- AO_CONNECTION_LOST;
- AO_IMPLEMENTATION_PROBLEM;
- AO_NOT_IMPLEMENTED;
- AO_NO_MEMORY;
- AO_SESSION_NOT_ACTIVE。

(3) NAMEITERATOR_NEXTN

目的：从序列中获取下 n 个元素。

参数：how_many (类型=T_long)

请求的元素数。

Java 调用序列：Name[] names = nameiterator.nextN (how_many) ;

返回：返回名称：names

返回类型：NameSequence

名称序列中的后 n 个名称。可能的异常详见 13.3.18 节如下函数：

——AO_BAD_PARAMETER;
 ——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

(4) NAMEITERATOR_NEXTONE

目的：从序列中获取下一个元素。参数：None.

Java 调用序列：Name name = nameIterator.nextOne ();

返回：返回名称：name

返回类型：Name

名称序列中的下一个名称。名称序列中的下一个名称。可能的异常详见 13.3.18 节如下函数：

——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

(5) NAMEITERATOR_RESET

目的：将元素序列中的指针重置为第一个元素。参数：None.

Java 调用序列：nameIterator.reset ();

返回：None.

可能的异常详见 13.3.18 节如下函数：

——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

13.2.20 NAMEVALUEITERATOR

(1) NAMEVALUEITERATOR_DESTROY

目的：销毁迭代器并释放关联的内存。参数：None.

Java 调用序列：nameValueIterator.destroy ();

返回: None.

可能的异常详见 13.3.18 节如下函数:

- AO_CONNECTION_LOST;
- AO_IMPLEMENTATION_PROBLEM;
- AO_NOT_IMPLEMENTED;
- AO_NO_MEMORY;
- AO_SESSION_NOT_ACTIVE。

(2) NAMEVALUEITERATOR_GETCOUNT

目的: 获取迭代器可访问的元素总数。

参数: None.

Java 调用序列: `T_LONG nvCount = nameValueIterator.getCount ()` ;

返回: 返回名称: `nvCount`

返回类型: `T_LONG`

迭代器可访问的元素数。可能的异常详见 13.3.18 节如下函数:

- AO_CONNECTION_LOST;
- AO_IMPLEMENTATION_PROBLEM;
- AO_NOT_IMPLEMENTED;
- AO_NO_MEMORY;
- AO_SESSION_NOT_ACTIVE。

(3) NAMEVALUEITERATOR_NEXTN

目的: 从序列中获取下 `n` 个元素。

参数: `how_many` (类型=`T_long`)

请求的元素数。Java 调用序列: `NameValue[] nameValues = nameValueIterator.nextN (how_many)` ;

返回: 返回名称: `nameValues`

返回类型: `NameValueSequence`

名称-值对序列中的下一个 `n` 个名称-值对。可能的异常详见 13.3.18 节如下函数:

- AO_BAD_PARAMETER;
- AO_CONNECTION_LOST;
- AO_IMPLEMENTATION_PROBLEM;
- AO_NOT_IMPLEMENTED;
- AO_NO_MEMORY;
- AO_SESSION_NOT_ACTIVE。

(4) NAMEVALUEITERATOR_NEXTONE

目的：从序列中获取下一个元素。

参数：None.

Java 调用序列：NameValue nameValue = nameValueIterator.nextOne () ;

返回：返回名称：nameValue

返回类型：NameValue

名称-值对序列中的下一个名称-值对。可能的异常详见 13.3.18 节如下函数：

——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

(5) NAMEVALUEITERATOR_RESET

目的：将元素序列中的指针重置为第一个元素。参数：None.

Java 调用序列：nameValueIterator.reset () ;

返回：None.

可能的异常详见 13.3.18 节如下函数：

——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

13.2.21 NAMEVALUEUNITIDITERATOR

(1) NAMEVALUEUNITIDITERATOR_DESTROY

目的：销毁迭代器并释放关联的内存。

参数：None.

Java 调用序列：nameValueUnitIdIterator.destroy () ;

返回：None.

可能的异常详见 13.3.18 节如下函数：

——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

(2) NAMEVALUEUNITIDITERATOR_GETCOUNT

目的：获取迭代器可访问的元素总数。参数：None.

Java 调用序列：T_LONG nvulidCount = nameValueUnitIdIterator.getCount () ;

返回：返回名称：nameValueUnitIdCount

返回类型：T_LONG

迭代器可访问的元素数。可能的异常详见 13.3.18 节如下函数：

——AO_CONNECTION_LOST;
——AO_IMPLEMENTATION_PROBLEM;
——AO_NOT_IMPLEMENTED;
——AO_NO_MEMORY;
——AO_SESSION_NOT_ACTIVE。

(3) NAMEVALUEUNITIDITERATOR_NEXTN

目的：从序列中获取下 n 个元素。参数：how_many (类型=T_long) 请求的元素数。Java 调用序

列：NameValueUnitId[] nameValueUnitIds = nameValueUnitIdIterator.nextN (how_many);

返回：返回名称：nameValueUnitIdSeq

返回类型：NameValueSeqUnitId

名称-值-单位元组序列中的下一个 n 个名称-值-单位元组。可能的异常详见 13.3.18 节如下函数：

——AO_BAD_PARAMETER;
——AO_CONNECTION_LOST;
——AO_IMPLEMENTATION_PROBLEM;
——AO_NOT_IMPLEMENTED;
——AO_NO_MEMORY;
——AO_SESSION_NOT_ACTIVE。

(4) NAMEVALUEUNITIDITERATOR_NEXTONE

目的：从序列中获取下一个元素。

参数：None.

Java 调用序列：NameValueUnitId nameValueUnitId = nameValueUnitIdIterator.nextOne () ;

返回：返回名称：nameValueUnitId

返回类型：NameValueUnitId

name-value-unitId 元组序列中的下一个 name-value-unitId 元组。

可能的异常详见 13.3.18 节如下函数：

——AO_CONNECTION_LOST;
——AO_IMPLEMENTATION_PROBLEM;
——AO_NOT_IMPLEMENTED;
——AO_NO_MEMORY;
——AO_SESSION_NOT_ACTIVE。

(5) NAMEVALUEUNITIDITERATOR_RESET

目的：将元素序列中的指针重置为第一个元素。

参数：None.

Java 调用序列：nameValueUnitIdIterator.reset () ;

返回：None.

可能的异常详见 13.3.18 节如下函数：

- AO_CONNECTION_LOST;
- AO_IMPLEMENTATION_PROBLEM;
- AO_NOT_IMPLEMENTED;
- AO_NO_MEMORY;
- AO_SESSION_NOT_ACTIVE。

13. 2. 22 NAMEVALUEUNITITERATOR**(1) NAMEVALUEUNITITERATOR_DESTROY**

目的：销毁迭代器并释放关联的内存。

参数：None.

Java 调用序列：nameValueUnitIterator.destroy () ;

返回：None。

可能的异常详见 13.3.18 节如下函数：

- AO_CONNECTION_LOST;
- AO_IMPLEMENTATION_PROBLEM;
- AO_NOT_IMPLEMENTED;
- AO_NO_MEMORY;
- AO_SESSION_NOT_ACTIVE。

(2) NAMEVALUEUNITITERATOR_GETCOUNT

目的：获取迭代器可访问的元素总数。

参数：None.

Java 调用序列：T_LONG nvuCount = nameValueUnitIterator.getCount () ;

返回：返回名称：nvuCount

返回类型：T_LONG

迭代器可访问的元素数。可能的异常详见 13.3.18 节如下函数：

- AO_CONNECTION_LOST;
- AO_IMPLEMENTATION_PROBLEM;
- AO_NOT_IMPLEMENTED;
- AO_NO_MEMORY;

——AO_SESSION_NOT_ACTIVE。

(3) NAMEVALUEUNITITERATOR_NEXTN

目的：从序列中获取下 n 个元素。

参数：how_many（类型=T_long）

请求的元素数。Java 调用序列：NameValueUnit[] nameValueUnits = nameValueUnitIterator.nextN
（how_many）；

返回：返回名称：nameValueUnits

返回类型：NameValueUnitSequence

名称-值-单位元组序列中的下一个 n 个名称-值-单位元组。

可能的异常详见 13.3.18 节如下函数：

——AO_BAD_PARAMETER；
——AO_CONNECTION_LOST；
——AO_IMPLEMENTATION_PROBLEM；
——AO_NOT_IMPLEMENTED；
——AO_NO_MEMORY；
——AO_SESSION_NOT_ACTIVE。

(4) NAMEVALUEUNITITERATOR_NEXTONE

目的：从序列中获取下一个元素。

参数：None.

Java 调用序列：NameValueUnit nameValueUnit = nameValueUnitIterator.nextOne（）；

返回：返回名称：nameValueUnit

返回类型：NameValueUnit

名称值单位元组序列中的下一个名称值单位元组。可能的异常详见 13.3.18 节如下函数：

——AO_CONNECTION_LOST；
——AO_IMPLEMENTATION_PROBLEM；
——AO_NOT_IMPLEMENTED；
——AO_NO_MEMORY；
——AO_SESSION_NOT_ACTIVE。

(5) NAMEVALUEUNITITERATOR_RESET

目的：将元素序列中的指针重置为第一个元素。

参数：None.

Java 调用序列：nameValueUnitIterator.reset（）；

返回：None.

可能的异常详见 13.3.18 节如下函数：

——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

13.2.23 NAMEVALUEUNITSEQUENCEITERATOR

(1) NAMEVALUEUNITSEQUENCEITERATOR_DESTROY

目的：销毁迭代器并释放关联的内存。

参数：None.

Java 调用序列：nameValueUnitSequenceIterator.destroy () ;

返回：None.

可能的异常详见 13.3.18 节如下函数：

——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIV。

(2) NAMEVALUEUNITSEQUENCEITERATOR_GETCOUNT

目的：获取迭代器可访问的元素总数。

参数：None.

Java 调用序列：T_LONG nameValueUnitCount = nameValueUnitSequenceIterator.getCount () ;

返回：返回名称：nameValueUnitCount

返回类型：T_LONG

迭代器可访问的元素数。可能的异常详见 13.3.18 节如下函数：

——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

(3) NAMEVALUEUNITSEQUENCEITERATOR_NEXTN

目的：从序列中获取下 n 个元素。

参数：how_many (类型=T_long)

请求的元素数。Java 调用序列：NameValueSeqUnit[] nameValueSeqUnits =
 nameValueUnitSequenceIterator.nextN (how_many) ;

返回：返回名称：nameValueSeqUnits

返回类型：NameValueSeqUnitSequence

名称值单位序列的下一个 n 个值。对于每个名称，下一个值存储在值序列中。可能的异常详见 13.3.18 节如下函数：

——AO_BAD_PARAMETER;
 ——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

(4) NAMEVALUEUNITSEQUENCEITERATOR_NEXTONE

目的：从迭代器中获取下一个元素。

参数：None.

Java 调用序列：NameValueSeqUnit nameValueSeqUnit =
 nameValueUnitSequenceIterator.nextOne();

返回：返回名称：nameValueSeqUnit

返回类型：NameValueSeqUnit

名称-值-单位元组序列中的下一个名称-值-单位元组序列。可能的异常详见 13.3.18 节如下函数：

——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

(5) NAMEVALUEUNITSEQUENCEITERATOR_RESET

目的：将元素迭代器中的指针重置为第一个元素。

参数：None.

Java 调用序列：nameValueUnitSequenceIterator.reset();

返回：None.

可能的异常详见 13.3.18 节如下函数：

——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

13.2.24 QUERY

(1) QUERY_EXECUTEQUERY

目的：以异步模式执行查询。

参数：params（类型=名称值序列）

- 参数名称和值的序列。应传递以下参数：Name: "QueryResultType";
Type: ResultType.
Comment：指定客户端预期的结果类型，如果在接口 queryevaluator 的方法 PrepareQuery 或方法 CreateQuery 中没有给定参数，则需要此参数。Default value: INSTELEM_ITERATOR_AS_RESULT
- Name: "Synchronous";
Type: T_BOOLEAN
Comment：如果为“真”，则保证同步执行。Default value: “假”

Java 调用序列：query.executeQuery (params) ;

返回：None.

可能的异常详见 13.3.18 节如下函数：

——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。

(2) QUERY_GETINSTANCES

目的：获取查询结果。只有在执行查询后才应调用此方法。它返回查询找到的实例的迭代器。

参数：None.

Java 调用序列：InstanceElementIterator result = query.getInstance () ;

返回：返回名称：result

返回类型：InstanceElementIterator

查询的结果作为实例迭代。

可能的异常详见 13.3.18 节如下函数：

——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。

(3) QUERY_GETQUERYEVALUATOR

目的：获取负责此查询的 QueryEvaluator 对象。

参数：None。

Java 调用序列：QueryEvaluator queryEvl = query.getQueryEvaluator () ;

返回：返回名称：queryEvl

返回类型：QueryEvaluator

负责查询此 QueryEvaluator 的对象。可能的异常详见 13.3.18 节如下函数：

- AO_BAD_PARAMETER;
- AO_CONNECTION_LOST;
- AO_IMPLEMENTATION_PROBLEM;
- AO_NOT_IMPLEMENTED;
- AO_NO_MEMORY;
- AO_SESSION_NOT_ACTIVE。

(4) QUERY_GETSTATUS

目的：返回查询状态。

如果查询仍在执行，则返回“INCOMPLETE”。如果查询已执行完毕，或者查询执行因错误或超时而停止，则返回 complete。参数：None。

Java 调用序列：QueryStatus status = query.getStatus () ;

返回：返回名称：status

返回类型：QueryStatus

The status of the query.

可能的异常详见 13.3.18 节如下函数：

- AO_BAD_PARAMETER;
- AO_CONNECTION_LOST;
- AO_IMPLEMENTATION_PROBLEM;
- AO_NOT_IMPLEMENTED;
- AO_NO_MEMORY;
- AO_SESSION_NOT_ACTIVE。

(5) QUERY_GETTABLE

目的：获取查询结果。只有在执行查询后才应调用此方法。它以结构形式返回结果。

参数: None.

Java 调用序列: `NameValueSeqUnit[] result = query.getTable ()` ;

返回: 返回名称: `result`

返回类型: `NameValueSeqUnitSequence`

作为名称值序列单位序列的查询结果。每个名称值序列单元元组是表中的一列。名称值序列单元序列

是桌子。结果结构可能非常巨大。可能的异常详见 13.3.18 节如下函数:

- `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`。

(6) `QUERY_GETTABLEROWS`

目的: 获取查询结果。只有在执行查询后才应调用此方法。它返回名称值单位序列的迭代器。

参数: None.

Java 调用序列: `NameValueUnitSequenceIterator result = query.getTableRows ()` ;

返回: 返回名称: `result`

返回类型: `NameValueUnitSequenceIterator`

作为名称值单位序列迭代器的查询结果。每个名称值单位元组是表中的一个单元格。名称值单位序列是表的一行。可能的异常详见 13.3.18 节如下函数:

- `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`。

(7) QUERY_PREPAREQUERY

目的: 客户机是否可以省略查询预处理(优化等)调用? 在这种情况下, 功能是在调用 `executeQuery` 时执行的。

参数: `params` (类型=名称值序列)

- 参数名和值的序列。应传递以下参数: Name: "QueryResultType";

Type: `ResultType`.

Comment: 指定客户端预期的结果类型, 如果在接口 `queryevaluator` 的方法 `createQuery` 中未给定参数, 则需要此参数。Default value :

INSTELEM_ITERATOR_AS_RESULT

Java 调用序列: `query.prepareQuery (params)` ;

返回: `None`.

可能的异常详见 13.3.18 节如下函数:

- `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_RESULTTYP`.

13. 2. 25 QUERYEVALUATOR

(1) QUERYEVALUATOR_CREATEQUERY

目的: 创建一个查询对象以异步模式执行它。

参数: `queryStr` (类型=`T_字符串`)

查询字符串 `params` (类型=名称值序列)

- 参数名和值的序列。应传递以下参数: Name: “查询结果类型”;

Type: 结果类型。

Comment : 指定客户端预期的结果类型。Default value :
INSTELEM_ITERATOR_AS_RESULT

- Name: "MaxDuration";

Type: `T_LONG`

Comment: 可用于限制处理时间。时间以毫秒为单位, Default value: 0 (无限制)

Java 调用序列: `Query queryObj = queryEvaluator.createQuery (queryStr,params)` ;

返回: 返回名称: `queryObj`

返回类型: `Query`

The query object.

可能的异常详见 13.3.18 节如下函数：

——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。

(2) QUERYEVALUATOR_GETINSTANCES

目的：以同步模式计算查询。参数：*queryStr*（类型=*T* 字符串）

查询字符串。*params*（类型=名称值序列）

- 参数名和值的序列。应传递以下参数：Name: "MaxDuration";

Type: T_LONG

Comment: 可用于限制处理时间。时间以毫秒为单位，Default value: 0（无限制）Java

调用序列：InstanceElementIterator result = queryEvaluator.getInstance
 (queryStr,params);

返回：返回名称：result

返回类型：InstanceElementIterator

作为实例迭代器的查询结果。可能的异常详见 13.3.18 节如下函数：

——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。

(3) QUERYEVALUATOR_GETTABLE

目的：以同步模式计算查询。

参数：*queryStr*（类型=*T* 字符串）

查询字符串。*params*（类型=名称值序列）

- 参数名和值的序列。应传递以下参数：Name: "MaxDuration";

Type: T_LONG

Comment: 可用于限制处理时间。时间以毫秒为单位, Default value: 0 (无限制) Java

调用序列: NameValueSeqUnit[] result = queryEvaluator.getTable (queryStr,params);

返回: 返回名称: result

返回类型: NameValueSeqUnitSequence

作为名称值序列单位序列的查询结果。每个名称值 **sequecne** 单元元组是表中的一列。名称值序列

单元序列是表。结果结构可能非常巨大。可能的异常详见 13.3.18 节如下函数:

——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) QUERYEVALUATOR_GETTABLEROWS

目的: 以同步模式计算查询。

参数: queryStr (类型=T_字符串)

查询字符串。params (类型=名称值序列)

— 参数名和值的序列。应传递以下参数: Name: "MaxDuration";

Type: T_LONG

Comment: 可用于限制处理时间。时间以毫秒为单位, Default value: 0 (无限制)

Java 调用序列: NameValueUnitSequenceIterator result =queryEvaluator.getTableRows
 (queryStr,params);

返回: 返回名称: result

返回类型: NameValueUnitSequenceIterator

作为名称值单位序列迭代器的查询结果。每个名称值单位元组是表中的一个单元格。名称值单位序

列是表的一行。可能的异常详见 13.3.18 节如下函数:

——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。

13.2.26 SMATLINK

(1) SMATLINK_GETLINKTYPE

目的：获取链接或生成类型。

参数：None.

Java 调用序列：BuildUpFunction linkType = sMatLink.getLinkType () ;

返回：返回名称：linkType

返回类型：BuildUpFunction

链接类型。可能的异常详见 13.3.18 节如下函数：

——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

(2) SMATLINK_GETORDINALNUMBER

目的：获取序号或序列号

参数：None.

Java 调用序列：T_LONG ordinalNumber = sMatLink.getOrdinalNumber () ;

返回：返回名称：ordinalNumber

返回类型：T_LONG

序列号。可能的异常详见 13.3.18 节如下函数：

——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

(3) SMATLINK_GETSMAT1

目的：获取链接的第一个子矩阵。

参数：None.

Java 调用序列：SubMatrix subMatrix = sMatLink.getSMat1 () ;

返回：返回名称：subMatrix

返回类型: SubMatrix

链接的第一个子矩阵。可能的异常详见 13.3.18 节如下函数:

——AO_CONNECTION_LOST;
——AO_IMPLEMENTATION_PROBLEM;
——AO_NOT_IMPLEMENTED;
——AO_NO_MEMORY;
——AO_SESSION_NOT_ACTIVE。

(4) SMATLINK_GETSMAT1COLUMNS

目的: 获取链接中使用的第一个子矩阵的绑定列 (例如时间)。

参数: None.

Java 调用序列: Column[] columns = sMatLink.getSMat1Columns ();

返回: 返回名称: columns

返回类型: ColumnSequence

第一个子矩阵的列。可能的异常详见 13.3.18 节如下函数:

——AO_CONNECTION_LOST;
——AO_IMPLEMENTATION_PROBLEM;
——AO_NOT_IMPLEMENTED;
——AO_NO_MEMORY;
——AO_SESSION_NOT_ACTIVE。

(5) SMATLINK_GETSMAT2

目的: 获取链接的第二个子矩阵。

参数: None.

Java 调用序列: SubMatrix subMatrix = sMatLink.getSMat2 ();

返回: 返回名称: subMatrix

返回类型: SubMatrix

链接的第二个子矩阵。可能的异常详见 13.3.18 节如下函数:

——AO_CONNECTION_LOST;
——AO_IMPLEMENTATION_PROBLEM;
——AO_NOT_IMPLEMENTED;
——AO_NO_MEMORY;
——AO_SESSION_NOT_ACTIVE。

(6) SMATLINK_GETSMAT2COLUMNS

目的: 获取链接中使用的第二个子矩阵的绑定列 (例如时间)。参数: None.

Java 调用序列: Column[] columns = sMatLink.getSMat2Columns ();

返回：返回名称：columns

返回类型：ColumnSequence

第二个子矩阵的列。可能的异常详见 13.3.18 节如下函数：

——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

(7) SMATLINK_SETLINKTYPE

目的：设置生成或链接类型。

参数：linkType (Type = BuildUpFunction)

请求的组建函数。Java 调用序列：sMatLink.setLinkType (linkType) ;

返回：None.

可能的异常详见 13.3.18 节如下函数：

——AO_BAD_PARAMETER;
 ——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_INVALID_BUILDUP_FUNCTION;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

(8) SMATLINK_SETORDINALNUMBER

目的：设置序号或序列号。参数：ordinalNumber (类型=T_long)

序列号。Java 调用序列：sMatLink.setOrdinalNumber (ordinalNumber) ;

返回：None.

可能的异常详见 13.3.18 节如下函数：

——AO_BAD_PARAMETER;
 ——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_INVALID_ORDINALNUMBER;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

(9) SMATLINK_SETSMAT1

目的：设置链接的第一个子矩阵。参数：subMat1（Type = SubMatrix）

子矩阵链接的第一个子矩阵。Java 调用序列：sMatLink.setSMat1（subMatrix）；

返回：None.

可能的异常详见 13.3.18 节如下函数：

- AO_BAD_PARAMETER;
- AO_CONNECTION_LOST;
- AO_IMPLEMENTATION_PROBLEM;
- AO_INVALID_SUBMATRIX;
- AO_NOT_IMPLEMENTED;
- AO_NO_MEMORY;
- AO_SESSION_NOT_ACTIVE。

（10）SMATLINK_SETSMAT1COLUMNS

目的：设置链接中使用的第一个子矩阵的绑定列（例如时间）。

参数：columns（类型=列序列）

子矩阵列的序列。Java 调用序列：sMatLink.setSMat1Columns（columns）；

返回：None.

可能的异常详见 13.3.18 节如下函数：

- AO_BAD_PARAMETER;
- AO_CONNECTION_LOST;
- AO_IMPLEMENTATION_PROBLEM;
- AO_INVALID_COLUMN;
- AO_NOT_IMPLEMENTED;
- AO_NO_MEMORY;
- AO_SESSION_NOT_ACTIVE。

（11）SMATLINK_SETSMAT2

目的：设置链接的第二个子矩阵。

参数：subMat2（类型=子矩阵）

子矩阵链接的第二个子矩阵。Java 调用序列：sMatLink.setSMat2（subMatrix）；

返回：None.

可能的异常详见 13.3.18 节如下函数：

- AO_BAD_PARAMETER;
- AO_CONNECTION_LOST;
- AO_IMPLEMENTATION_PROBLEM;
- AO_INVALID_SUBMATRIX;
- AO_NO_MEMORY;

——AO_NOT_IMPLEMENTED;
 ——AO_SESSION_NOT_ACTIVE。

(12) SMATLINK_SETSMAT2COLUMNS

目的：设置链接中使用的第二个子矩阵的绑定列（例如时间）。如果有多个列绑定，则列序列必须与第一个子矩阵的列序列相同。参数：columns（类型=列序列）

子矩阵的列序列。Java 调用序列：sMatLink.setSMat2Columns（columns）；

返回：None。

可能的异常详见 13.3.18 节如下函数：

——AO_BAD_PARAMETER;
 ——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_INVALID_COLUMN;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

13.2.27 SUBMATRIX

SubMatrix 接口扩展了 InstanceElement 接口，因此 InstanceElement 接口的所有方法也适用于它。

(1) SUBMATRIX_GETCOLUMNS

目的：获取子矩阵的列。该列不是从 InstanceElement 接口继承的。这是获取列的唯一方法。在 smatlink 接口中使用列来构建值矩阵。模式区分大小写，可以包含通配符。

参数：colPattern（类型=模式）

列名称的名称或搜索模式。Java 调用序列：Column[] columns = subMatrix.getCOLUMNS（colPattern）；

返回：返回名称：columns

返回类型：ColumnSequence

子矩阵的列。可能的异常详见 13.3.18 节如下函数：

——AO_BAD_PARAMETER;
 ——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

(2) SUBMATRIX_GETVALUEMATRIX

目的：获取子矩阵的值矩阵。

通过调用此方法，返回值矩阵的值矩阵模式将取自上下文变量“value matrix_mode”。有关值矩阵不同模式的详细信息，请参阅此接口的 `getValueMatrixinMode()` 方法中的说明。参数：
None.

Java 调用序列：ValueMatrix valueMatrix = subMatrix.getValueMatrix ();

返回：返回名称：valueMatrix

返回类型：ValueMatrix

价值矩阵。可能的异常详见 13.3.18 节如下函数：

——AO_CONNECTION_LOST;
——AO_IMPLEMENTATION_PROBLEM;
——AO_INVALID_SMATLINK;
——AO_NOT_IMPLEMENTED;
——AO_NO_MEMORY;
——AO_SESSION_NOT_ACTIVE。

(3) SUBMATRIX_GETVALUEMATRIXINMODE

目的：以请求的模式获取子矩阵的值矩阵。

服务器不检查任何独立列。值矩阵的局部列正是该子矩阵的局部列。TS_ValueSeq 中的标志是本地列的标志。如果没有给定标志，则物理存储中没有可用的标志，客户机必须考虑全局标志。如果请求的模式为“CALCULATED”，服务器将返回隐式和原始本地列的计算值。隐式局部列将使用生成参数扩展为显式值序列。存储原始值的本地列将使用原始值和生成参数重新计算。如果请求的模式是“STORAGE”，服务器将返回本地列的值，如同它们在物理存储中一样。只有当值矩阵模式为“STORAGE”时，才允许将值写入值矩阵。参数：
vmMode (类型=值矩阵模式)

值矩阵的请求模式。Java 调用序列：ValueMatrix vm = subMatrix.getValueMatrixInMode (vmMode);

返回：返回名称：vm

返回类型：ValueMatrix

The value matrix.

可能的异常详见 13.3.18 节如下函数：

——AO_CONNECTION_LOST;
——AO_IMPLEMENTATION_PROBLEM;
——AO_NOT_IMPLEMENTED;
——AO_NO_MEMORY;
——AO_SESSION_NOT_ACTIVE。

(4) SUBMATRIX_LISTCOLUMNS

目的：获取子矩阵列的名称。名称序列与相关实例的名称相同。模式区分大小写，可以包含通配符。

参数: colPattern (类型=模式)

名称或搜索模式获取列的名称 Java 调用序列: `Name[] columnNames = subMatrix.listColumns(colPattern);`

返回: 返回名称: columnNames

返回类型: NameSequence

子矩阵的列名。可能的异常详见 13.3.18 节如下函数:

——AO_BAD_PARAMETER;
 ——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

13.2.28 VALUEMATRIX

(1) VALUEMATRIX_ADDCOLUMN

目的: 在值矩阵中添加一列。只有从子矩阵创建值矩阵时,才允许创建值向量。允许在事务外添加列,但建议开始事务。数据在事务提交时是不变的。在提交事务之前或在访问值矩阵的测量点之前,允许在不同的值向量中具有不同数量的值。添加新列后,可以设置列的值。

参数: newColumn (类型=名称单位)

要添加的列的名称和单位。Java 调用序列: `valueMatrix.addColumn(newColumn);`

返回: 返回名称: column

返回类型: Column

The new column.

可能的异常 (详见 13.3.18 节):

——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。

(2) VALUEMATRIX_ADDCOLUMNSCALEDDBY

目的: 在值矩阵中添加一列。只有从子矩阵创建值矩阵时,才允许创建值向量。列将按给定的缩放

列进行缩放。

允许在事务外添加列，但建议开始事务。数据在事务提交时是不变的。在提交事务之前或在访问值矩阵的测量点之前，允许在不同的值向量中具有不同数量的值。添加新列后，可以设置列的值。参数：newColumn（类型=名称单位）

要添加的列的名称和单位。scalingColumn（类型=名称单位）

缩放列。Java 调用序列：Column column = valueMatrix.addColumnScaledBy
(newColumn,scalingColumn);

返回：返回名称：column

返回类型：Column

缩放列。可能的异常详见 13.3.18 节如下函数：

——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。

(3) VALUEMATRIX_DESTROY

目的：销毁服务器上的对象。此方法用于告诉服务器客户端不再使用此对象。销毁方法后访问此对象将导致异常。

参数：None.

Java 调用序列：vmObj.destroy ()

返回：None.

可能的异常详见 13.3.18 节如下函数：

——AO_BAD_PARAMETER;
——AO_CONNECTION_LOST;
——AO_IMPLEMENTATION_PROBLEM;
——AO_NOT_IMPLEMENTED;
——AO_NO_MEMORY;
——AO_SESSION_NOT_ACTIVE。

(4) VALUEMATRIX_GETCOLUMNCOUNT

目的：获取值矩阵的列计数。

参数：None.

Java 调用序列：T_LONG columnCount = valueMatrix.getColumnCount () ;

返回：返回名称：columnCount

返回类型：T_LONG

值矩阵的列数。可能的异常详见 13.3.18 节如下函数：

——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

(5) VALUEMATRIX_GETCOLUMNS

目的：获取值矩阵的列，无论该列是依赖列还是独立列。模式区分大小写，可以包含通配符。

参数：colPattern（类型=模式）

名称或搜索模式获取列的名称

Java 调用序列：Column[] columns = valueMatrix.getColumns (colPattern) ;

返回：返回名称：columns

返回类型：ColumnSequence

可能的异常详见 13.3.18 节如下函数：

——AO_BAD_PARAMETER;
 ——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

(6) VALUEMATRIX_GETCOLUMNSSCALED BY

目的：获取按给定列缩放的列。参数：scalingColumn（Type = Column）

缩放列。Java 调用序列：Column[] columns = valueMatrix.getColumnsScaledBy (scalingColumn);

返回：返回名称：columns

返回类型：ColumnSequence

按给定输入列缩放的列。可能的异常详见 13.3.18 节如下函数：

——AO_BAD_PARAMETER;
 ——AO_CONNECTION_LOST;
 ——AO_IMPLEMENTATION_PROBLEM;

——AO_NOT_IMPLEMENTED;
——AO_NO_MEMORY;
——AO_SESSION_NOT_ACTIVE;
——AO_NO_SCALING_COLUMN。

(7) VALUEMATRIX_GETINDEPENDENTCOLUMNS

目的：获取值矩阵的独立列。独立列是用于构建值矩阵的列。

参数：colPattern（类型=模式）

独立列名的名称或搜索模式。Java 调用序列：`Column[] columns = valueMatrix.getIndependentColumns (colPattern) ;`

返回：返回名称：columns

返回类型：ColumnSequence

值矩阵的独立列。

可能的异常详见 13.3.18 节如下函数：

——AO_BAD_PARAMETER;
——AO_CONNECTION_LOST;
——AO_IMPLEMENTATION_PROBLEM;
——AO_NOT_IMPLEMENTED;
——AO_NO_MEMORY;
——AO_SESSION_NOT_ACTIVE。

(8) VALUEMATRIX_GETMODE

目的：获取值矩阵的当前模式。

参数：None.

Java 调用序列：`ValueMatrixMode vmMode = valueMatrix.getMode () ;`

返回：返回名称：vmMode

返回类型：ValueMatrixMode

值矩阵的模式。可能的异常详见 13.3.18 节如下函数：

——AO_CONNECTION_LOST;
——AO_IMPLEMENTATION_PROBLEM;
——AO_NOT_IMPLEMENTED;
——AO_NO_MEMORY;
——AO_SESSION_NOT_ACTIVE。

(9) VALUEMATRIX_GETROWCOUNT

目的：获取值矩阵的行计数。

参数：None.

Java 调用序列: `T_LONG rowCount = valueMatrix.getRowCount ( ) ;`

返回: 返回名称: `rowCount`

返回类型: `T_LONG`

值矩阵的行数。可能的异常详见 13.3.18 节如下函数:

——`AO_CONNECTION_LOST`;
 ——`AO_IMPLEMENTATION_PROBLEM`;
 ——`AO_NOT_IMPLEMENTED`;
 ——`AO_NO_MEMORY`;
 ——`AO_SESSION_NOT_ACTIVE`。

(10) VALUEMATRIX_GETSCALINGCOLUMNS

目的: 获取值矩阵的缩放列。

参数: `colPattern` (类型=模式)

缩放列的名称或搜索模式。

Java 调用序列: `Column[] columns = valueMatrix.getScalingColumns (colPattern) ;`

返回: 返回名称: `columns`

返回类型: `ColumnSequence`

值矩阵的缩放列。可能的异常详见 13.3.18 节如下函数:

——`AO_BAD_PARAMETER`;
 ——`AO_CONNECTION_LOST`;
 ——`AO_IMPLEMENTATION_PROBLEM`;
 ——`AO_NOT_IMPLEMENTED`;
 ——`AO_NO_MEMORY`;
 ——`AO_SESSION_NOT_ACTIVE`。

(11) VALUEMATRIX_GETVALUE

目的: 获取值矩阵的不同列的值。

服务器行为取决于值矩阵的模式。值矩阵模式 “`CALCULATED`”: 如果相应的本地列的 “`sequence_representation`” 是中心 “`raw_linear`”、“`raw_polynomial`”、“`raw_linear_external`”、“`raw_polynomial_external`” 之一,

“`raw-linear-calibrated`” 或 “`raw-linear-calibrated-external`”, 服务器将首先从原始值和生成参数计算物理值, 然后将它们返回到请求客户机。值矩阵模式 “`STORAGE`”: 如果相应的局部列的 “`sequence_representation`” 是 “`raw_linear`”、“`raw_polynomial`”、“`raw_linear_external`”、“`raw_polynomial_external`”,

“`raw-linear-calibrated`” 或 “`raw-linear-calibrated-external`”, 服务器将返回本地列的 `rawvalues`。

参数: `columns` (类型=列序列)

请求的列。*startPoint* (类型=*T_long*)

The starting point in the column.

count (类型=*T_long*)

要检索的点数。0 表示直到列结束。

Java 调用序列: `Column columns[] = vmObj.getColumns ("*");`

`T LONG startPoint = 0;`

`T LONG count = 100;`

`NameValueSeqUnit values[] = vmObj.getValue (columns, startPoint, count);`

返回: 返回名称: `values`

返回类型: `NameValueSeqUnitSequence`

不同列的值。返回结构的名称与列的名称。单位与列的单位相对应。秩序结果可能与请求序列中的顺序不匹配。可能的异常详见 13.3.18 节如下函数:

——`AO_BAD_PARAMETER;`
 ——`AO_CONNECTION_LOST;`
 ——`AO_IMPLEMENTATION_PROBLEM;`
 ——`AO_NOT_IMPLEMENTED;`
 ——`AO_NO_MEMORY;`
 ——`AO_SESSION_NOT_ACTIVE.`

(12) VALUEMATRIX_GETVALUEMEAPPOINT

目的: 获取值矩阵的测量点。参数 *meapoint* 指定矩阵的行。迭代器允许访问行中的所有元素。

服务器行为取决于值矩阵的模式。值矩阵模式 ‘**CALCULATED**’: 如果相应局部列的 “*sequence_representation*” 是 “*raw_linear*”、“*raw_polynomial*”、“*raw_linear_external*”、“*raw_polynomial_external*” 条目之一,

“*raw-linear-calibrated*” 或 “*raw-linear-calibrated-external*”, 服务器将首先从原始值和生成参数计算物理值, 然后将它们返回到请求客户机。值矩阵模式 “**STORAGE**”: 如果相应的局部列的 “*sequence_representation*” 是 “*raw_linear*”、“*raw_polynomial*”、“*raw_linear_external*”、“*raw_polynomial_external*”,

“*raw-linear-calibrated*” 或 “*raw-linear-calibrated-external*”, 服务器将返回 *rawvalues*。参数:

meaPoint (类型=*T_long*)

测量点。Java 调用序列: `NameValueUnitIterator valueMeaPoint = valueMatrix.getValueMeaPoint (meaPoint);`

返回: 返回名称: `valueMeaPoint`

返回类型: `NameValueUnitIterator`

请求的测量点。可能的异常详见 13.3.18 节如下函数:

——`AO_BAD_PARAMETER;`
 ——`AO_CONNECTION_LOST;`

——AO_IMPLEMENTATION_PROBLEM;
 ——AO_NOT_IMPLEMENTED;
 ——AO_NO_MEMORY;
 ——AO_SESSION_NOT_ACTIVE。

(13) VALUEMATRIX_GETVALUEVECTOR

目的：从值矩阵中获取列的值或部分值。参数列指定从哪个列返回值。**startpoint** 和 **pointcount** 指定向量的一部分。**startpoint=0** 和 **pointcount=rowcount** 将返回整个向量。当 **startpoint>=rowcount** 时，将引发异常。如果 **startpoint+pointcount>rowcount**，则只返回向量的其余值，不会引发异常。将 **InterfaceColumn** 的 **GetName** 和 **GetUnit** 方法用于列的名称和单位。名称和价值不是存储在向量的每个元素上的。返回类型 **ts_valueseq** 不是 **s_value** 的序列，而是一个特殊的结构。

服务器行为取决于值矩阵的模式。

值矩阵模式 ‘CALCULATED’：如果相应的局部列的 “sequence_representation” 是 “raw_linear”、
 “raw_polynomial”、“raw_linear_external”、“raw_polynomial_external”，
 “raw-linear-calibrated” 或 “raw-linear-calibrated-external”，服务器将首先从原始值和生成参数计算物理值，然后将它们返回到请求客户机。值矩阵模式 “STORAGE”：如果相应的局部列的 “sequence_representation” 是 “raw_linear”、“raw_polynomial”、
 “raw_linear_external”、“raw_polynomial_external”，“raw-linear-calibrated” 或
 “raw-linear-calibrated-external”，服务器将返回本地列的 **rawvalues**。参数：**col**（类型=列）

从中检索值的列。**startPoint**（类型=**T_long**）

列中的起始点。**count**（类型=**T_long**）

要检索的点数。Java 调用序列：**TS_Value[] valueVector = valueMatrix.getValueVector**
(col,startPoint,count);

返回：返回名称：**valueVector**

返回类型：**TS_ValueSeq**

值矩阵的请求列值。可能的异常详见 13.3.18 节如下函数：

——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。

(14) VALUEMATRIX_LISTCOLUMNS

目的：获取值矩阵各列的名称，无论该列是依赖列还是独立列。该模式区分大小写，可能包含通配符。

参数：colPattern（类型=模式）

名称或搜索模式获取列的名称

Java 调用序列：Name[] columnNames = valueMatrix.listColumns（colPattern）；

返回：返回名称：columnNames

返回类型：NameSequence

值矩阵的列名称，无论该列是依赖列、独立列还是由另一列缩放。可能的异常详见 13.3.18 节如下函数：

- AO_BAD_PARAMETER;
- AO_CONNECTION_LOST;
- AO_IMPLEMENTATION_PROBLEM;
- AO_NOT_IMPLEMENTED;
- AO_NO_MEMORY;
- AO_SESSION_NOT_ACTIVE。

(15) VALUEMATRIX_LISTCOLUMNSSCALED BY

目的：列出按给定列缩放的列的名称。

参数：scalingColumn（类型=列）

缩放列。Java 调用序列：Name[] columnNames = valueMatrix.listColumnsScaledBy（scalingColumn）；

返回：返回名称：columnNames

返回类型：NameSequence

按给定输入列缩放的列的名称。可能的异常详见 13.3.18 节如下函数：

- AO_BAD_PARAMETER;
- AO_CONNECTION_LOST;
- AO_IMPLEMENTATION_PROBLEM;
- AO_NOT_IMPLEMENTED;
- AO_NO_MEMORY;
- AO_SESSION_NOT_ACTIVE;
- AO_NO_SCALING_COLUMN。

(16) VALUEMATRIX_LISTINDEPENDENTCOLUMNS

目的：获取值矩阵独立列的名称。独立列是用于构建值矩阵的列。

参数：colPattern（Type = Pattern）

名称或搜索模式获取列的名称

Java 调用序列: `Name[] columnNames = valueMatrix.listIndependentColumns (colPattern) ;`

返回: 返回名称: `columnNames`

返回类型: `NameSequence`

值矩阵独立列的名称。可能的异常详见 13.3.18 节如下函数:

——`AO_BAD_PARAMETER;`
 ——`AO_CONNECTION_LOST;`
 ——`AO_IMPLEMENTATION_PROBLEM;`
 ——`AO_NOT_IMPLEMENTED;`
 ——`AO_NO_MEMORY;`
 ——`AO_SESSION_NOT_ACTIVE。`

(17) VALUEMATRIX_LISTSCALINGCOLUMNS

目的: 获取值矩阵的缩放列的名称。

参数: `colPattern (Type = Pattern)`

名称或搜索模式获取列的名称

Java 调用序列: `Name[] columnNames = valueMatrix.listScalingColumns (colPattern) ;`

返回: 返回名称: `columnNames`

返回类型: `NameSequence`

值矩阵的缩放列的名称。可能的异常详见 13.3.18 节如下函数:

——`AO_BAD_PARAMETER;`
 ——`AO_CONNECTION_LOST;`
 ——`AO_IMPLEMENTATION_PROBLEM;`
 ——`AO_NOT_IMPLEMENTED;`
 ——`AO_NO_MEMORY;`
 ——`AO_SESSION_NOT_ACTIVE。`

(18) VALUEMATRIX_REMOVEVALUEMEAPPOINT

目的: 删除给定测量点处的列值。删除给定列的点数。如果计数为 0, 则删除列结尾之前的所有点。

参数: `columnNames (类型=名称序列)`

要从中删除测量点的列。 `meaPoint (类型=T_long)`

要删除的测量点。 `count (类型=T_long)`

要从每列中删除的点数。 Java 调用序列: `valueMatrix.removeValueMeaPoint (columnNames,meaPoint,count) ;`

返回: `None.`

可能的异常详见 13.3.18 节如下函数:

——`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。

(19) VALUEMATRIX_REMOVEVALUEVECTOR

目的：从值向量中移除值。从开始点开始，计数中指定的值数将被删除。如果 **count** 为 0，则从起点到向量结束的所有值都将被删除。

参数：**col**（类型=列）

要从中删除值的列。**startPoint**（类型=T_long）

值移除的起点。**count**（类型=T_long）

要从列中删除的点数。Java 调用序列：**valueMatrix.removeValueVector (col,startPoint,count)**；

返回：None。

可能的异常详见 13.3.18 节如下函数：

——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;

(20) VALUEMATRIX_SETVALUE

目的：在值矩阵中创建或修改多个值向量。

允许在事务外部修改值矩阵的值，但建议激活一个事务。数据在事务提交时是不变的。在提交事务之前或在访问值矩阵的测量点之前，允许在不同的值向量中具有不同数量的值。参数值的名称是列的名称。这些值是列的新值。（**setValueMeaPoint** 只允许一个点，**TS_ValueSeq** 允许多个点）有一个名称-值对序列（**setValueVector** 只允许一列），因此可以修改一个值块。当客户机基于 **AOMeasurement** 创建 **ValueMatrix** 时，此方法的行为如下：服务器检查子矩阵并创建所有必要的 **AOSubMatrix**、**AOLocalColumn** 和 **AOMeasurementQuantity** 实例。**osubmatrix** 的 **name** 属性值必须由服务器生成。该值将等于属性 ID 的值（转换为 **DT_STRING**）。服务器也将创建缺少的 **AOMeasurementQuantity** 实例。强制属性将获得以下

默认值：客户提供的名称从 `AoQuantity.default_datatype` 复制的 `Datatype` 从 `AoQuantity.default_typesize` 复制的 `TypesizeInterpolation` 无插值从 `AoQuantity.default_rank` 复制的 `Rank` 从 `AoQuantity.default_dimension` 复制的 `Dimension` 服务器获取通道的值（由客户机提供），并使用属性 `default_meq_name` 在 `AoQuantity` 中查找相应的实例。接口 `ValueMatrix` 处理值，但不处理原始值。若要修改原始值，必须创建本地列的实例，在修改值之前必须创建基本属性“`sequence_representation`”、“`raw_datatype`”和“`generation_parameters`”。值的数据类型必须与“原始数据类型”的值相同；任何其他数据类型都将导致异常——`AO_INVALID_DATATYPE`；。参数：`set`（类型=`settype`）

它可能是集合类型之一 **INSERT**：插入起始点的值，当前起始点的可用值将移动到新插入值的末尾。**APPEND**：忽略起始点的值，这些值将附加在当前值的末尾。**UPDATE**：更新或修改起始点的值，当前值为改写。如果值的数目大于矢量中的值的数目，测量点将自动附加。**REMOVE**：从起始点开始，删除每列中的值数。列的名称作为名称 `ValuesUnit` 的名称提供，要移除的值的数目以值的值的数目提供。`startPoint`（类型=`T_LONG`）

测量点。值（类型 = `NameValueSeqUnitSequence`）

要插入的值。

Java 调用序列：`valueMatrix.setValue (set,startPoint,value)`；

返回：`None`。

可能的异常详见 13.3.18 节如下函数：

——`AO_BAD_PARAMETER`；
 ——`AO_CONNECTION_LOST`；
 ——`AO_IMPLEMENTATION_PROBLEM`；
 ——`AO_INVALID_COLUMN`；
 ——`AO_INVALID_DATATYPE`；
 ——`AO_INVALID_SET_TYPE`；
 ——`AO_IS_MEASUREMENT_MATRIX`；
 ——`AO_NOT_IMPLEMENTED`；
 ——`AO_NO_MEMORY`；
 ——`AO_SESSION_NOT_ACTIVE`。

(21) VALUEMATRIX_SETVALUEMEAPPOINT

目的：创建或修改值矩阵的测量点。名称值序列用新值指定列的名称列的名称必须存在。

- 当客户机基于 `AOMeasurement` 创建 `ValueMatrix` 时，此方法的行为如下：服务器检查子矩阵，并创建 `AOSubMatrix`、`AOLocalColumn` 和 `AOMeasurementQuantity` 的所有必要实例。`AOSubMatrix` 的 `name` 属性值必须由服务器生成。该值将等于属性 `id` 的值（转换为 `dt_` 字符串）。服务器也将创建缺少的 `AOMeasurementQuantity` 实例。强制属性将获得以下默认值：客户提供的名称从 `AoQuantity.default_datatype` 复制的 `Datatype` 从 `AoQuantity.default_typesize` 复

制的 Typesize

- 插 值 无 插 值 从 AoQuantity.default_rank 复 制 的 Rank 从
AoQuantity.default_dimension 复制的 Dimension

服务器获取通道的值（由客户机提供），并使用属性 default_meq_name 在 AoQuantity 中查找相应的实例。参数：set（类型=SetType）

它可能是集合类型之一

INSERT: 在 Meapoint 处插入值，将 Meapoint 处的当前值移动到新插入值的末尾。APPEND: 忽略 meaPoint 的值，这些值将附加在当前值的末尾。UPDATE: 更新或修改 meaPoint 处的值，当前值将被覆盖。如果 meaPoint 大于向量中的值的数目，则会自动附加测量点。meaPoint (Type = T_LONG)

测量点。value（类型 = NameValueSequence）

要插入的值。

Java 调用序列：valueMatrix.setValueMeaPoint (set,meaPoint,value) ;

返回: None.

可能的异常详见 13.3.18 节如下函数:

- AO_BAD_PARAMETER;
- AO_CONNECTION_LOST;
- AO_IMPLEMENTATION_PROBLEM;
- AO_IS_MEASUREMENT_MATRIX;
- AO_NOT_IMPLEMENTED;
- AO_NO_MEMORY;
- AO_SESSION_NOT_ACTIVE.

(22) VALUEMATRIX_SETVALUEVECTOR

目的：在值矩阵中创建或修改值向量。

- 允许在事务外部修改值矩阵的值，但建议激活事务。数据在事务提交时是永久的。在提交事务或访问值矩阵的测量点之前，允许在不同的值向量中具有不同数量的值。当客户机基于 AOMeasurement 创建 ValueMatrix 时，此方法的行为如下：服务器检查子矩阵，并创建 AOSubMatrix、AOLocalColumn 和 AOMeasurementQuantity 的所有必要实例。AOSubMatrix 的 name 属性值必须由服务器生成。该值将等于属性 ID 的值（转换为 dt_字符串）。服务器也将创建缺少的 AOMeasurementQuantity 实例。强制属性将获得以下默认值：客户提供的名称
- 从 AoQuantity.default_datatype 复制的 Datatype 从 AoQuantity.default_typesize 复制的 TypesizeInterpolation 无插值从 AoQuantity.default_rank 复制的 Rank

从 AoQuantity.default_dimension 复制的 Dimension 服务器获取通道的值（由客户机提供），并使用属性 default_meq_name 在 aoquantity 中查找相应的实例。参数：col（类型=列）
要设置其值的列。

set（类型=**setype**）它可能是集合类型。**INSERT**：插入起始点的值，当前起始点的可用值将移动到新插入值的末尾。**APPEND**：忽略起始点的值，这些值将附加在当前值的末尾。**UPDATE**：更新或修改起始点的值，当前值将被覆盖。如果参数值中的值太大，则会自动追加这些值。

startPoint（类型= **T_LONG**）

新值的起点。**value**（类型 = **TS_ValueSeq**）

要插入的值。**Java** 调用序列：**valueMatrix.setValueVector**（**col,set,startPoint,value**）；

返回：**None**。

可能的异常详见 13.3.18 节如下函数：

——**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**。

13.3 ASAM ODS 类型定义

本节除了规定数据类型的一般信息外，还包含了 **ASAO ODS** 枚举类型（例如数据类型、约束、关系、异常等）的表格和说明以及结构、联合等的定义。

13.3.1 ASAM ODS 数据类型

下表所示为 **OO-API** 的 **ASAM ODS** 数据类型，**ASAM** 对所有 **ASAM** 标准的数据类型进行了协调一致。2.5 节给出了 **ASAM** 和 **ODS** 数据类型的一般信息并描述了数据类型之间的关系。由于基于 **ASAM ODS** 数据类型可有多种实现方式，故在 **OO-API** 中保留它们。任何人都可以基于 **ASAM** 协调一致的数据类型开发。13.6 节提供的 **IDL** 文件包含 **ASAM ODS** 数据类型与协调一致 **ASAM** 数据类型之间的映射关系。

下表中数据类型的信息将用于 **OO-API**，尽管数据类型的名称常常被映射到特定的语境（例如 **Java**、**C++** 等）中的基本类型上，但枚举项的名称却是独立于平台和编程语言的。因此表中列出了枚举名称及其相应的数值和下划线数据类型的简短描述。有关信息详见 5.5 节。需要注意 **OO-API** 只传输枚举的数值。

ASAM ODS 数据类型枚举值见表 207。

表207 ASAM ODS 数据类型枚举值

数据类型枚举名		描述
DT_UNKNOWN	(=0)	Unknown 数据类型
DT_STRING	(=1)	字符串

DT_SHORT	(=2)	短整型（整数，16 位）
DT_FLOAT	(=3)	浮点值（32 位）
DT_BOOLEAN	(=4)	布尔值
DT_BYTE	(=5)	字节值（整数，8 位）
DT_LONG	(=6)	长整型（整数，32 位）
DT_DOUBLE	(=7)	双精度浮点值（64 位）
DT_LONGLONG	(=8)	长长整型（整数，64 位）
DT_ID	(=9)	Id（64 bit）；已弃用；使用 DT_LONGLONG
DT_DATE	(=10)	日期，以字符串形式给出
DT_BYTESTR	(=11)	字节流
DT_BLOB	(=12)	二进制大对象
DT_COMPLEX	(=13)	复数（每个部分 32 位）
DT_DCOMPLEX	(=14)	复数（每个部分 64 位）
DS_STRING	(=15)	字符串序列
DS_SHORT	(=16)	短序列
DS_FLOAT	(=17)	浮点值序列
DS_BOOLEAN	(=18)	布尔值序列
DS_BYTE	(=19)	字节序列
DS_LONG	(=20)	长整型序列
DS_DOUBLE	(=21)	双精度浮点序列
DS_LONGLONG	(=22)	长长整型序列
DS_COMPLEX	(=23)	复数序列
DS_DCOMPLEX	(=24)	双复数序列
DS_ID	(=25)	ID 序列；已弃用；见上文
DS_DATE	(=26)	数据序列
DS_BYTESTR	(=27)	字节串序列
DT_EXTERNALREFERENCE	(=28)	外部引用
DS_EXTERNALREFERENCE	(=29)	外部引用序列
DT_ENUM	(=30)	枚举
DS_ENUM	(=31)	枚举序列

13.3.2 数据类型转换

ASAM ODS 基础模型和 API 规范中包含的数据类型名称可能不同，2.5.2 节中描述了 API 数据类型和基础模型数据类型之间的关系。

在基础模型中，所有基础属性都有严格的预定义数据类型，正常情况下从基础属性派生的应用属性应当具有相同的数据类型。由于有些数据类型是为了兼容性而定义的，所以 ASAM ODS 允许应用模型的设计者重载基础属性的数据类型，2.4.3 节中描述了基础模型给定的属性数据类型的许可替代类型。该节中没有提到的所有基础属性只允许使用基础模型中定义的默认数据类型。

ASAM ODS 服务器和客户端必须注意属性的数据类型可以不同于基础模型中定义的数据类型。

类似的情况出现在基础关系中：它们的关系范围在基础模型中用一些许可的异常来描述，详见 5.4.6 节的说明。

13.3.3 ASAM ODS 常数

ASAM ODS常数见表208。

表208 ASAM ODS 常数

常数	值	名称	描述	数据类型
Lock Mode	0	LOCK_INSTANCEELEMENT	锁定实例（默认 LockMode）	T_SHORT
	1	LOCK_APPLICATIONELEMENT	锁定应用元素，该应用元素的所有实例都被锁定	T_SHORT
	2	LOCK_CHILDREN	锁定被锁定对象的子类。该模式可与上述 2 种模式组合使用	T_SHORT
QueryConstants	0	MaxDurationDEFAULT	查询的最大时长参数的默认值（无限定）	T_LONG
	“MaxDuration”	MaxDuration	查询的最大时长参数	T_STRING
	“QueryResultType”	QueryResultType	查询结果类型参数	T_STRING
	ResultType::INSTELEM_ITERATOR_AS_RESULT	QueryResultTypeDEFAULT	查询结果类型参数的默认值	T_SHORT
ResultType	0	INSTELEM_ITERATOR_AS_RESULT	作为查询结果的实例迭代器（默认）	T_SHORT
	1	TABLE_ITERATOR_AS_RESULT	作为查询的结果类型的表访问迭代器	T_SHORT
	2	TABLE_AS_RESULT	表作为查询的结果类型	T_SHORT
SecurityLevel	0	NO_SECURITY	未定义的安全	T_SHORT
	1	ELEMENT_SECURITY	应用元素的安全等级	T_SHORT
	2	INSTANCE_SECURITY	实例的安全等级	T_SHORT
	4	ATTRIBUTE_SECURITY	应用属性的安全等级	T_SHORT
SecurityRights	1	SEC_READ	允许读访问	T_SHORT
	2	SEC_UPDATE	允许对现有对象的更新访问	T_SHORT
	4	SEC_INSERT	允许创建新实例	T_SHORT
	8	SEC_DELETE	允许删除对象	T_SHORT
	16	SEC_GRANT	允许授权	T_SHORT

13.3.4 ASAM ODS 属性类型

AttrType的含义见表209。

表209 AttrType 的含义

枚举	排序关键字	名称	描述
AttrType	0	APPLATTR_ONLY	仅报告应用属性
	1	INSTATTR_ONLY	仅报告实例属性
	2	ALL	所有属性

13.3.5 ASAM ODS 构建函数

BuildUpFunction的含义见表210。

表210 BuildUpFunction 的含义

枚举	排序关键字	名称	描述
BuildUpFunction	0	BUP_JOIN	连接列
	1	BUP_MERGE	合并列
	2	BUP_SORT	列排序

13.3.6 ASAM ODS 查询状态

QueryStatus的含义见表211。

表211 QueryStatus 的含义

枚举	排序关键字	名称	描述
QueryStatus	0	COMPLETE	以完成执行
	1	INCOMPLETE	还在执行中

13.3.7 ASAM ODS 连接类型

JoinType的含义见表212。

表212 JoinType 的含义

枚举	排序关键字	名称	描述
JoinType	0	JTDEFAULT	强制内连接
	1	JTOUTER	强制外连接在目标 AID

13.3.8 ASAM ODS 权限设置

RightsSet的含义见表213。

表213 RightsSet 的含义

枚举	排序关键字	名称	描述
RightsSet	0	SET_RIGHT	设置给定的权限，覆盖已有的权限

	1	ADD_RIGHT	为已有的权限增加给定的权限
	2	REMOVE_RIGHT	从已有的权限中删除给定的权限

13.3.9 ASAM ODS 聚合函数

AggrFunc的含义见表214。

表214 AggrFunc 的含义

枚举	排序关键字	名称	描述
AggrFunc	0	NONE	没有适用于属性的聚合函数
	1	COUNT	计数
	2	DCOUNT	差异计数
	3	MIN	最小值；仅用于数值
	4	MAX	最大值；仅用于数值
	5	AVG	平均值；仅用于数值
	6	STDDEV	标准差；仅用于数值
	7	SUM	合计；仅用于数值
	8	DISTINCT	仅是差异值；如果一个属性值出现超过一次，则服务器创建的返回列表中仅考虑它的一次出现

13.3.10 ASAM ODS 选择操作代码

SelOpCode 给出查询指令，例如“等于”、“大于”等见表 215。目前这些参数是区分大小写的，但也需要增加大小写不区分的对比操作参数。因此在 SelOpCodes 中增加了不区分大小写的需求，这些参数有前缀“CI_”。

表215 SelOpCode 的含义

枚举	排序关键字	名称	描述
SelOpCode	0	EQ	等于
	1	NEQ	不等于
	2	LT	小于
	3	GT	大于
	4	LTE	小于等于
	5	GTE	大于等于
	6	INSET	在集合中，值可能是个序列
	7	NOTINSET	不在集合中，值可能是个序列
	8	LIKE	相似，使用模式匹配，见通配符定义的模式
	9	CI_EQ	等于（不区分大小写）

枚举	排序关键字	名称	描述
	10	CI_NEQ	不等于（不区分大小写）
	11	CI_LT	小于（不区分大小写）
	12	CI_GT	大于（不区分大小写）
	13	CI_LTE	小于等于（不区分大小写）
	14	CI_GTE	大于等于（不区分大小写）
	15	CI_INSET	在集合中，值可能是个序列（不区分大小写）
	16	CI_NOTINSET	不在集合中，值可能是个序列（不区分大小写）
	17	CI_LIKE	相似，使用模式匹配，见通配符定义的模式（不区分大小写）
	18	IS_NULL	值是 NULL
	19	IS_NOT_NULL	值不是 NULL
	20	NOTLIKE	不相似；选择 ‘attr’ 值不匹配 ‘value’ 中给定的模式的所有实例（‘attr’ 和 ‘value’ 是 SelValue 结构的元素）
	21	CI_NOTLIKE	不相似（不区分大小写）；选择 ‘attr’ 值不匹配 ‘value’ 中给定的模式的所有实例（‘attr’ 和 ‘value’ 是 SelValue 结构的元素）

13.3.11 ASAM ODS 选择操作符

SelOperator的含义见表216。

表216 SelOperator 的含义

枚举	排序关键字	名称	描述
SelOperator	0	AND	两个条件的交
	1	OR	两个条件的或
	2	NOT	下个条件无效
	3	OPEN	左方括号
	4	CLOSE	右方括号

13.3.12 ASAM ODS 选择类型

SelType的含义见表217。

表217 SelType 的含义

枚举	排序关键字	名称	描述
SelType	0	SEL_VALUE_TYPE	选择值
	1	SEL_OPERATOR_TYPE	选择逻辑操作符

13.3.13 ASAM ODS 设置类型

SetType的含义见表218。

表218 SetType 的含义

枚举	排序关键字	名称	描述
SetType	0	APPEND	向值矩阵附加数据
	1	INSERT	在值矩阵中插入数据
	2	UPDATE	修改值矩阵中已有的数据
	3	REMOVE	删除给定的信息

13.3.14 ASAM ODS 严重性标志

SeverityFlag的含义见表219。

表219 SeverityFlag 的含义

枚举	排序关键字	名称	描述
SeverityFlag	0	SUCCESS	成功
	1	INFORMATION	消息
	2	WARNING	警告
	3	ERROR	错误

13.3.15 ASAM ODS 值矩阵模式

ValueMatrixMode的含义见表220。

表220 ValueMatrixMode 的含义

枚举	排序关键字	名称	描述
ValueMatrixMode	0	CALCULATED	值矩阵返回计算值。 显式列的值按照它们物理存储的方式返回。 隐式列的值经过扩展后显式返回。 原始列的值根据相应算法计算（使用从基础属性“sequence_representation”和“generation_parameter”派生的应用属性值给定的算法）并返回最终结果的物理值。 该模式不允许对值矩阵进行写操作。
	1	STORAGE	值矩阵返回以物理存储方式给出的值。 显式列的值按照它们物理存储的方式返回。 隐式列的值经过扩展后显式返回。 原始列的值按照存储的方式返回（不使用从基础属性“sequence_representation”和“generation_parameter”派生的应用属性值给定的算法）并返回最终结果的物

			理值。 该模式允许对值矩阵进行写操作。

13.3.16 ASAM ODS 关系

关系表示两个 ASAM ODS 元素之间的联系，它可以有两个方向。第一个元素定义了 `get`-和 `set`-方法，从第二个元素出发的方法是 `getInverse`-和 `setInverse`-。

示例：该例表示 ODS 存储中的两个元素间的关系。假设两个元素及其关系如图 108 所示。

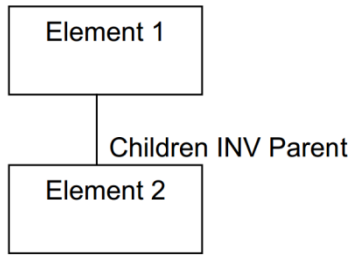


图108 两个元素之间的关系

```
GetRelationName (...) //returns "Children"
GetInverseRelationName (...) //returns "Parent"
GetElem1 (...) //returns Element 1
GetElem2 (...) //returns Element 2
```

当创建两个元素的关系时逆向关系也同时被创建，当删除该关系时同时逆向关系也被删除。这种情况适用于应用模型和实例。

当创建一个新关系时必须注意在相应的 `Set` 函数处给出正确的 “Element1” 和 “Element2”。

(1) `RELATION_TYPE` 关系类型

有三种类型的关系（描述见表 221）：

- `FATHER_CHILD`：两个元素之间的层次关系。`FATHER_CHILD` 关系适用于当通过 `ASAM-PATH` 访问一个实例时确定该实例的唯一性。（例如 `AoTest` 和 `AoSubTest`）；
- `INFO`：两个元素之间的信息关系。（例如 `AoMeasurementQuantity` 和 `AoQuantity`）；
- `INHERITANCE`：两个元素之间的继承关系，仅适用于具有相同基础类型的两个元素。

表221 `RelationType` 的含义

枚举	排序关键字	名称	描述
RelationType	0	FATHER_CHILD	父子关系
	1	INFO	信息关系
	2	INHERITANCE	继承关系

(2) RELATIONSHIP 联系

除了关系类型外还要有联系。当查询相关元素时需要联系而不是关系类型。有以下几类联系：

FATHER：在 FATHER_CHILD 关系类型中所需的父元素。该父元素是关系的“element 1”。

CHILD：在 FATHER_CHILD 关系类型中所需的子元素。该子元素是关系的“element 2”。

INFO_TO：在关系类型 INFO 中所需的目標元素，目標元素是关系的“element 2”。

INFO_FROM：在关系类型 INFO 中所需的源元素，源元素是关系的“element 1”。

INFO_REL：信息关系的方向不起作用，故与信息关系相关联的所有元素都被报告。

SUPERTYPE：给出继承关系的超类元素，超类元素是关系的“element 1”。

SUBTYPE：给出继承关系的子类元素，子类元素是关系的“element 2”。

ALL_REL：报告各种关系类型的所有关联元素。

InstanceElement 接口的 CreateRelation 和 RemoveRelation 方法能够识别哪些实例应用元素是应用关系的“element 1”哪些是“element 2”。

Relationship 的含义见表 222。

表222 Relationship 的含义

枚举	排序关键字	名称	描述
Relationship	0	FATHER	父
	1	CHILD	子
	2	INFO_TO	定向的信息关系
	3	INFO_FROM	定向的信息关系
	4	INFO_REL	信息关系（无方向）
	5	SUPERTYPE	继承关系：超类型
	6	SUBTYPE	继承关系：子类型
	7	ALL_REL	任何上述关系

(3) RELATION_RANGE 关系范围

关系范围规定了两个元素之间关系的许可数量，以下为预定义的关系范围：

Min 是关系的最小数量（0 或 1）；

Max 是关系的最大数量（1 或更多）。

getRelationRange、getInverseRelationRange、setRelationRange 和 setInverseRelationRange 方法适用于处理关系范围。

若使用 RELATION_RANGE，则关系不再需要使用强制性标志，也不再需要接口 BaseRelation 和 ApplicationRelation 的相应方法。

如果存在一个基础关系，则该基础关系的关系类型和关系范围适用于应用关系。如果不存在基础关系，则默认的关系类型是 INFO 并包含一个联系 INFO_TO，关系范围的设置取决于该属性的数据类型。数据类型枚举名 DT_* 意味着关系范围是 0:1，名称 DS_* 意味着关系范围是 0: 多。

关系范围的结构如下：

```
typedef struct RELATIONRANGE RelationRange;
```

```
struct RELATIONRANGE {
    T_SHORT min;
    T_SHORT max;
};
```

其中:

- min:** 关系的最小数量, 是 { -2, 0, 1 } 之一。如果最小值为 1, 则必须设置一个引用。值-2 意味着无初始化。
- max:** 关系的最大数量, 是 { -2, -1, 1 } 之一。值-1 意味着多个 { }; 值-2 意味着无初始化。

13.3.17 ASAM ODS 模式

Pattern (模式) 总是区分大小写的。
模式可以有如下所示的通配符:
——通配符 “?” 表示一个匹配字符对应一个问号;
——通配符 “*” 表示一个匹配字符序列对应着星号;
——通配符 ESCAPE **。

13.3.18 ASAM ODS 异常

当发生问题或出错时, ASAM ODS 的 OO-API 方法将引发异常 (AoException), 表 223 规定了 AoException 的结构。

表223 异常的结构 (AoException)

结构	数据类型	名称	描述
AoException	ErrorCode	errCode	该异常的错误代码。
	ServerityFlag	sevFlag	该异常的严重性(成功、消息、警告或是错误)。
	T_LONG	minorCode	次要代码, 用于该异常的进一步分类。
	T_STRING	reason	原因说明, 说明异常发生的原因。

异常的原因说明用字符串表示。原因字符串包含下表所列的内容之一。如果提供了附加的信息, 则原因描述后面带一个 “:”, 之后可提供任何与特定实现有关的信息。原因是按照字母顺序排序的。

与特定实现信息有关的异常原因说明示例如下:
“NetworkFailure: WSAENETDOWN: 网络子系统失效。”
如果没有与实现信息相关的内容, 则原因说明字符串将是:
“NetworkFailure ”

表 224 给出了所有异常及其说明, 相应的枚举项编号在括号中给出。

表224 OO-API 的异常及其错误代码

异常 (错误代码)	说明

异常（错误代码）	说明
AO_ACCESS_DENIED (1)	远程服务器拒绝访问。如果发生此错误，甚至无法提供身份验证信息。这意味着身份验证信息可能是正确的，但服务器拒绝了较低级别的访问。
AO_BAD_OPERATION (2)	当方法在封送处理操作中无效时，将使用错误的 BAD_OPERATION 操作错误代码。
AO_BAD_PARAMETER (3)	向方法传递了错误类型的参数。 minorcode 指示哪个参数（1、2、3 或 4）不正确。如果有多个参数不正确，则只报告第一个参数。此错误只能出现在非类型保存语言绑定中。这些语言绑定也带来了一个问题，即并非所有参数错误都能自动检测到。
AO_CONNECTION_LOST (6)	由于硬件或网络软件问题，与服务器的连接丢失。
AO_CONNECT_FAILED (4)	连接到服务器失败。如果服务器关闭或当前无法访问，则可能发生此错误。
AO_CONNECT_REFUSED (5)	服务器拒绝了连接。如果提供的身份验证信息不正确或不完整，则可能发生此错误。如果服务器由于过载问题而不再接受任何会话时，则不会发生此错误。本例见 AO_SESSION_LIMIT_REACHED 。
AO_DUPLICATE_BASE_ATTRIBUTE (7)	任何应用元素只能有一个特定类型的基属性。这意味着它可能只有一个基本属性类型名称、一个 ID、一个版本等属性。
AO_DUPLICATE_NAME (8)	隐式或显式指定的名称已在使用中，但必须是唯一的。
AO_DUPLICATE_VALUE (9)	该属性在应用模型中标记为唯一。因此不允许重复值。
AO_HAS_BASE_ATTRIBUTE (51)	创建基础属性。不允许修改数据类型、唯一标志和强制标志。
AO_HAS_BASE_RELATION (50)	创建基础关系。不允许修改从基关系派生的应用程序关系的关系类型、关系范围或关系。
AO_HAS_INSTANCES (10)	不允许对具有实例的元素执行该操作。
AO_HAS_REFERENCES (11)	不允许执行请求的操作，因为目标元素具有引用。
AO_IMPLEMENTATION_PROBLEM (12)	此错误保留用于报告标准错误定义未正确处理的特定于实现的问题。如果发生此错误，应用程序不应崩溃，但除了报告和忽略此错误之外，没有其他方法可以对此错误做出响应。此错误的目的不是不报告特定于实现的错误。
AO_INCOMPATIBLE_UNITS (13)	单位不兼容。不知道转换规则。
AO_INVALID_ASAM_PATH (14)	指定的 asam 路径无效。
AO_INVALID_ATTRIBUTE_TYPE (15)	请求的属性类型无效。
AO_INVALID_BASETYPE (17)	指定的基类型无效。
AO_INVALID_BASE_ELEMENT (16)	基元素在此上下文中无效。如果这是 AOMeasurement 类型的元素，则此类型的另一个元

异常（错误代码）	说明
	素可能已经存在。
AO_INVALID_BUILDUP_FUNCTION (18)	指定的生成函数无效。
AO_INVALID_COLUMN (19)	指定的列无效。
AO_INVALID_COUNT (20)	指定的点数无效（可能为负数）。
AO_INVALID_DATATYPE (21)	在给定的上下文中不允许该数据类型，或者该数据类型与现有的数据类型定义冲突。为避免在所有语言绑定中出现此错误，建议始终使用枚举“datatype”的定义。
AO_INVALID_ELEMENT (22)	元素在此上下文中无效。
AO_INVALID_LENGTH (23)	给定的长度无效。不允许长度值为负。
AO_INVALID_ORDINALNUMBER (24)	序号已被使用或小于零。
AO_INVALID_RELATION (25)	关系无效。相关元素和基关系不匹配。
AO_INVALID_RELATIONSHIP (28)	此错误只能出现在非类型安全的语言绑定中。为了避免在所有语言绑定中出现此错误，建议始终使用枚举“relationship”的定义。
AO_INVALID_RELATION_RANGE (26)	指定的关系范围无效。
AO_INVALID_RELATION_TYPE (27)	此错误只能出现在非类型安全的语言绑定中。为了避免在所有语言绑定中出现此错误，建议始终使用枚举“relationtype”的定义。
AO_INVALID_SET_TYPE (29)	指定的集类型无效。
AO_INVALID_SMATLINK (30)	Submatrix 链接无效。如果有多个 smatlink，则未指定 submatrix 1 或 2，或者缺少序号。
AO_INVALID_SUBMATRIX (31)	指定的子矩阵无效。
AO_INVALID_VALUEMATRIX_STRUCTURE (60)	当由于度量值的数据而无法创建 ValueMatrix 时，服务器将抛出此异常。就是这样 -如果没有独立柱； -如果有不同的独立柱； -如果存在具有多个独立柱的子矩阵。
AO_IS_BASE_ATTRIBUTE (32)	应用程序属性是基属性类型。它不能改变。如果需要这样做，则必须从应用元素中删除应用程序属性并重新创建。如果从基属性派生的应用程序属性可能发生此错误 应由不同的基本属性类型覆盖。 -应接收不同的数据类型。 -应接收不同的唯一标志。 — -应收到不同的强制性标志。
AO_IS_BASE_RELATION (33)	不能更改基关系的属性。
AO_IS_MEASUREMENT_MATRIX (34)	矩阵是一个复杂的，由测量生成的矩阵，而不仅仅是一个简单的子矩阵。只允许修改子矩阵，不允许修改组成的测量矩阵。
AO_MATH_ERROR (35)	发生计算错误。这可以是溢出、下溢或被零除。

异常（错误代码）	说明
AO_MISSING_APPLICATION_ELEMENT (36)	缺少必需的应用元素。
AO_MISSING_ATTRIBUTE (37)	缺少必需（必需）属性。
AO_MISSING_RELATION (38)	缺少必需的关系。
AO_MISSING_VALUE (39)	缺少强制值。
AO_NOT_FOUND (42)	找不到请求的元素。只有在未找到操作主题的情况下，才会在删除或重命名操作中发生此错误。如果找不到请求的项，则所有 get 和 list 方法都返回空列表。
AO_NOT_IMPLEMENTED (43)	请求的方法尚未实现。在经过认证的实现中不允许出现此错误（除非客户机访问查询接口；查询接口不是 ASAM ODS 5.2.0 版的正式功能；如果客户机访问该接口，服务器必须抛出此异常）。它旨在允许在开发过程中进行部分操作测试。
AO_NOT_UNIQUE (44)	如果要求属性的实例唯一，则会发生此错误。
AO_NO_MEMORY (40)	没有更多的易失性内存可用。
AO_NO_PATH_TO_ELEMENT (41)	检测到一个自由浮动元素。没有指向此元素的导航路径。
AO_NO_SCALING_COLUMN (53)	该列不是缩放列。
AO_OPEN_MODE_NOT_SUPPORTED (45)	不支持请求的打开模式。有效的打开模式是“读”和“写”。由于此错误，任何其他内容都将被拒绝，并且不会创建任何会话。
AO_QUERY_INCOMPLETE (58)	查询的执行尚未完成。
AO_QUERY_INVALID (55)	查询字符串中存在错误，或者查询字符串的返回类型与参数“ queryresultType ”指定的结果类型不一致。
AO_QUERY_INVALID_RESULTTYPE (59)	查询的请求结果类型与结果类型的上一个定义不匹配。
AO_QUERY_PROCESSING_ERROR (56)	执行查询时出错。
AO_QUERY_TIMEOUT_EXCEEDED (57)	无法在参数“ maxduration ”设置的时间限制内执行查询。
AO_QUERY_TYPE_INVALID (54)	服务器不支持指定的查询语言类型。
AO_SESSION_LIMIT_REACHED (46)	服务器不接受任何新连接。如果服务器达到不同用户的会话限制或允许的会话总数，则可能发生此错误。
AO_SESSION_NOT_ACTIVE (47)	会话不再处于活动状态。如果试图调用已关闭会话的方法，则会发生此错误。此错误不应与 AO_CONNECTION_LOST 丢失的错误混淆。
AO_TRANSACTION_ALREADY_ACTIVE (48)	一次只能有一个活动事务。如果发生此错误，则已存在活动事务。如果出现此错误，该事务将保持活动状态。
AO_TRANSACTION_NOT_ACTIVE (49)	必须始终在事务上下文中执行写操作。如果在写入操作期间没有活动事务，则会发生此错误。
AO_UNKNOWN_ERROR (0)	使用零作为未知错误，以避免在未设置错误代码的情况下混淆错误消息。

异常（错误代码）	说明
AO_UNKNOWN_UNIT (52)	单位未知。

13.3.19 ASAM ODS 特定类型

ASAM ODS 00_API特定的数据类型见表225。

表225 ASAM ODS 00_API 特定的数据类型

类型	数据类型	预定义类型	描述
Scalar	string	T_STRING	
Scalar	boolean	T_BOOLEAN	
Scalar	short	T_SHORT	
Scalar	float	T_FLOAT	
Scalar	octet	T_BYTE	
Scalar	long	T_LONG	
Scalar	double	T_DOUBLE	
Scalar	T_STRING	Name	
Scalar	T_STRING	Pattern	一个字符的通配符是“?”。多个字符的通配符为“*”。
Scalar	T_STRING	BaseType	
Scalar	T_STRING	T_DATE	
Scalar	Blob	T_BLOB	
Sequence	T_BYTE	T_BYTESTR	
Sequence	T_BOOLEAN	S_BOOLEAN	
Sequence	T_BYTE	S_BYTE	
Sequence	T_DOUBLE	S_DOUBLE	
Sequence	T_FLOAT	S_FLOAT	
Sequence	T_LONG	S_LONG	
Sequence	T_SHORT	S_SHORT	
Sequence	T_STRING	S_STRING	
Sequence	T_COMPLEX	S_COMPLEX	
Sequence	T_DCOMPLEX	S_DCOMPLEX	
Sequence	T_LONGLONG	S_LONGLONG	
Sequence	BaseType	BaseTypeSequence	
Sequence	Name	NameSequence	
Sequence	NameValue	NameValueSequence	
Sequence	NameValueUnit	NameValueUnitSequence	
Sequence	Column	ColumnSequence	
Sequence	SMatLink	SMatLinkSequence	
Sequence	SubMatrix	SubMatrixSequence	
Sequence	T_DATE	S_DATE	

类型	数据类型	预定义类型	描述
Sequence	T_BYTESTR	S_BYTESTR	
Sequence	S_STRING	SS_STRING	
Sequence	S_SHORT	SS_SHORT	
Sequence	S_FLOAT	SS_FLOAT	
Sequence	S_BOOLEAN	SS_BOOLEAN	
Sequence	S_BYTE	SS_BYTE	
Sequence	S_LONG	SS_LONG	
Sequence	S_DOUBLE	SS_DOUBLE	
Sequence	S_LONGLONG	SS_LONGLONG	
Sequence	S_COMPLEX	SS_COMPLEX	
Sequence	S_DCOMPLEX	SS_DCOMPLEX	
Sequence	S_DATE	SS_DATE	
Sequence	S_BYTESTR	SS_BYTESTR	
Sequence	T_BLOB	S_BLOB	
Sequence	ApplicationElement	ApplicationElementSequence	
Sequence	ApplicationRelation	ApplicationRelationSequence	
Sequence	ApplicationAttribute	ApplicationAttributeSequence	
Sequence	BaseRelation	BaseRelationSequence	
Sequence	BaseAttribute	BaseAttributeSequence	
Sequence	BaseElement	BaseElementSequence	
Sequence	InstanceElement	InstanceElementSequence	
Sequence	NameValueSeqUnit	NameValueSeqUnitSequence	
Sequence	T_ExternalReference	S_ExternalReference	
Sequence	S_ExternalReference	SS_ExternalReference	
Sequence	ApplAttr	ApplAttrSequence	
Sequence	ApplElem	ApplElemSequence	应用元素定义序列。应用元素通过接口 <code>applicationstructure</code> 中的 <code>getElements()</code> 方法提供。
Sequence	ApplRel	ApplRelSequence	应用关系序列。应用程序关系通过接口 <code>applicationstructure</code> 中的 <code>getRelations()</code> 方法给出。
Sequence	AIDName	AIDNameSequence	AID-Name 名称对。
Sequence	AIDNameUnitId	AIDNameUnitIdSequence	AID-Name-UnitID 数据序列。
Sequence	AIDNameValueSeqUnitId	AIDNameValueSeqUnitIdSequence	AID-Name-UnitID 数据序列数组。
Sequence	ElemId	ElemIdSequence	元素 ID 的序列。
Sequence	AttrResultSet	AttrResultSetSequence	属性结果集的序列。
Sequence	ElemResultSet	ElemResultSetSequence	元素结果集的序列。
Sequence	SelOrder	SelOrderSequence	顺序标准的顺序。第一个标准是重要性标准。

类型	数据类型	预定义类型	描述
Sequence	SelValue	SelValueSequence	属性搜索条件的序列。
Sequence	SelOperator	SelOperatorSequence	
Sequence	ACL	ACLSequence	所请求对象的访问控制列表项的顺序。
Sequence	EnumerationAttributeStructure	EnumerationAttributeStructureSequence	枚举类型的所有属性的序列。
Sequence	EnumerationItemStructure	EnumerationItemStructureSequence	枚举中所有项的顺序。
Sequence	EnumerationStructure	EnumerationStructureSequence	应用程序模型中所有枚举的序列。
Sequence	InitialRight	InitialRightSequence	初始权利的顺序。
Sequence	SelAIDNameUnitId	SelAIDNameUnitIdSequence	包含 attr.name-unitid-aggregate 的 数组序列。
Sequence	JoinDef	JoinDefSequence	连接定义的序列。
Sequence	ApplicationRelationInstanceElementSeq	ApplicationRelationInstanceElementSeqSequence	序列数组, 每个元组包含一个应用元素及其相关实例的序列。
Sequence	NameValueSeqUnitId	NameValueSeqUnitIdSequence	序列数组, 每个都包含 name-values-unitid。
Sequence	SelItem	SelItemSequence	选择项的序列。
Sequence	ElemResultSetExt	ElemResultSetExtSequence	序列数组, 每个包含 aid 和一个 name-values-unitid 序列。
Sequence	ResultSetExt	ResultSetExtSequence	序列数组, 每个元组包含一个 elemresultsetext 序列和一个其 余的迭代器。

13.3.20 ASAM ODS 联合

ASAM ODS 00-API联合见表226。

表226 ASAM ODS 00-API 联合

联合名	用例名	数据类型	名称
SelItem	SEL_VALUE_TYPE	SelValueExt	value
SelItem	SEL_OPERATOR_TYPE	SelOperator	operator
TS_Union	DT_STRING	T_STRING	stringVal
TS_Union	DT_SHORT	T_SHORT	shortVal
TS_Union	DT_FLOAT	T_FLOAT	floatVal
TS_Union	DT_BYTE	T_BYTE	byteVal
TS_Union	DT_BOOLEAN	T_BOOLEAN	booleanVal
TS_Union	DT_LONG	T_LONG	longVal
TS_Union	DT_DOUBLE	T_DOUBLE	doubleVal

联合名	用例名	数据类型	名称
TS_Union	DT_LONGLONG	T_LONGLONG	longlongVal
TS_Union	DT_COMPLEX	T_COMPLEX	complexVal
TS_Union	DT_DCOMPLEX	T_DCOMPLEX	dcomplexVal
TS_Union	DT_DATE	T_DATE	dateVal
TS_Union	DT_BYTESTR	T_BYTESTR	bytestrVal
TS_Union	DT_BLOB	T_BLOB	blobVal
TS_Union	DS_STRING	S_STRING	stringSeq
TS_Union	DS_SHORT	S_SHORT	shortSeq
TS_Union	DS_FLOAT	S_FLOAT	floatSeq
TS_Union	DS_BYTE	S_BYTE	byteSeq
TS_Union	DS_BOOLEAN	S_BOOLEAN	booleanSeq
TS_Union	DS_LONG	S_LONG	longSeq
TS_Union	DS_DOUBLE	S_DOUBLE	doubleSeq
TS_Union	DS_LONGLONG	S_LONGLONG	longlongSeq
TS_Union	DS_COMPLEX	S_COMPLEX	complexSeq
TS_Union	DS_DCOMPLEX	S_DCOMPLEX	dcomplexSeq
TS_Union	DS_DATE	S_DATE	dateSeq
TS_Union	DS_BYTESTR	S_BYTESTR	bytestrSeq
TS_Union	DT_EXTERNALREFERENCE	T_ExternalReference	extRefVal
TS_Union	DS_EXTERNALREFERENCE	S_ExternalReference	extRefSeq
TS_Union	DT_ENUM	T_LONG	enumVal
TS_Union	DS_ENUM	S_LONG	enumSeq
TS_UnionSeq	DT_STRING	S_STRING	stringVal
TS_UnionSeq	DT_SHORT	S_SHORT	shortVal
TS_UnionSeq	DT_FLOAT	S_FLOAT	floatVal
TS_UnionSeq	DT_BYTE	S_BYTE	byteVal
TS_UnionSeq	DT_BOOLEAN	S_BOOLEAN	booleanVal
TS_UnionSeq	DT_LONG	S_LONG	longVal
TS_UnionSeq	DT_DOUBLE	S_DOUBLE	doubleVal
TS_UnionSeq	DT_LONGLONG	S_LONGLONG	longlongVal
TS_UnionSeq	DT_COMPLEX	S_COMPLEX	complexVal
TS_UnionSeq	DT_DCOMPLEX	S_DCOMPLEX	dcomplexVal
TS_UnionSeq	DT_DATE	S_DATE	dateVal
TS_UnionSeq	DT_BYTESTR	S_BYTESTR	bytestrVal
TS_UnionSeq	DT_BLOB	S_BLOB	blobVal
TS_UnionSeq	DS_STRING	SS_STRING	stringSeq
TS_UnionSeq	DS_SHORT	SS_SHORT	shortSeq
TS_UnionSeq	DS_FLOAT	SS_FLOAT	floatSeq
TS_UnionSeq	DS_BYTE	SS_BYTE	byteSeq
TS_UnionSeq	DS_BOOLEAN	SS_BOOLEAN	booleanSeq

联合名	用例名	数据类型	名称
TS_UnionSeq	DS_LONG	SS_LONG	longSeq
TS_UnionSeq	DS_DOUBLE	SS_DOUBLE	doubleSeq
TS_UnionSeq	DS_LONGLONG	SS_LONGLONG	longlongSeq
TS_UnionSeq	DS_COMPLEX	SS_COMPLEX	complexSeq
TS_UnionSeq	DS_DCOMPLEX	SS_DCOMPLEX	dcomplexSeq
TS_UnionSeq	DS_DATE	SS_DATE	dateSeq
TS_UnionSeq	DS_BYTESTR	SS_BYTESTR	bytestrSeq
TS_UnionSeq	DT_EXTERNALREFERENCE	S_ExternalReference	extRefVal
TS_UnionSeq	DS_EXTERNALREFERENCE	SS_ExternalReference	extRefSeq
TS_UnionSeq	DT_ENUM	S_LONG	enumVal
TS_UnionSeq	DS_ENUM	SS_LONG	enumSeq

13.3.21 ASAM ODS 结构

下列各表描述了 ASAM ODS 的结构，当一个数据类型被引用且用 **DataType** 枚举表达时，应说明它的的枚举名。

安全相关的结构（ACL，...）见表 227。

属性 ID 结构（AID...）见表 228。

应用属性的结构（ApplAttr）见表 229。

应用元素的结构（ApplElem）见表 230。

应用结构值的结构（ApplicationStructureValue）见表 231。

应用关系的结构（ApplRel）见表 232。

应用关系与实例（ApplicationRelationInstanceElementSeq）见表 233。

属性结果集的结构（AttrResultSet）见表 234。

扩展结果集的结构（ResultSetExt）见表 235。

元素 ID 的结构（ElemId）见表 236。

元素结果集的结构（ElemResultSet）见表 237。

扩展元素结果集的结构（ElemResultSetExt）见表 238。

名称单位的结构（NameUnit）见表 239。

名称值的结构（NameValue...）见表 240。

查询的结构（QueryStructure）见表 241。

扩展查询的结构（QueryStructureExt）见表 242。

关系范围的结构（RelationRange）见表 243。

选择顺序的结构（SelOrder）见表 244。

选择值的结构（SelValue）见表 245。

扩展选择值的结构（SelValueExt）见表 246。

连接的定义（JoinDef）见表 247。

选择带 UnitId 的 AIDName 和聚合（SelAIDNameUnitId）见表 248。

枚举属性的结构（EnumerationAttributeStructure）见表 249。

枚举项的结构（EnumerationItemStructure）见表 250。

枚举的结构（EnumerationStructure）见表 251。

结构类型定义见表 252。

表227 安全相关的结构（ACL，...）

结构	数据类型	名称	描述
ACL	T_SHORT	rights	所需对象的访问权限
	T_LONGLONG	usergroupId	用户组标识 ID
InitialRight	T_LONG	rights	所需数据对象的初始访问权限
	T_LONGLONG	usergroupId	初始权限列表的用户组标识 ID
	T_LONGLONG	refAID	引用它的应用元素

表228 属性 ID 结构（AID...）

结构	数据类型	名称	描述
AIDName	Name	aaName	属性或被测量量的名称
	T_LONGLONG	aid	应用元素的标识 ID
AIDNameUnitId	T_LONGLONG	unitId	列的属性单位。unitId 是基础类型 AoUnit 实例的标识 ID
	AIDName	attr	应用元素的属性（aid，name）
AIDNameValueSeqUnitId	T_LONGLONG	unitId	列的属性单位。unitId 是基础类型 AoUnit 实例的标识 ID
	AIDName	attr	应用元素的属性（aid，name）
	TS_ValueSeq	values	带有值标志的列值
AIDNameValueUnitId	T_LONGLONG	unitId	列的属性单位。unitId 是基础类型 AoUnit 实例的标识 ID
	TS_Value	values	带有值标志的属性值
	AIDName	attr	应用元素的属性（aid，name）

表229 应用属性的结构（ApplAttr）

结构	数据类型	名称	描述
ApplAttr	T_BOOLEAN	isUnique	唯一属性的指示器。相同的布尔值由 interface applicationattribute 的 isunique（）方法返回。
	T_BOOLEAN	isObligatory	在物理存储器的表的列中设置强制属性的指示符 notNullIndicator。在 interfaceApplication 属性的方法 isobligatory（）上返回相同的布尔值。
	T_LONG	length	值的最大可能长度。samelength 由 interface

结构	数据类型	名称	描述
			applicationattribute 的 getLength () 方法返回。
	DataType	dType	属性数据类型。接口应用程序属性的 getdatatype 方法提供了相同的数据类型。在 rpc-api 中, 此信息存储在 structureattrseq 和 request aop_getattr 的 adatatype 字段中。
	Name	baName	基属性的名称, 如果应用程序属性不是从基属性派生的, 则为空 (“ ”)。baseattribute 接口的 methodsgetname () 返回相同的名称。baseattribute 由接口 applicationattribute 的 getbaseattribute () 方法提供。在 rpc api 中, 此信息存储在 structure attrseq 和 request aop_getattr 的 abname 字段中。
	Name	aaName	应用程序属性名称。应用程序属性接口的 getname () 方法将返回相同的名称。在 rpc-api 中, 此信息存储在结构 attrseq 和请求 aop_getattr 的 aaname 字段中。
	T_LONGLONG	unitId	全局定义的单位的 ID。相同的 ID 是由 InterfaceApplicationAttribute 的 getUnit () 方法返回。在 rpc-api 中, 此信息存储在结构 attrseqand 请求 aop_getattr 的 aunit 字段中。

表230 应用元素的结构 (ApplElem)

结构	数据类型	名称	描述
ApplElem	Name	beName	基本元素名称。所有应用元素都与基元素相关。baseElement 接口的 getName () 方法返回相同的名称。基本元素通过接口 applicationElement 处的 getBaseElement () 方法提供。在 rpc-api 上, 未传递信息, 但基本元素的对应 ID 存储在结构 applinfseq 和请求 aop_getapplinf 的 aibid 字段中。
	Name	aeName	应用元素名称。该名称也与接口上的 getname () 方法一起给出。 应用元素。在 rpc-api 中, 此信息存储在结构 applinfseq 的 ainame 字段和请求 aop_getapplinf 字段中。
	T_LONGLONG	aid	应用元素 ID。该 ID 也在接口 applicationElement 上用 themethod getid () 给出。在 therpc api 中, 此信息存储在结构 applinfseq 和请求 aop_getapplinf 的 aaid 字段中。
	ApplAttr Sequence	attributes	应用元素的属性。这些属性是通过 interfaceApplicationElement 的 getAttributes () 方法提供的。这个顺序中没有给出任何关系。

表231 应用结构值的结构 (ApplicationStructureValue)

结构	数据类型	名称	描述
ApplicationStructureValue	ApplRelSequence	applRels	应用模型中的关系列表。应用元素列表包含关系及其反义词的不同中心。字段 invname 包含 PartnerRelation 。正态关系和逆关系都在序列中给出。
	ApplElemSequence	applElems	应用元素列表。

表232 应用关系的结构 (ApplRel)

结构	数据类型	名称	描述
ApplRel	RelationRange	invRelationRange	反关系的范围。范围关系未存储在物理存储中。反向关系范围在 Interface ApplicationRelation 的 Themethod GetInverseRelationRange () 处返回。
	RelationRange	arRelationRange	关系的范围。的范围关系未存储在物理存储中。在方法处返回相关范围接口 applicationRelation 的 getRelationRange () 。
	RelationType	arRelationType	关系的类型。不存储在物理存储中的关系类型。在接口 applicationRelation 的 getRelationType () 方法中返回 RelationType 。
	Name	invBrName	来自的逆基关系的名称 elem2 到 elem1 。基本关系不是存储在物理存储器中。
	T_LONGLONG	elem2	目标应用元素 ID。给定的 ID 是从接口的 getelem2 () 方法返回的应用元素的 ID。 应用关系。在 rpc-api 中，此信息存储在结构 applrelseq 和 requestaop_getapplinf 。
	Name	brName	从 elem1 到 elem2 的基关系的名称。基本关系不存储在物理存储中。
	T_LONGLONG	elem1	源应用元素 ID。给定的 ID 是从接口的 getelem1 () 方法返回的应用元素的 ID。 应用关系。在 rpc-api 中，此信息存储在结构 applrelseq 和 requestaop_getapplinf 。
	Name	invName	反关系的名称。名字是返回的方法 getInverseName () 。 接口应用程序关系。 invname 在关系数据库的物理存储中不可用
	Name	arName	关系名称。返回名称时接口 applicationrelation 的 getname () 方法。在 rpc-api 中，此信息存储在结构 applrelseq 和 requestaop_getapplinf 的 arname 字段中。

表233 应用关系与实例 (ApplicationRelationInstanceElementSeq)

结构	数据类型	名称	描述
ApplicationRelationInstanceElementSeq	ApplicationRelation	applRel	应用关系。
	InstanceElementSequence	instances	包含实例的列表。实例的应用元素列表中必须与应用关系的应用元素之一匹配的所有实例列表必须具有 sameapplication 元素。

表234 属性结果集的结构 (AttrResultSet)

结构	数据类型	名称	描述
AttrResultSet	NameValueSeqUnitId	attrValues	一个元素的属性结果集。所有值具有相同的 aidname 和相同的 unitid。此结果集中只有可能的属性值的第一部分可用。该部分的大小取决于接口 “applelemaccess” 的方法 “getInstances” 的参数 “多少”。
	NameValueUnitIdIterator	rest	剩下的结果。迭代器为连续结果集的名称值初始化包。

表235 扩展结果集的结构 (ResultSetExt)

结构	数据类型	名称	描述
ResultSetExt	ElemResultSetExtSequence	firstElems	第一个 how_many 结果元素的序列。
	ElemResultSetExtSeqIterator	restElems	迭代器，它允许对结果值、每个迭代的一个实例的属性进行迭代。

表236 元素 ID 的结构 (ElemId)

结构	数据类型	名称	描述
ElemId	T_LONGLONG	iid	实例的 ID。
	T_LONGLONG	aid	应用元素的 ID。

表237 元素结果集的结构 (ElemResultSet)

结构	数据类型	名称	描述
ElemResultSet	T_LONGLONG	aid	应用元素的 ID。
	AttrResultSetSequence	attrValues	选定元素的属性。每个 attrresultset 中的值的数目相同，一个元素的属性总是位于

			attrresultset 中。

表238 扩展元素结果集的结构 (ElemResultSetExt)

结构	数据类型	名称	描述
ElemResultSetExt	T_LONGLONG	aid	应用元素的 ID。
	NameValueSeqUnitIdSequence	values	给定应用元素实例的属性值。

表239 名称单位的结构 (NameUnit)

结构	数据类型	名称	描述
NameUnit	T_STRING	unit	列单位为字符串。
	Name	valName	属性名称或测量的数量名称。

表240 名称值的结构 (NameValue...)

结构	数据类型	名称	描述
NameValue	TS_Value	value	属性值或列值 (向量)。
	Name	valName	属性名称或测量的数量名称。
NameValueSeqUnit	Name	valName	列名称或测量的数量名称。
	T_STRING	unit	列单位为字符串。
	TS_ValueSeq	value	列值 (矢量)。
NameValueSeqUnitId	T_LONGLONG	unitId	列单位为 ID。
	TS_ValueSeq	value	列值 (矢量)。
	Name	valName	列名称或测量的数量名称。
NameValueUnit	TS_Value	value	属性值或列值 (向量)。
	T_STRING	unit	属性或列单位作为字符串。
	Name	valName	属性名称或测量的数量名称。
NameValueUnitId	Name	valName	属性名称或测量的数量名称。
	T_LONGLONG	unitId	属性或列单位的 ID。
	TS_Value	value	属性值或列值 (向量)。

表241 查询的结构 (QueryStructure)

结构	数据类型	名称	描述
QueryStructure	SelValueSequence	condSeq	查询条件序列。在 rpc-api 中, 此信息存储在结构 getvalreq 和请求 aop getval 的 nsseq 字段中。
	SelOrderSequence	orderBy	按顺序排序。结果集的顺序。在 rpc-api 上, 无法设置顺序。
	Name	relName	关系的名称。在 rpc-api 中, 此信息存储在结

结构	数据类型	名称	描述
			构 getvalreq 和请求 aop_getval 的字段 refname 中。
	SelOperatorSequence	operSeq	查询条件运算符序列。在 rpc-api 中，运算符始终是 “and”。
	AIDNameUnitIdSequence	anuSeq	要报告的属性序列。在 rpc-api 中，这些信息存储在 getvalreq 结构和请求 aop_getval 的 applid 和 nuseq 字段中。在 rpc-api 中，只能选择一个应用元素。属性名只接受 apattern。
	ElemId	rellnst	相关实例。(aid==0&&iid==0) 表示未指定相关实例。在 rpc-api 中，此信息存储在结构 getvalreq 和请求 aop_getval 的 elemid 字段中。

表242 扩展查询的结构（QueryStructureExt）

结构	数据类型	名称	描述
QueryStructureExt	SelAIDNameUnitIdSequence	anuSeq	要报告的属性序列。在 RPC-API 中，这些信息存储在结构 getvalreq 和 requestaop_getval 的 applid 和 nuseq 字段中。在 rpc-api 中，只能选择一个应用元素。属性名接受 apattern
	SetItemSequence	condSeq	查询条件序列。在 rpc-api 中，此信息存储在结构 getvalreq 和请求 aop_getval 的 nsseq 字段中。
	JoinDefSequence	joinSeq	定义应用元素之间的联接。
	SelOrderSequence	orderBy	按顺序排序。结果集的顺序。在 rpc-api 上，无法设置顺序。
	AIDNameSequence	groupBy	定义的分组属性 请求,如果在 selaidnameUnitIdSequence 中定义了聚合函数，则是必需的。

表243 关系范围的结构（RelationRange）

结构	数据类型	名称	描述
RelationRange	T_SHORT	max	范围内的最大值。-1 表示没有指定最大值的任何值。
	T_SHORT	min	范围内的最小数字。

表244 选择顺序的结构（SelOrder）

结构	数据类型	名称	描述
SelOrder	AIDName	attr	属性规范。
	T_BOOLEAN	ascending	升序，False 表示降序。

表245 选择值的结构 (SelValue)

结构	数据类型	名称	描述
SelValue	TS_Value	value	条件的值。
	SelOpcode	oper	属性和值之间的比较运算符。
	AIDNameValueUnitId	attr	具有值单位的属性规范。

表246 扩展选择值的结构 (SelValueExt)

结构	数据类型	名称	描述
SelValueExt	AIDNameUnitId	attr	具有单元 ID 的属性规范。
	SelOpcode	oper	属性和值之间的比较运算符。
	TS_Value	value	条件的值。

表247 连接的定义 (JoinDef)

结构	数据类型	名称	描述
JoinDef	T_LONGLONG	fromAID	源元素的应用元素 ID。
	T_LONGLONG	toAID	目标元素的应用元素 ID。
	Name	refName	与要联接的元素相关的应用关系的名称。
	JoinType	joiningType	联接的类型。

表248 选择带 UnitId 的 AIDName 和聚合 (SelAIDNameUnitId)

结构	数据类型	名称	描述
SelAIDNameUnitId	AIDName	attr	应用元素的属性 (aid, name)
	T_LONGLONG	unitId	列属性的单位。 UnitID 是具有 BaseType AUnit 的实例的 ID。
	AggrFunc	aggregate	聚合函数。

表249 枚举属性的结构 (EnumerationAttributeStructure)

结构	数据类型	名称	描述
EnumerationAttributeStructure	T_LONGLONG	aid	应用元素的 ID。
	T_STRING	aaName	枚举类型的应用属性的名称。

	T_STRING	enumName	提供允许的属性值的枚举的名称。

表250 枚举项的结构 (EnumerationItemStructure)

结构	数据类型	名称	描述
EnumerationItemStructure	T_LONG	index	枚举项的索引。
	T_STRING	itemName	枚举项的名称。

表251 枚举的结构 (EnumerationStructure)

结构	数据类型	名称	描述
EnumerationStructure	T_STRING	enumName	枚举的名称
	EnumerationItemStructureSequence	items	枚举的所有项的列表

表252 结构类型定义

结构	数据类型	名称	描述
T_COMPLEX	T_FLOAT	i	复数的虚数部分。
	T_FLOAT	r	复数的实部。
T_DCOMPLEX	T_DOUBLE	i	双精度复数的虚部。
	T_DOUBLE	r	双精度复数的实部。
T_ExternalReference	T_STRING	location	外部引用的位置。(ASAM 路径或 URL)
	T_STRING	contentType	外部对象的 mime 类型。
	T_STRING	description	外部引用的描述。
T_LONGLONG	T_LONG	low	64 位值的最低有效 32 位。这应被视为 32 位无符号整数，这样 t_longlong 总共具有与 64 位有符号整数相同的物理表示。
	T_LONG	high	64 位值中最有意义的 32 位。
TS_Value	TS_Union	u	所有已知数据类型值的值联合。
	T_SHORT	flag	值标志。 对于属性值，“flag”将是 15 (0x0F) in 如果该值存在 (不为空)，则为 0。 对于本地列的值：每个值有四个标志可用，每个标志用一个字节编码，每个标志用一个位。对于单个标志的访问，定义了位掩码的常量。支持以下值标志： ao_vf_valid (0x01) -值有效 ao_vf_visible (0x02) -必须可视化该值 ao_vf_未修改 (0x04) -值未修改 ao_vf_defined (0x08) -定义值 (基层也使用此标志来标记值矩阵中的间隙)
TS_ValueSeq	TS_UnionSeq	u	所有已知数据类型值的值联合。

	S_SHORT	flag	请参见 ts_value, 标记。

13.3.22 ASAM ODS OO-API 相关语境变量

ASAM ODS OO-API 允许在多个方面控制服务器的运行，主要通过一些相关语境变量实现，它们在一个会话中是有效的。其中有些是在创建一个 **AoFactory** 会话时设置的，另外一些可通过 **AoSession** 的特定方法设置和/或获取。

与相关语境变量对应的是 **RPC-API** 服务器端的参数。

表 253 出了目前 **OO-API** 相关语境变量集并做了简要说明。

表253 OO-API 的相关语境变量

名称	描述	缺省值
WILDCARD_ALL	零个或多个字符匹配的通配符	*
WILDCARD_ESC	用于 NextCharacter 匹配的通配符转义符	\
WILDCARD_ONE	一个字符匹配的通配符	.
USER	当前会话中 ASAM ODS 用户的用户名	
PASSWORD	当前会话中 asam ods 用户的密码	
ODSVERSION	ASAM ODS 服务器版本；只读	
write_mode	asam ods 定义的变量，它告诉服务器在哪里写入本地列的标志和值。如果将此变量设置为“file”，则服务器会在文件中写入显式通道的值。	database
EXT_COMP_SEGSIZE	以字节为单位定义新段的最大文件大小。当客户端更改此参数的值时，以下所有写入操作都必须使用新定义。此参数的默认值应在服务器启动时使用服务器端配置机制设置为适当的值	
CREATE_COSESSION_ALLOWED	指定是否可以通过创建会话的副本来克隆会话（“true”），或者是否不允许克隆会话（“false”）；必须在创建会话时设置；在会话中只读	FALSE
FILE_NOTATION	字符串值“unc-win”、“unc-unix”或“url”之一；描述指定外部文件位置的语法（具有某些语义）（有关详细信息，请参阅第 5.10.6 节）；只读	
FILE_MODE	字符串值“absolute”、“single_volume”或“multi_volume”之一；指定服务器管理外部文件位置的方式（有关详细信息，请参阅第 5.10.6 节）；只读	
FILE_SYMBOLS	提供 ODS 服务器已知的所有文件符号名称的列表（逗号分隔）字符串列表。文件符号指定外部文件的根路径（有关详细信息，请参阅第 5.10.6 节）；只读	
FILE_ROOT	指定外部组件文件的根路径（有关详细信息，请参阅第 5.10.6 节）；只读	
FILE_ROOT_EXTREF	指定外部引用文件的根路径（有关详细信息，请参阅第 5.10.6 节）；只读	

further file symbol names	字符串列表 givenby file_symbols 的每个文件符号名称本身就是一个上下文变量的名称；每个变量指定外部文件的根路径（请参阅第 5.10.6 节的详细信息）；只读	
VALUEMATRIX_MODE	值矩阵的默认模式。它可以设置为“计算”或“存储”。使用此变量可以更改值矩阵的默认模式。此变量应在会话中首次创建 ValueMatrix 之前设置；否则，服务器将显示其默认模式（可能是特定于服务器的模式）。创建第一个值矩阵后，会话中所有其他值矩阵都将具有相同的模式。	

13.4 编程示例

13.4.1 通过 CORBA 访问 ASAM ODS 工厂对象

任何解释或编译语言都可以调用 OO-API。该应用编程接口的设计是利用 CORBA IDL 实现的。因此通过 CORBA 使用 OO-API 是非常自然的，以下给出了基于 CORBA 使用 OO-API 的示例。

这种方式的优点是只需很少的努力就可以部署 CORBA，基于和不基于 CORBA 之间的唯一区别是获取对象 AoFactory 的方式。通过 CORBA 调用还是直接本地调用 OO-API 接口中的其它方法，两者之间没有区别---CORBA 对于编程人员来说是透明的。

对 COEBA 服务的需求仅限于 CORBA 名称服务，必须启动称作“NameService”的 CORBA 名称服务，之后，ODS 服务器可在名称服务中注册实现了 AoFactory 接口的对象。

然后任何客户端都可通过 CORBA 名称服务器查询已注册的 factory 对象，从而客户端收到 ASAM ODS Factory 对象（AoFactory）的句柄，利用该句柄可在 ODS 服务器中建立任意数量的会话。

示例虽然是用 Java 语言编写的，但与其它编程语言非常相似。在 Java 语言中，名称之间的句点被解释为目录名之间的分隔符。路径是相对于 Java 环境变量 CLASSPATH 进行解释的，另一个 CLASSPATH 意味着另一个实现（可基于或不基于 CORBA）。

该示例没有包含任何错误检查或者失效预防代码，在 CORBA 名称服务器中注册的对象名称可类似于目录结构中的文件名具有层次结构。为了简化该示例，factory 对象的名称为“serverName”。有关 CORBA 服务和层次结构名称空间的信息详见有关的 CORBA 文档（www.omg.org）。

该示例说明如何使用 CORBA 命名服务获取 AoFactory，以及如果通过 CORBA 访问 OO-API 的方法。

```
// Import ASAM ODS factory classes.
import org.asam.ods.AoFactory;
import org.asam.ods.AoFactoryHelper;
// Import CORBA Name Service and ORB classes.
import org.omg.CosNaming.NameComponent;
import org.omg.CosNaming.NamingContext;
import org.omg.CosNaming.NamingContextHelper;
```

```

import org.omg.CORBA.ORB;
// Create and initialize the ORB.
ORB orb = ORB.init (args, null) ;
// Get the root naming context.
org.omg.CORBA.Object objRef =
orb.resolve_initial_references ("NameService") ;
NamingContext ncRef = NamingContextHelper.narrow (objRef) ;
// Resolve the Object Reference in Naming.
NameComponent nc = new NameComponent ("serverName", "serverType") ;
NameComponent path[] = {nc};
AoFactory aoFactory = AoFactoryHelper.narrow (ncRef.resolve (path)) ;

```

13.5 OO-API 定义文件

以下所列为 **ASAM ODS** 定义文件，这是 **OO-API** 的定义文件 **ODS.IDL**。它利用 **CORBA IDL**（接口定义语言）描述该接口。定义文件中的所有模块和函数的说明（目的、调用序列、返回、示例等）详见 13.2 节。

13.5.1 ODS.IDL

```

// *****
// * ASAM ODS 5.2.0 Interface Definition *
// *****
//
// Generated: Fri Sep 25 14:43:10 CEST 2009
// by: HighQSoft GmbH www.highqsoft.com
//
// 2003-12-31
// ODS Version 5.0
//
// 2005-05-02
// Method getRelationByBaseName of the interface ApplicationElement added.
// Method getInverseRelationName of instance BaseRelation added. Better
// explanation in the comments. Removed unused error codes of the exceptions.
//
// 2005-07-11
// More explanation for the method getInstancesEx of the interface
// ApplElemAccess. More details for the method setRelationType of the
// interface ApplicationRelation.
//

```

```
// 2005-09-26
// Add SelOpcode IS_NULL and IS_NOT_NULL. Add exception AO_INVALID_DATATYPE
// to the method setEnumerationDefinition of the interface
ApplicationElement.
// Method getEnumerationDefinition added to the Interface BaseRelation.
Add
// comment to methods getFormula and setFormula of interface Column.
//
// 2005-12-05
// Add the variable CREATE_COSESSION_ALLOWED at the method newSession at
the
// interface AoFactory. Add the methods createCoSession and getUser at the
// interface AoSession. Add values of localcolumn at the methods
// getInstances, getInstancesExt and updateInstances of the interface
// ApplElemAccess. Comment of method setValue of the interface ValueMatrix
// reworked.
//
// 2006-06-26
// Correct the Method getRelationByBaseName of the interface
// ApplicationElement to the Method getRelationsByBaseName, mistake in
// Interface definition there are at some elements relations derived from
the
// same base relation allowed.
//
// 2007-03-23
// Add, SUM to the AggrFunc. Change comment of method
getApplicationStructure
// and getBaseStructure of the interface AoSession, the method
// getApplicationStructure of the interface ApplicationElement.and the
method
// getRelation of the interface BaseStructure. Add version string
description
// at method getInterfaceVersion of interface AoFactory.
//
// 2007-06-22
// Add the Enumeration ValueMatrixMode. Change the comment of the Methods
// 13001, 13004, 13006, 14018, 14022, 16003, 21002, 24007 Add the exception
```

```
// AO_INVALID_VALUEMATRIX_STRUCTURE for the method getValueMatrix of the
// interface Measurement and ApplElemAccess. Add the new method
// getValueMatrixInMode of the interfaces Measurement (16005), SubMatrix
// (21004) and ApplElemAccess Add the methods getSequenceRepresentation
// (13013), setSequenceRepresentation (13014), getGenerationParameters
// (13015), setGenerationParameters (13016) and getRawDataType (13017)
of the
// interface Column. Add the method getMode (22022) of the interface
// ValueMatrix. Little changes in description of the methods, specify which
// object is modified, with the numbers 4008, 4009, 4010, 4011, 4012, 4013,
// 4014, 4019, 5001, 5002, 5020, 5021, 5022, 5023, 6010, 6011, 6012, 6013,
// 6014, 6015, 6016, 6018, 7002, 7003, 7014, 7015, 12001, 12006, 12007,
13005,
// 14001, 14002, 14014, 14016, 14017, 22014, 22019. Correct typing errors
in
// method 24005
//
// 2007-08-27
// Correct the enumeration ValueMatrixMode. Correct typing errors at method
// 13014. Extend explanation at 13015 and 13016
//
// 2007-09-18
// Typing corrections in IDL: Method 24005 comment changed. Change the word
// 'calibrated' into 'recalculated' in method 4014 setUnit.
//
// 2007-11-08
// Set the methods hasValueFlag and withValueFlag of the interface
// ApplicationAttribute to deprecated. Change description of
createAttribute
// of interface ApplicationElement and ErrorCode AO_MISSING_VALUE.
//
// 2009-05-25
// Add the interfaces ODSDirectory, ODSFile and ODSFileTransfer and
// Enumeration FileOpenMode for the managed files. Add the methods
// getODSDirectory in the interfaces ApplElemAccess and InstanceElement.
//
// 2009-05-28
```

```

// Add the methods getEnumerationAttributes and getEnumerationStructure
of the
// interface AoSession with the required structures and sequences.
//
// 2009-06-18
// Correct typing error's.
//
// 2009-06-29
// Add AggrFunc DISTINCT, SelOpcode NOTLIKE and CI_NOTLIKE
//
// 2009-06-30
// Remove ODSDirectory, change the way ODSFile is created. Change
// FileOpenMode.
//
// 2009-08-27
// Change the description of the method getUnit of the interface Column.
// Change the description of the methods of the interfaces ODSFile and
// ODSFileTransfer. Change the description of the enumerations
// ValueMatrix_Mode and FileOpenMode. Change the description of the
structure
// ApplRel.
//
// 2009-09-14
// Remove Interface ODSFile and ODSFileTransfer and Enumeration
FileOpneMode
// with the methods which creates the ODSFile interface.
//
// 2009-09-25
// Correct of some interfacaes and methods the description. Finalize for
// ASAM ODS version 5.2.0
//
//
#ifndef ods_idl
#define ods_idl
module org {
/**
* ASAM services.

```



```

*/
module asam {
/**
 * ASAM ODS service.
 */
module ods {
interface AoFactory;
interface AoSession;
interface ApplicationAttribute;
interface ApplicationElement;
interface ApplicationRelation;
interface ApplicationStructure;
interface BaseAttribute;
interface BaseElement;
interface BaseRelation;
interface BaseStructure;
interface Blob;
interface Column;
interface InstanceElement;
interface InstanceElementIterator;
interface Measurement;
interface NameIterator;
interface NameValueIterator;
interface NameValueUnitIterator;
interface SMatLink;
interface SubMatrix;
interface ValueMatrix;
interface NameValueUnitIdIterator;
interface ApplElemAccess;
interface QueryEvaluator;
interface Query;
interface NameValueUnitSequenceIterator;
interface EnumerationDefinition;
interface ElemResultSetExtSeqIterator;
/**
 * The ASAM ODS error severity flags.
 */

```

```

enum SeverityFlag {
    SUCCESS, // Ok.
    INFORMATION, // Information.
    WARNING, // Warning.
    ERROR // Error.
};

/**
 * The ASAM ODS relation types.
 */
enum RelationType {
    FATHER_CHILD, // Father-child relation.
    INFO, // Info relation.
    INHERITANCE // Inheritance relation.
};

/**
 * The ASAM ODS relationships.
 */
enum Relationship {
    FATHER, // Father.
    CHILD, // Child.
    INFO_TO, // Directed informational relationship.
    INFO_FROM, // Directed informational relationship.
    INFO_REL, // Informational relationship (no direction)
    SUPERTYPE, // Inheritance relationship: supertype.
    SUBTYPE, // Inheritance relationship: subtype.
    ALL_REL // Any of the relationships above.
};

/**
 * The ASAM ODS data types.
 * DT_xxx Basic data types.
 * DS_xxx Sequence of basic data type.
 * ||
 * |+- T == Type, S == Sequences.
 * +-- D == Datatype.
 */
enum DataType {
    DT_UNKNOWN, // Unknown datatype.

```

DT_STRING, // String.
 DT_SHORT, // Short value (16 bit) .
 DT_FLOAT, // Float value (32 bit) .
 DT_BOOLEAN, // Boolean value.
 DT_BYTE, // Byte value (8 bit) .
 DT_LONG, // Long value (32 bit) .
 DT_DOUBLE, // Double precision float value (64 bit) .
 DT_LONGLONG, // LongLong value (64 bit) .
 DT_ID, // LongLong value (64 bit) . Not used. DT_LONGLONG is
 // used instead.
 DT_DATE, // Date. Meaning: YYYYMMDDhhmmsslllcccnnn.... -// 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 = microse,
 // optional, not supported by Oracle timestamp. -// nnn = nanosec, optional,
 not supported by Oracle
 // timestamp.
 DT_BYTESTR, // Bytestream.
 DT_BLOB, // Blob.
 DT_COMPLEX, // Complex value (32 bit each part) .
 DT_DCOMPLEX, // Complex value (64 bit each part) .
 DS_STRING, // String sequence.
 DS_SHORT, // Short sequence.
 DS_FLOAT, // Float sequence.
 DS_BOOLEAN, // Boolean sequene.
 DS_BYTE, // Byte sequence.
 DS_LONG, // Long sequence.
 DS_DOUBLE, // Double sequence.
 DS_LONGLONG, // Longlong sequence.
 DS_COMPLEX, // Complex sequence.
 DS_DCOMPLEX, // Double complex sequence.
 DS_ID, // LongLong sequence. Not used. DS_LONGLONG is used
 // instead.
 DS_DATE, // Date sequence.
 DS_BYTESTR, // Bytestream sequence.

```

DT_EXTERNALREFERENCE, // External reference.
DS_EXTERNALREFERENCE, // Sequence of external reference.
DT_ENUM, // The enumeration datatype.
DS_ENUM // The enumeration sequence datatype.
};
/**
 * The ASAM ODS build-up function codes for measurement views.
 */
enum BuildUpFunction {
  BUP_JOIN, // Join the columns
  BUP_MERGE, // Merge the columns
  BUP_SORT // Sort the columns
};
/**
 * The ASAM ODS attribute type codes.
 */
enum AttrType {
  APPLATTR_ONLY, // Report only application attributes.
  INSTATTR_ONLY, // Report only instance attributes.
  ALL // All attributes.
};
/**
 * The ASAM ODS types for setting values.
 */
enum SetType {
  APPEND, // Append data to the value matrix.
  INSERT, // Insert data into the value matrix.
  UPDATE, // Modify existing data of the value matrix.
  REMOVE // Remove the given information.
};
/**
 * The ASAM ODS error codes.
 */
enum ErrorCode {
  AO_UNKNOWN_ERROR,
  AO_ACCESS_DENIED,
  AO_BAD_OPERATION,

```

AO_BAD_PARAMETER,
AO_CONNECT_FAILED,
AO_CONNECT_REFUSED,
AO_CONNECTION_LOST,
AO_DUPLICATE_BASE_ATTRIBUTE,
AO_DUPLICATE_NAME,
AO_DUPLICATE_VALUE,
AO_HAS_INSTANCES,
AO_HAS_REFERENCES,
AO_IMPLEMENTATION_PROBLEM,
AO_INCOMPATIBLE_UNITS,
AO_INVALID_ASAM_PATH,
AO_INVALID_ATTRIBUTE_TYPE,
AO_INVALID_BASE_ELEMENT,
AO_INVALID_BASETYPE,
AO_INVALID_BUILDUP_FUNCTION,
AO_INVALID_COLUMN,
AO_INVALID_COUNT,
AO_INVALID_DATATYPE,
AO_INVALID_ELEMENT,
AO_INVALID_LENGTH,
AO_INVALID_ORDINALNUMBER,
AO_INVALID_RELATION,
AO_INVALID_RELATION_RANGE,
AO_INVALID_RELATION_TYPE,
AO_INVALID_RELATIONSHIP,
AO_INVALID_SET_TYPE,
AO_INVALID_SMATLINK,
AO_INVALID_SUBMATRIX,
AO_IS_BASE_ATTRIBUTE,
AO_IS_BASE_RELATION,
AO_IS_MEASUREMENT_MATRIX,
AO_MATH_ERROR,
AO_MISSING_APPLICATION_ELEMENT,
AO_MISSING_ATTRIBUTE,
AO_MISSING_RELATION,
AO_MISSING_VALUE,

```

AO_NO_MEMORY,
AO_NO_PATH_TO_ELEMENT,
AO_NOT_FOUND,
AO_NOT_IMPLEMENTED,
AO_NOT_UNIQUE,
AO_OPEN_MODE_NOT_SUPPORTED,
AO_SESSION_LIMIT_REACHED,
AO_SESSION_NOT_ACTIVE,
AO_TRANSACTION_ALREADY_ACTIVE,
AO_TRANSACTION_NOT_ACTIVE,
AO_HAS_BASE_RELATION,
AO_HAS_BASE_ATTRIBUTE,
AO_UNKNOWN_UNIT,
AO_NO_SCALING_COLUMN,
AO_QUERY_TYPE_INVALID,
AO_QUERY_INVALID,
AO_QUERY_PROCESSING_ERROR,
AO_QUERY_TIMEOUT_EXCEEDED,
AO_QUERY_INCOMPLETE,
AO_QUERY_INVALID_RESULTTYPE,
AO_INVALID_VALUEMATRIX_STRUCTURE

```

```
};
```

```
/**
```

```

* The selection operators. SelOpcode gives query instructions like
* "equal", "greater" etc. So far, these arguments were case sensitive.
* There was a demand to add these arguments also for case insensitive
* comparison operations. Therefore, the SelOpCodes for case insensitivity
* were added. These arguments have the prefix "CI_".

```

```
*/
```

```

enum SelOpcode {
EQ, // Equal
NEQ, // Not equal
LT, // Less than
GT, // Greater than
LTE, // Less than equal
GTE, // Greater than equal
INSET, // In set, value can be a sequence.

```

```

NOTINSET, // Not in set, value can be a sequence.
LIKE, // like, use pattern matching, see Pattern for the wildcard
// definitions.
CI_EQ, // Equal. case insensitive for DT_STRING.
CI_NEQ, // Not equal. case insensitive for DT_STRING.
CI_LT, // Less then. case insensitive for DT_STRING.
CI_GT, // Greater then. case insensitive for DT_STRING.
CI_LTE, // Less then equal. case insensitive for DT_STRING.
CI_GTE, // Greater then equal. case insensitive for DT_STRING.
CI_INSET, // In set, value can be a sequence. case insensitive for
// DT_STRING.
CI_NOTINSET, // Not in set, value can be a sequence. case insensitive for
// DT_STRING.
CI_LIKE, // like, use pattern matching, see Pattern for the wildcard
// definitions. case insensitive for DT_STRING.
IS_NULL, // Value is NULL
IS_NOT_NULL, // Value is not NULL
NOTLIKE, // Not LIKE
CI_NOTLIKE // Not LIKE, case insensitive for DT_STRING.
};
/**
 * Operator, bracket open and close.
 */
enum SelOperator {
AND, // AND the two conditions.
OR, // OR the two conditions.
NOT, // Negate the next condition.
OPEN, // Open brackets.
CLOSE // Close brackets
};
/**
 * The ASAM ODS types for setting access rights.
 */
enum RightsSet {
SET_RIGHT, // Set the given rights, overwrite the existing rights.
ADD_RIGHT, // Add the given rights to the existing rights.
REMOVE_RIGHT // Remove the given rights form the existing rights.

```

```

};
/**
 * Status of the query execution.
 */
enum QueryStatus {
COMPLETE, // The execution is ready.
INCOMPLETE // The execution is still running.
};
/**
 * The supported aggregate functions of the GetInstancesExt.
 */
enum AggrFunc {
NONE, // No aggregate function is used for attribute.
COUNT, // Count
DCOUNT, // Distinct count
MIN, // Min; only for numerical values
MAX, // Max; only for numerical values
AVG, // Average; only for numerical values
STDDEV, // Standard deviation; only for numerical values
SUM, // Sum; only for numerical values
DISTINCT // Distinct
};
/**
 * The selection type.
 */
enum SelType {
SEL_VALUE_TYPE, // Selection value.
SEL_OPERATOR_TYPE // Selection logical operator.
};
/**
 * The type of the join.
 */
enum JoinType {
JTDEFAULT, // Force inner join.
JTOUTER // Force outer join on destination AID.
};
/**

```



```

* The mode of the value matrix.
*/
enum ValueMatrixMode {
CALCULATED, // The value matrix returns the calculated values. The values
// of explicit columns are returned as they are stored in the
// physical storage. The values of implicit columns are
// expanded and returned explicitly. The values of raw columns
// are calculated according to the corresponding algorithm
// (using the calculation algorithms given by the values of the
// application attributes derived from the base attributes
// 'sequence_representation' and 'generation_parameters') and
// the resulting physical values are returned. Writing to the
// value matrix is not allowed in this mode.
STORAGE // The value matrix returns the values given in the physical
// storage. The values of explicit columns are returned as
// they are stored in the physical storage. The values of
// implicit columns are expanded and returned explicitly. The
// values of raw columns are returned as they are stored
// (without using the calculation algorithms given by the
// values of the application attributes derived from the base
// attributes 'sequence_representation' and
// 'generation_parameters'). Writing to value matrix is
// allowed in this mode.
};

// Datatype definitions (T_xxx) .
typedef string T_STRING;
typedef boolean T_BOOLEAN;
typedef short T_SHORT;
typedef float T_FLOAT;
typedef octet T_BYTE;
typedef long T_LONG;
typedef double T_DOUBLE;
typedef T_STRING Name;
typedef T_STRING Pattern;
typedef T_STRING BaseType;
typedef T_STRING T_DATE;
typedef Blob T_BLOB;

```

```

// Sequence definitions (S_xxx) .
typedef sequence<T_BYTE> T_BYTESTR;
typedef sequence<T_BOOLEAN> S_BOOLEAN;
typedef sequence<T_BYTE> S_BYTE;
typedef sequence<T_DOUBLE> S_DOUBLE;
typedef sequence<T_FLOAT> S_FLOAT;
typedef sequence<T_LONG> S_LONG;
typedef sequence<T_SHORT> S_SHORT;
typedef sequence<T_STRING> S_STRING;
typedef sequence<BaseType> BaseTypeSequence;
typedef sequence<Name> NameSequence;
typedef sequence<Column> ColumnSequence;
typedef sequence<SMatLink> SMatLinkSequence;
typedef sequence<SubMatrix> SubMatrixSequence;
typedef sequence<T_DATE> S_DATE;
typedef sequence<T_BYTESTR> S_BYTESTR;
typedef sequence<S_STRING> SS_STRING;
typedef sequence<S_SHORT> SS_SHORT;
typedef sequence<S_FLOAT> SS_FLOAT;
typedef sequence<S_BOOLEAN> SS_BOOLEAN;
typedef sequence<S_BYTE> SS_BYTE;
typedef sequence<S_LONG> SS_LONG;
typedef sequence<S_DOUBLE> SS_DOUBLE;
typedef sequence<S_DATE> SS_DATE;
typedef sequence<S_BYTESTR> SS_BYTESTR;
typedef sequence<T_BLOB> S_BLOB;
typedef sequence<ApplicationElement> ApplicationElementSequence;
typedef sequence<ApplicationRelation> ApplicationRelationSequence;
typedef sequence<ApplicationAttribute> ApplicationAttributeSequence;
typedef sequence<BaseRelation> BaseRelationSequence;
typedef sequence<BaseAttribute> BaseAttributeSequence;
typedef sequence<BaseElement> BaseElementSequence;
typedef sequence<InstanceElement> InstanceElementSequence;
typedef sequence<SelOperator> SelOperatorSequence;
/**
 * The ASAM ODS relation range structure.
 */

```

```

struct RelationRange {
T_SHORT min; // The minimum number in the range.
T_SHORT max; // The maximum number in the range. -1 means MANY without a
// specified maximum number.
};
/**
 * The ASAM ODS 64 bit integer structure. This type is represented in the
 * datatype enumeration by DT_LONGLONG.
 */
struct T_LONGLONG {
T_LONG high; // The most significant 32 bits of the 64 bit value.
T_LONG low; // The least significant 32 bits of the 64 bit value.
};
/**
 * The ASAM ODS complex data structure. This type is represented in the
 * datatype enumeration by DT_COMPLEX.
 */
struct T_COMPLEX {
T_FLOAT r; // The real part of the complex number.
T_FLOAT i; // The imaginary part of the complex number.
};
/**
 * The ASAM ODS double-precision complex data structure. This type is
 * represented in the datatype enumeration by DT_DCOMPLEX.
 */
struct T_DCOMPLEX {
T_DOUBLE r; // The real part of the double precision complex number.
T_DOUBLE i; // The imaginary part of the double precision complex number.
};
/**
 * The ASAM ODS name-unit tuple structure.
 */
struct NameUnit {
Name valName; // Attribute name or measured quantity name.
T_STRING unit; // Column unit as string.
};
/**

```

* The description of 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.

*/

```
struct T_ExternalReference {
T_STRING description; // Description of the external reference.
T_STRING mimeType; // MIME-type of the external object.
T_STRING location; // Location of the external reference. (asam path or
// URL)
};
```

/**

* The application attribute information (metadata) definition. The same
 * information is available at the interface ApplicationAttribute
 */

```
struct ApplAttr {
Name aaName; // The application attribute name. The same name
// is returned by the method getName () of the
// ApplicationAttribute interface. At the RPC
// API this information was stored in the field
// aaName of the structure AttrSeq and the request
// AOP_GetAttr.
Name baName; // The name of the base attribute, empty ("" ) if
// the application attribute is not derived from a
// base attribute. The same name is returned by
// the methods getName () of the BaseAttribute
// interface. The base attribute is given by the
// the method getBaseAttribute () of the interface
// ApplicationAttribute. At the RPC API this
// information was stored in the field aBName of
// the structure AttrSeq and the request
// AOP_GetAttr.
```

```
DataType dType; // The attribute data type. The same data type is
// given by the method getDataType of the
// interface ApplicationAttribute. At the RPC
// API this information was stored in the field
// aDataType of the structure AttrSeq and the
// request AOP_GetAttr.
```

```

T_LONG length; // The maximum possible length of values. The same
// length is returned by the method getLength () of
// the interface ApplicationAttribute.
T_BOOLEAN isObligatory; // The indicator for mandatory attributes, the
// notNull indicator is set at the column of the
// table in the physical storage. The same
// boolean is returned at the method
// isObligatory () of the interface
// ApplicationAttribute.
T_BOOLEAN isUnique; // The indicator for unique attributes. The same
// boolean is returned by the method isUnique () of
// the interface ApplicationAttribute.
T_LONGLONG unitId; // Id of the unit if global defined. The same Id
// is returned by the method getUnit () of the
// interface ApplicationAttribute. At the RPC
// API this information was stored in the field
// aUnit of the structure AttrSeq and the request
// AOP_GetAttr.
};
/**
 * The application relation info structure. The same information is
 * available at the interface ApplicationRelation.
 *
 * A relation is the connection between two ASAM ODS elements; it may be
 * navigated in both
 * directions. The get- and set- methods are defined from the first
 * element, the getInverse- and
 * setInverse- methods work from the second element.
 */
struct ApplRel {
T_LONGLONG elem1; // The source application element Id. The
// given Id is the Id of the application
// element returned from the method
// getElem1 () of the interface
// ApplicationRelation. At the RPC API
// this information was stored in the field
// arAid1 of the structure ApplRelSeq and

```

```

// the request AOP_GetApplInf.
T_LONGLONG elem2; // The target application element Id. The
// given Id is the Id of the application
// element returned from the method
// getElem2 () of the interface
// ApplicationRelation. At the RPC API
// this information was stored in the field
// arAid2 of the structure ApplRelSeq and
// the request AOP_GetApplInf.
Name arName; // The relation name. The name is 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.
Name invName; // Name of the inverse relation. The name
// is return with the method
// getInverseName () of the interface
// ApplicationRelation. The invName is not
// available in the physical storage for
// relation databases.
Name brName; // Name of the base relation from the elem1
// to the elem2. The base relation is also
// not stored in the physical storage.
Name invBrName; // Name of the inverse base relation from
// the elem2 to the elem1. The base
// relation is also not stored in the
// physical storage.
RelationType arRelationType; // The type of the relation. Type of the
// relation is not stored in the physical
// storage. The relation type is return at
// the method getRelationType () of the
// interface ApplicationRelation.
RelationRange arRelationRange; // The range of the relation. Range of the
// relation is not stored in the physical
// storage. The relation range is return at

```

```

// the method getRelationRange () of the
// interface ApplicationRelation.
RelationRange invRelationRange; // The inverse range of the relation. Range
// of the relation is not stored in the
// physical storage. The inverse relation
// range is return at the method
// getInverseRelationRange () of the
// interface ApplicationRelation.
};
/**
 * AID - Name pair.
 */
struct AIDName {
T_LONGLONG aid; // The Id of the application element.
Name aaName; // The attribute, or measured quantity name.
};
/**
 * Instance element Id. The unique description of an instance element.
 */
struct ElemId {
T_LONGLONG aid; // The Id of the application element.
T_LONGLONG iid; // The Id of the instance element.
};
/**
 * AID - Name - UnitId tuple.
 */
struct AIDNameUnitId {
AIDName attr; // The attribute of the application element (aid,
// name) .
T_LONGLONG unitId; // The unit of the attribute or the column. The unitId
// is the Id of instance element with the basetype
// AoUnit.
};
/**
 * Order criteria.
 */
struct SelOrder {

```

```

AIDName attr; // Attribute specification.
T_BOOLEAN ascending; // ascending order, FALSE means descending.
};
/**
 * The access control list entry.
 */
struct ACL {
T_LONGLONG usergroupId; // The usergroup Id.
T_LONG rights; // The access rights of the requested object.
};
/**
 * An initial right.
 */
struct InitialRight {
T_LONG rights; // The initial access rights of the requested
// object.
T_LONGLONG usergroupId; // The usergroup Id of the Initial right list.
T_LONGLONG refAid; // The referencing application element.
};
/**
 * it's quite the same sequence as in the QueryStructure of GetInstances
 * with one exception. It has one new attribute called function, which is
 * of the type AggrFunc. Thereby it is possible to define aggregate
 * functions on attribute level, without the need to parse the attribute
 * name for a known aggregate function name. The default value of that
 * attribute function is NONE, it symbolizes that no aggregate function
 * should be applied on that attribute. If an aggregate function is used,
 * it is also required to define a GroupSequence. It is also defined that
 * a
 * '*' as attribute name, delivers all attributes of an element.
 */
struct SelAIDNameUnitId {
AIDName attr; // The attribute of the application element (aid,
// name) .
T_LONGLONG unitId; // The unit of the attribute of the column. The
// unitId is the Id of instance element with the
// basetype AoUnit.

```



```

AggrFunc aggregate; // The aggregate function.
};
/**
 * Basically, joins can only be realized between application elements that
 * are linked via a reference defined in the model. From the definition of
 * attributes or application elements, the references for the joins are
 * determined. It is also taken into account that the application elements
 * involved are not linked directly. However, there must be an unambiguous
 * path between the application elements. The path may also include n:m
 * relations. The unambiguousness of relations between two application
 * elements no longer exists if more than one reference has been defined
 * between the application elements. In this case, these references must
 * have names and must be indicated explicitly in the request. For this
 * purpose, the request structure provides a sequence of relation
 * definitions (JoinDefSequence). The sequence in which the application
 * elements are addressed in the request also determines the sequence in
 * which the references between application elements are searched. Thus,
 * for every new application element, the server begins with the first
 * application element addressed in the request and tries to find a
 * relation from there. If no reference to the first application element
 * can be found, the search continues with the application element that
 * comes next in the request. Furthermore, the explicit relation
 * definitions (JoinDefSequence) enable an OUTER join, i.e. the result
 * also includes those records for which the join could not be
 * established.
 */
struct JoinDef {
    T_LONGLONG fromAID;
    T_LONGLONG toAID;
    Name refName;
    JoinType joiningType;
};
/**
 * The application relation with the instances to create the relation with
 * new instances.
 */
struct ApplicationRelationInstanceElementSeq {

```

```

ApplicationRelation applRel; // The application relation.
InstanceElementSequence instances; // The list with instances. The
// application element of the instances
// in the list must match one of the
// application elements of the
// application relation. All instances
// of the list must have the same
// application element.
};
/**
 * The structure with the attribute and the name of the enumeration.
 */
struct EnumerationAttributeStructure {
T_LONGLONG aid; // The application element Id.
T_STRING aaName; // The name of the attribute
T_STRING enumName; // The name of the enumeration
};
/**
 * The structure with the items of the enumeration.
 */
struct EnumerationItemStructure {
T_LONG index; // The index of the enumeration item.
T_STRING itemName; // The name of the item.
};
// Sequence definitions (S_xxx) .
typedef                      sequence<EnumerationAttributeStructure>
EnumerationAttributeStructureSequence;
typedef                      sequence<EnumerationItemStructure>
EnumerationItemStructureSequence;
typedef sequence<T_COMPLEX> S_COMPLEX;
typedef sequence<T_DCOMPLEX> S_DCOMPLEX;
typedef sequence<T_LONGLONG> S_LONGLONG;
typedef sequence<S_LONGLONG> SS_LONGLONG;
typedef sequence<S_COMPLEX> SS_COMPLEX;
typedef sequence<S_DCOMPLEX> SS_DCOMPLEX;
typedef sequence<T_ExternalReference> S_ExternalReference;
typedef sequence<S_ExternalReference> SS_ExternalReference;

```

```

typedef sequence<ApplAttr> ApplAttrSequence;
typedef sequence<ApplRel> ApplRelSequence;
typedef sequence<AIDName> AIDNameSequence;
typedef sequence<AIDNameUnitId> AIDNameUnitIdSequence;
typedef sequence<ElemId> ElemIdSequence;
typedef sequence<SelOrder> SelOrderSequence;
typedef sequence<ACL> ACLSequence;
typedef sequence<InitialRight> InitialRightSequence;
typedef sequence<SelAIDNameUnitId> SelAIDNameUnitIdSequence;
typedef sequence<JoinDef> JoinDefSequence;
typedef sequence<ApplicationRelationInstanceElementSeq>
ApplicationRelationInstanceElementSeqSequence;
/**
 * The application element definition. The same information is available
 * at the interface ApplicationElement.
 */
struct ApplElem {
T_LONGLONG aid; // The application element id. The id is given
// also with the method getId () at the
// interface ApplicationElement. At the
// RPC API this information was stored in the
// field aiAId of the structure ApplInfSeq and
// the request AOP_GetApplInf.
Name beName; // The base element name, all elements have a
// basic element. The same name is returned by
// the methods getType () of the BaseElement
// interface. The base element is given with
// the method getBaseElement () at the
// interface ApplicationElement. At the
// 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.
Name aeName; // The application element name. The name is
// given also with the method getName () at the
// interface ApplicationElement. At the

```

```

// RPC API this information was stored in the
// field aiName of the structure ApplInfSeq
// and the request AOP_GetApplInf.
ApplAttrSequence attributes; // The attributes of application element. The
// attributes are given with the method
// getAttributes () of the interface
// ApplicationElement. There are no relations
// given in the this sequence.
};
/**
 * The structure with the enumeration and there items.
 */
struct EnumerationStructure {
T_STRING enumName; // The name of the enueration.
EnumerationItemStructureSequence items; // The items of the enumeration.
};
/**
 * The union definition for all datatypes.
 */
union TS_Union switch (DataType) {
case DT_STRING: T_STRING stringVal;
case DT_SHORT: T_SHORT shortVal;
case DT_FLOAT: T_FLOAT floatVal;
case DT_BYTE: T_BYTE byteVal;
case DT_BOOLEAN: T_BOOLEAN booleanVal;
case DT_LONG: T_LONG longVal;
case DT_DOUBLE: T_DOUBLE doubleVal;
case DT_LONGLONG: T_LONGLONG longlongVal;
case DT_COMPLEX: T_COMPLEX complexVal;
case DT_DCOMPLEX: T_DCOMPLEX dcomplexVal;
case DT_DATE: T_DATE dateVal;
case DT_BYTESTR: T_BYTESTR bytestrVal;
case DT_BLOB: T_BLOB blobVal;
case DS_STRING: S_STRING stringSeq;
case DS_SHORT: S_SHORT shortSeq;
case DS_FLOAT: S_FLOAT floatSeq;
case DS_BYTE: S_BYTE byteSeq;

```

```

case DS_BOOLEAN: S_BOOLEAN booleanSeq;
case DS_LONG: S_LONG longSeq;
case DS_DOUBLE: S_DOUBLE doubleSeq;
case DS_LONGLONG: S_LONGLONG longlongSeq;
case DS_COMPLEX: S_COMPLEX complexSeq;
case DS_DCOMPLEX: S_DCOMPLEX dcomplexSeq;
case DS_DATE: S_DATE dateSeq;
case DS_BYTESTR: S_BYTESTR bytestrSeq;
case DT_EXTERNALREFERENCE: T_ExternalReference extRefVal;
case DS_EXTERNALREFERENCE: S_ExternalReference extRefSeq;
case DT_ENUM: T_LONG enumVal;
case DS_ENUM: S_LONG enumSeq;
};
/**
 * Define a union with sequences of a certain type. Using this union
 * instead of sequence <TS_Union> gives much better performance.
 */
union TS_UnionSeq switch (DataType) {
case DT_STRING: S_STRING stringVal;
case DT_SHORT: S_SHORT shortVal;
case DT_FLOAT: S_FLOAT floatVal;
case DT_BYTE: S_BYTE byteVal;
case DT_BOOLEAN: S_BOOLEAN booleanVal;
case DT_LONG: S_LONG longVal;
case DT_DOUBLE: S_DOUBLE doubleVal;
case DT_LONGLONG: S_LONGLONG longlongVal;
case DT_COMPLEX: S_COMPLEX complexVal;
case DT_DCOMPLEX: S_DCOMPLEX dcomplexVal;
case DT_DATE: S_DATE dateVal;
case DT_BYTESTR: S_BYTESTR bytestrVal;
case DT_BLOB: S_BLOB blobVal;
case DS_STRING: SS_STRING stringSeq;
case DS_SHORT: SS_SHORT shortSeq;
case DS_FLOAT: SS_FLOAT floatSeq;
case DS_BYTE: SS_BYTE byteSeq;
case DS_BOOLEAN: SS_BOOLEAN booleanSeq;
case DS_LONG: SS_LONG longSeq;

```

```

case DS_DOUBLE: SS_DOUBLE doubleSeq;
case DS_LONGLONG: SS_LONGLONG longlongSeq;
case DS_COMPLEX: SS_COMPLEX complexSeq;
case DS_DCOMPLEX: SS_DCOMPLEX dcomplexSeq;
case DS_DATE: SS_DATE dateSeq;
case DS_BYTESTR: SS_BYTESTR bytestrSeq;
case DT_EXTERNALREFERENCE: S_ExternalReference extRefVal;
case DS_EXTERNALREFERENCE: SS_ExternalReference extRefSeq;
case DT_ENUM: S_LONG enumVal;
case DS_ENUM: SS_LONG enumSeq;
};

// Sequence definitions (S_xxx) .
typedef sequence<EnumerationStructure> EnumerationStructureSequence;
typedef sequence<ApplElem> ApplElemSequence;
/**
 * The ASAM ODS value structure. There is one flag for each value. If the
 * union (u) contains a sequence, the flag is valid for all values in that
 * sequence.
 *
 * Meaning of flags:
 * AO_VF_VALID (0x01) The value is valid.
 * AO_VF_VISIBLE (0x02) The value has to be
 * visualized.
 * AO_VF_UNMODIFIED (0x04) The value has not been
 * modified.
 * AO_VF_DEFINED (0x08) The value is defined. If
 * the value in a value matrix
 * is not available this bit
 * is not set.
 * The normal value of the flag is 15.
 */
struct TS_Value {
    TS_Union u; // The value union for values of all known datatypes.
    T_SHORT flag; // The value flags.
};
/**
 * A structure with sequences of a certain type. Using this union instead

```

```

* of sequence <TS_Value> gives much better performance.
*/
struct TS_ValueSeq {
    TS_UnionSeq u; // The value union for values of all known datatypes.
    S_SHORT flag; // See TS_Value flag.
};
/**
 * Application 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.
 */
struct ApplicationStructureValue {
    ApplElemSequence applElems; // The list of application elements.
    ApplRelSequence applRels; // 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.
};
/**
 * The ASAM ODS name-value-unit tuple structure with a sequence of values.
 */
struct NameValueSeqUnit {
    Name valName; // Column name or measured quantity name.
    TS_ValueSeq value; // Column value (vector) .
    T_STRING unit; // Column unit as string.
};
/**
 * AID - Name - Value - UnitId quartet.
 */
struct AIDNameValueUnitId {
    AIDName attr; // The attribute of the application element (aid,

```

```

// name) .
T_LONGLONG unitId; // The unit of the attribute of the column. The unitId
// is the Id of instance element with the basetype
// AoUnit.
TS_Value values; // The attribute values with value flags.
};
/**
 * AID - Name - Value - UnitId quartet. Multiple values for one attribute.
 */
struct AIDNameValueSeqUnitId {
    AIDName attr; // The attribute of the application element (aid,
// name) .
    T_LONGLONG unitId; // The unit of the attribute of the column. The unitId
// is the Id of instance element with the basetype
// AoUnit.
    TS_ValueSeq values; // The column values with value flags.
};
/**
 * The ASAM ODS name-value-unitId tuple structure with a sequence of
 * values.
 */
struct NameValueSeqUnitId {
    Name valName; // Column name or measured quantity name.
    TS_ValueSeq value; // Column value (vector) .
    T_LONGLONG unitId; // Column unit as Id.
};
/**
 * Structure for name value query or attribute search conditions.
 */
struct SelValue {
    AIDNameValueUnitId attr; // The attribute specification with unit of the
// value.
    SelOpcode oper; // The compare operator between the attribute and
// value.
    TS_Value value; // Value for the condition.
};
/**

```



```

* The ASAM ODS name-value-unitid tuple structure. This structure is
* identical with the NameValueUnit, except the unit is given as an Id
* instead of a string.
*/
struct NameValueUnitId {
    Name valName; // Attribute name or measured quantity name.
    TS_Value value; // Attribute value or column value (vector) .
    T_LONGLONG unitId; // Id of attribute or column unit.
};
/**
* The attribute selection structure.
*/
struct SelValueExt {
    AIDNameUnitId attr; // The attribute specification with unit Id.
    SelOpcode oper; // The compare operator between the attribute and
    // value.
    TS_Value value; // Value for the condition.
};
/**
* Defines the sequence of selection attributes with their logical
* operators. The Idea is to have the logical operators and the selection
* values in one sequence. Therefore no implicit rules are necessary how
* logical operators have to be interpreted to the corresponding selection
* value.
*/
union SelItem switch (SelType) {
    case SEL_VALUE_TYPE: SelValueExt value;
    case SEL_OPERATOR_TYPE: SelOperator operator;
};
// Sequence definitions (S_xxx) .
typedef sequence<NameValueSeqUnit> NameValueSeqUnitSequence;
typedef sequence<AIDNameValueSeqUnitId> AIDNameValueSeqUnitIdSequence;
typedef sequence<SelValue> SelValueSequence;
typedef sequence<NameValueSeqUnitId> NameValueSeqUnitIdSequence;
typedef sequence<SelItem> SelItemSequence;
/**
* The ASAM ODS name-value pair structure.

```

```

*/
struct NameValue {
Name valName; // Attribute name or measured quantity name.
TS_Value value; // Attribute value or column value (vector) .
};
/**
* The ASAM ODS name-value-unit tuple structure. This structure is
* identical with the NameValueUnitId, except the unit is given as a
* string instead of an Id.
*/
struct NameValueUnit {
Name valName; // Attribute name or measured quantity name.
TS_Value value; // Attribute value or column value (vector) .
T_STRING unit; // Attribute or column unit as string.
};
/**
* The results for one attribute. The result set for all attributes are
* given in the sequence of the result set.
*/
struct AttrResultSet {
NameValueSeqUnitId attrValues; // The first 'how_many' results. All
// values have the same AIDName and the
// same UnitId.
NameValueUnitIdIterator rest; // The rest of the results. The
// iterator provides access to the
// consecutive
// NameValueSeqUnitId-packages of the
// result set.
};
/**
* 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.
*/
struct QueryStructure {
    AIDNameUnitIdSequence anuSeq; // 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.
    SelValueSequence condSeq; // 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.
    SelOperatorSequence operSeq; // The query condition operator sequence.
    // At the RPC API was the operator always
    // 'AND'.
    ElemId relInst; // 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.
    Name relName; // 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.
    SelOrderSequence orderBy; // The order by sequence. The order of the
    // result set. At the RPC API it was not
    // possible to set the order.
};
/**
* Result set of one application element.
*/
struct ElemResultSetExt {
    T_LONGLONG aid; // The application element Id.

```

```

NameValueSeqUnitIdSequence values; // The attribute values of the instances
// of the given application element.
};
/**
 * The extended query structure.
 */
struct QueryStructureExt {
    SelAIDNameUnitIdSequence anuSeq; // The sequence of attributes to be
// reported. 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.
    SelItemSequence condSeq; // 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.
    JoinDefSequence joinSeq; // Defined the join between the
// application elements.
    SelOrderSequence orderBy; // The order by sequence. The order of
// the result set. At the RPC API
// interface it was not possible to set
// the order.
    AIDNameSequence groupBy; // Defines the grouping attributes for a
// request, necessary if aggregate
// functions are defined in the
// SelAIDNameUnitIdSequence.
};
// Sequence definitions (S_xxx) .
typedef sequence<NameValue> NameValueSequence;
typedef sequence<NameValueUnit> NameValueUnitSequence;
typedef sequence<AttrResultSet> AttrResultSetSequence;
typedef sequence<ElemResultSetExt> ElemResultSetExtSequence;
/**

```

```

* The result set for one element. The result set for all elements are
* given in the sequence of the result set.
*/
struct ElemResultSet {
T_LONGLONG aid; // The Id of the application element.
AttrResultSetSequence attrValues; // The selected attributes of the
// element. The number of values in each
// AttrResultSet are identical, the
// attributes of one element has always
// the position in the AttrResultSet.
};
/**
* The Result set of the extended query. The iterator is for instance
* oriented access.
*/
struct ResultSetExt {
ElemResultSetExtSequence firstElems; // The sequence of the first
// how_many result elements.
ElemResultSetExtSeqIterator restElems; // the iterator, which allows to
// iterate above the result values,
// the attributes of one instance
// each iteration.
};
// Sequence definitions (S_XXX) .
typedef sequence<ElemResultSet> ElemResultSetSequence;
typedef sequence<ResultSetExt> ResultSetExtSequence;
/**
* The ASAM ODS query result types.
*/
interface ResultType {
const T_SHORT INSTELEM_ITERATOR_AS_RESULT = 0; // Iterator of instance
elements as
result of the query (the default) .
const T_SHORT TABLE_ITERATOR_AS_RESULT = 1; // Iterator for table access
as
result type of the query.
const T_SHORT TABLE_AS_RESULT = 2; // Table as result type of the

```

```

query.
};
/**
 * The lock mode of the server. The lock mode tells the way the server
 * will lock the objects as soon a modification of the server will be
 * done.
 */
interface LockMode {
const T_SHORT LOCK_INSTANCEELEMENT = 0; // Lock the instance element.
(Default
LockMode)
const T_SHORT LOCK_APPLICATIONELEMENT = 1; // Lock the application element,
all
instances of the application element are locked.
const T_SHORT LOCK_CHILDREN = 2; // Lock the children of the locked
object. This mode can be combined with one of the upper two modi.
};
/**
 * The bits of the security rights.
 */
interface SecurityRights {
const T_LONG SEC_READ = 1; // Read access is allowed.
const T_LONG SEC_UPDATE = 2; // Update access to an existing object is
allowed.
const T_LONG SEC_INSERT = 4; // Creating new instances is allowed.
const T_LONG SEC_DELETE = 8; // Delete of the object is allowed.
const T_LONG SEC_GRANT = 16; // Access rights may be passed on.
};
/**
 * The security level of an application element.
 */
interface SecurityLevel {
const T_LONG NO_SECURITY = 0; // No security defined.
const T_LONG ELEMENT_SECURITY = 1; // Security scaled for the application
element.
const T_LONG INSTANCE_SECURITY = 2; // Security scaled for instance
elements.

```

```

const T_LONG ATTRIBUTE_SECURITY = 4; // Security scaled for appliation
attributes.
};
/**
 * The ASAM ODS query constants.
 */
interface QueryConstants {
const T_LONG MaxDurationDEFAULT = 0; // Default value of max duration
parameter
of the query (no limitations) .
const T_STRING MaxDuration = "MaxDuration"; // The ASAM ODS max duration
parameter of the query.
const T_STRING QueryResultType = "QueryResultType"; // The ASAM ODS query
result type parameter.
const          T_LONG          QueryResultTypeDEFAULT          =
ResultType::INSTELEM_ITERATOR_AS_RESULT; //
Default value of the ASAM ODS query result type parameter.
};
/**
 * The ASAM ODS exception structure.
 */
exception AoException {
ErrorCode errCode;
SeverityFlag sevFlag;
T_LONG minorCode;
T_STRING reason;
};
/**
 * The ASAM factory interface.
 */
interface AoFactory {
/** (2001)
 * Get the description of the ASAM ODS factory. If the description is
 * not available an empty string is returned and no exception is
 * thrown. The server loads the description from the base attribute
 * "description" of the instance of AoEnvironment.
 *

```



```

* @throws AoException
* with the following possible error codes:
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
*
* @return The description of the ASAM ODS factory.
*/
T_STRING getDescription ()
raises (AoException) ;
/** (2002)
* Get the interface version of the ASAM ODS factory. The interface
* version is for each ODS version a fixed string. The string is the
* version of the interface implemented at the server.
*
* The format of the string is:
* V<Major Version Nr>.<Minor Version Nr>.<Revision Nr>
* - Major Version Nr (numeric: [0-9]+)
* - Minor Version Nr (numeric: [0-9]+)
* - Revision Nr (numeric: [0-9]+)
* Example of the version specification is 'V5.1.0'.
*
* @throws AoException
* with the following possible error codes:
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
*
* @return The interface version of the ASAM ODS factory.
*/
T_STRING getInterfaceVersion ()
raises (AoException) ;
/** (2003)
* Get the name of the ASAM ODS factory. If the name is not available
* an empty string is returned and no exception is thrown.
*
* @throws AoException

```

```

* with the following possible error codes:
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
*
* @return The name of the ASAM ODS factory.
*/

T_STRING getName ()
raises (AoException) ;
/** (2004)
* Get the type of the ASAM ODS factory. If the type is not available
* an empty string is returned and no exception is thrown. The server
* loads the type from the base attribute "Application_model_type" of
* the instance of AoEnvironment.
*
* @throws AoException
* with the following possible error codes:
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
*
* @return The type of the ASAM ODS factory.
*/

T_STRING getType ()
raises (AoException) ;
/** (2005)
* Establish a new session to an ASAM ODS server. The server normally
* checks the activity of the session and will close the session after
* a time period of inactivity.
*
* @throws AoException
* with the following possible error codes:
* AO_CONNECT_FAILED
* AO_CONNECT_REFUSED
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED

```

```

* AO_NO_MEMORY
* AO_OPEN_MODE_NOT_SUPPORTED
* AO_SESSION_LIMIT_REACHED
*
* @param auth A string that may contain authentication information.
* The following values are currently supported:
* USER
* PASSWORD
* 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
* read-only variable in the session.
*
* Example:
* "USER=hans, PASSWORD=secret, CREATE_COSESSION_ALLOWED
* = TRUE"
*
* @return The new created ASAM ODS session.
*/
AoSession newSession (
in T_STRING auth)
raises (AoException);
}; // Interface AoFactory.
/**
* The ASAM ODS session interface.
*/
interface AoSession {
/** (3001)
* Abort (rollback) a transaction. The changes made in the transaction
* are lost.
*
* @throws AoException

```

```

* with the following possible error codes:
* AO_ACCESS_DENIED
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
* AO_TRANSACTION_NOT_ACTIVE
*/

void abortTransaction ()
raises (AoException) ;
/** (3002)
* Close session to an ASAM ODS server. Active transactions are
* committed.
*
* @throws AoException
* with the following possible error codes:
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*/

void close ()
raises (AoException) ;
/** (3003)
* Commit a transaction. The changes made in the transaction become
* permanent.
*
* @throws AoException
* with the following possible error codes:
* AO_ACCESS_DENIED
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE

```

```

* AO_TRANSACTION_NOT_ACTIVE
*/
void commitTransaction ()
raises (AoException) ;
/** (3004)
* Get the application model from the current session by returning an
* object with the interface ApplicationStructure. The complete
* information on the application model is available through that
* interface.
*
* @throws AoException
* with the following possible error codes:
* AO_ACCESS_DENIED
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @return The application structure.
*/
ApplicationStructure getApplicationStructure ()
raises (AoException) ;
/** (3005)
* Get the application model as values from the current session.
*
* @throws AoException
* with the following possible error codes:
* AO_ACCESS_DENIED
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @return The application structure as value.
*/

```

```

ApplicationStructureValue getApplicationStructureValue ()
raises (AoException) ;
/** (3006)
 * Get the ASAM ODS base model from the current session by returning an
 * object with the interface BaseStructure. The complete information on
 * the base model is available through that interface; it includes all
 * possible base elements with all possible base attributes as
 * specified by ASAM ODS.
 *
 * @throws AoException
 * with the following possible error codes:
 * AO_CONNECTION_LOST
 * AO_IMPLEMENTATION_PROBLEM
 * AO_NOT_IMPLEMENTED
 * AO_NO_MEMORY
 * AO_SESSION_NOT_ACTIVE
 *
 * @return The base structure.
 */
BaseStructure getBaseStructure ()
raises (AoException) ;
/** (3007)
 * Get context variables from the session. A pattern string can be
 * specified to select groups of variables.
 *
 * @throws AoException
 * with the following possible error codes:
 * AO_BAD_PARAMETER
 * AO_CONNECTION_LOST
 * AO_IMPLEMENTATION_PROBLEM
 * AO_NOT_FOUND
 * AO_NOT_IMPLEMENTED
 * AO_NO_MEMORY
 * AO_SESSION_NOT_ACTIVE
 *
 * @param varPattern The name or the search pattern for the context
 * variable (s) .

```

```

*
* @return A list of context variables.
*/
NameValueIterator getContext (
in Pattern varPattern)
raises (AoException);
/** (3008)
* Get a context variable by its name from the session.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_FOUND
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param varName The name of the requested context variable.
*
* @return The requested context variable.
*/
NameValue getContextByName (
in Name varName)
raises (AoException);
/** (3009)
* List the names of context variables from the session. A pattern
* string can be specified to select groups of variables.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NOT_FOUND

```

```

* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param varPattern The name or the search pattern for the context
* variable (s) .
*
* @return A list of context variable names.
*/

NameIterator listContext (
in Pattern varPattern)
raises (AoException) ;
/** (3010)
* Remove context variables from the session. A pattern string can be
* specified to remove groups of variables.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NOT_FOUND
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param varPattern The name or the search pattern for the context
* variable (s) to be removed.
*/

void removeContext (
in Pattern varPattern)
raises (AoException) ;
/** (3011)
* Set/modify a known context variable or add a new context variable to
* the session.
*
* @throws AoException
* with the following possible error codes:

```



```

* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param contextVariable The context variable.
*/
void setContext (
in NameValue contextVariable)
raises (AoException);
/** (3012)
* Set/modify a known context variable or add a new context variable to
* the session. This is a convenience method for the frequently used
* string variable type. It uses setContext internally.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param varName The name of the context variable.
*
* @param value The value of the context variable.
*/
void setContextString (
in Name varName,
in T_STRING value)
raises (AoException);
/** (3013)
* Start a transaction on the physical storage system (e.g. database
* system). Only when a transaction is started it is allowed to create

```

* or modify instances or measurement data. The changes get permanent
 * with a commit of the transaction or will be lost with an abort of
 * the transaction. If the session is closed the transaction will be
 * committed automatically. If a transaction is already active an
 * exception is thrown.

*

* @throws AoException
 * with the following possible error codes:

* AO_ACCESS_DENIED
 * AO_CONNECTION_LOST
 * AO_IMPLEMENTATION_PROBLEM
 * AO_NOT_IMPLEMENTED
 * AO_NO_MEMORY
 * AO_SESSION_NOT_ACTIVE
 * AO_TRANSACTION_ALREADY_ACTIVE

*/

void startTransaction ()

raises (AoException) ;

/** (3014)

* Make the changes permanent.

*

* @throws AoException
 * with the following possible error codes:

* AO_ACCESS_DENIED
 * AO_CONNECTION_LOST
 * AO_IMPLEMENTATION_PROBLEM
 * AO_NOT_IMPLEMENTED
 * AO_NO_MEMORY
 * AO_SESSION_NOT_ACTIVE
 * AO_TRANSACTION_NOT_ACTIVE

*/

void flush ()

raises (AoException) ;

/** (3015)

* Every new created instance will set its initial rights to <acl> .
 * This method overrides the default-methods for applying initial
 * rights. The initial rights are only valid for the current session.

```

*
* @throws AoException
* with the following possible error codes:
* AO_ACCESS_DENIED
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
* AO_TRANSACTION_NOT_ACTIVE
*
* @param irlEntries The current initial rights.
*
* @param set Set (1) or remove (0) the current initial rights. The
* previous initial rights get lost. If the parameter set
* is 0 (remove) the parameter irlEntries will be ignored.
*/
void setCurrentInitialRights (
in InitialRightSequence irlEntries,
in T_BOOLEAN set)
raises (AoException);
/** (3016)
* Get the current lock mode. The lock mode tells the server which
* objects to lock for upcoming changes. Application elements, instance
* elements or children of elements can be locked.
*
* @throws AoException
* with the following possible error codes:
* AO_ACCESS_DENIED
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*

```

```

* @return The current lock mode. The lock mode constants are defined
* in the interface LockMode. The interface definition
* language IDL does not allow to set the values of
* enumerations thus the constant definitions had to be done
* in an interface.
*/
T_SHORT getLockMode ()
raises (AoException) ;
/** (3017)
* Set the new lock mode. The lock mode tells the server which objects
* to lock for upcoming changes. Application elements, instance
* elements or children of elements can be locked.
*
* @throws AoException
* with the following possible error codes:
* AO_ACCESS_DENIED
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
* AO_TRANSACTION_NOT_ACTIVE
*
* @param lockMode The new lock mode. The lock mode constants are
* defined in the interface LockMode. The interface
* definition language IDL does not allow to set the
* values of enumerations thus the constant
* definitions had to be done in an interface.
*/
void setLockMode (
in T_SHORT lockMode)
raises (AoException) ;
/** (3018)
* Get the application element access object from the current session.
*
* @throws AoException

```

```

* with the following possible error codes:
* AO_ACCESS_DENIED
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @return The application element access object.
*/
AppElemAccess getAppElemAccess ()
raises (AoException) ;
/** (3019)
* Change the password for user defined by <username> to <newPassword>.
* A normal user must supply his current password <oldPassword>. The
* super user 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 super user.
*
* @throws AoException
* with the following possible error codes:
* AO_ACCESS_DENIED
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
* AO_TRANSACTION_NOT_ACTIVE
* AO_WRONG_PASSWORD
*
* @param username The name of the user for which the password will
* be changed. If no username is given the password
* of the current user will be changed. If the

```

```

* username differs from the current user the current
* user must be a super user.
*
* @param oldPassword The current password of the user. A normal user
* must supply his current password. The super
* user can change the password without supplying
* the current password.
*
* @param newPassword The new password of the user.
*/
void setPassword (
in T_STRING username,
in T_STRING oldPassword,
in T_STRING newPassword)
raises (AoException) ;
/** (3020)
* Get the description of the ASAM ODS session. The description of the
* session is identical with description of the ASAM ODS factory. If
* the description is not available an empty string is returned and no
* exception is thrown. The server loads the description from the base
* attribute "description" of the instance of AoEnvironment.
*
* @throws AoException
* with the following possible error codes:
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
* AO_CONNECTION_LOST
*
* @return The description of the ASAM ODS session.
*/
T_STRING getDescription ()
raises (AoException) ;
/** (3021)
* Get the name of the ASAM ODS session. The name of the session is
* identical with the name of the ASAM ODS factory. If the name is not

```

```

* available an empty string is returned and no exception is thrown.
* The server loads the name from the base attribute "name" of the
* instance of AoEnvironment.
*
* @throws AoException
* with the following possible error codes:
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
* AO_CONNECTION_LOST
*
* @return The name of the ASAM ODS session.
*/
Name getName ()
raises (AoException) ;
/** (3022)
* Get the type of the ASAM ODS session. The type of the session is
* identical with the type of the ASAM ODS factory. If the type is not
* available an empty string is returned and no exception is thrown.
* The server loads the type from the base attribute
* "Application_model_type" of the instance of AoEnvironment.
*
* @throws AoException
* with the following possible error codes:
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
* AO_CONNECTION_LOST
*
* @return The type of the ASAM ODS session.
*/
T_STRING getType ()
raises (AoException) ;
/** (3023)
* Create a QueryEvaluator object.

```

```

*
* @throws AoException
* with the following possible error codes:
* AO_ACCESS_DENIED
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @return The new created query evaluator object
*/
QueryEvaluator createQueryEvaluator ()
raises (AoException) ;
/** (3024)
* Create a new object with the Interface Blob on the server. This
* object can be used to create an attribute value of the data type
* DT_BLOB.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @return The reference of the blob object which is generated at the
* server.
*/
Blob createBlob ()
raises (AoException) ;
/** (3025)
* 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.
*
* @throws AoException
* with the following possible error codes:
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_ACCESS_DENIED
*
* @return The co session.
*/
AoSession createCoSession ()
raises (AoException) ;
/** (3026)
* 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.
*
* @throws AoException
* with the following possible error codes:
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM

```

```

* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
*
* @return The instance element of the logged in user.
*/
InstanceElement getUser ()
raises (AoException) ;
/** (3027)
* Get the list with the attributes of all elements and the enumeration
* name which has an enumeration.
*
* @throws AoException
* with the following possible error codes:
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @return The list with attribute and there enumeration.
*/
EnumerationAttributeStructureSequence getEnumerationAttributes ()
raises (AoException) ;
/** (3028)
* Get all enumerations used in the application model. The enumerations
* and all items of the enumerations are returned to the client.
*
* @throws AoException
* with the following possible error codes:
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @return The enumeration with there Items.
*/

```

```

EnumerationStructureSequence getEnumerationStructure ()
raises (AoException) ;
}; // Interface AoSession.
/**
 * The ASAM ODS application attribute interface.
 */
interface ApplicationAttribute {
/** (4001)
 * Get the base attribute of the application attribute.
 *
 * @throws AoException
 * with the following possible error codes:
 * AO_CONNECTION_LOST
 * AO_IMPLEMENTATION_PROBLEM
 * AO_NOT_IMPLEMENTED
 * AO_NO_MEMORY
 * AO_SESSION_NOT_ACTIVE
 *
 * @return The base attribute of the application attribute. A 'null'
 * is returned if the application attribute has no base
 * attribute.
 */
BaseAttribute getBaseAttribute ()
raises (AoException) ;
/** (4002)
 * Get the data type of the application attribute.
 *
 * @throws AoException
 * with the following possible error codes:
 * AO_CONNECTION_LOST
 * AO_IMPLEMENTATION_PROBLEM
 * AO_NOT_IMPLEMENTED
 * AO_NO_MEMORY
 * AO_SESSION_NOT_ACTIVE
 *
 * @return The data type of the application attribute.
 */

```

```

DataType getDataType ()
raises (AoException) ;
/** (4003)
 * Get the maximum allowed length of the value of the application
 * attribute.
 *
 * @throws AoException
 * with the following possible error codes:
 * AO_CONNECTION_LOST
 * AO_IMPLEMENTATION_PROBLEM
 * AO_NOT_IMPLEMENTED
 * AO_NO_MEMORY
 * AO_SESSION_NOT_ACTIVE
 *
 * @return The maximum allowed length of the application attribute.
 */
T_LONG getLength ()
raises (AoException) ;
/** (4004)
 * Get the name of the application attribute.
 *
 * @throws AoException
 * with the following possible error codes:
 * AO_CONNECTION_LOST
 * AO_IMPLEMENTATION_PROBLEM
 * AO_NOT_IMPLEMENTED
 * AO_NO_MEMORY
 * AO_SESSION_NOT_ACTIVE
 *
 * @return The name of the application attribute.
 */
Name getName ()
raises (AoException) ;
/** (4005)
 * Get the unit Id of the application attribute. The unit Id is only
 * valid for the current server.
 *
```

```

* @throws AoException
* with the following possible error codes:
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @return The unit Id of the application attribute.
*/
T_LONGLONG getUnit ()
raises (AoException) ;
/** (4006)
* Get the obligatory flag of the application attribute.
*
* @throws AoException
* with the following possible error codes:
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @return The obligatory flag of the application attribute.
*/
T_BOOLEAN isObligatory ()
raises (AoException) ;
/** (4007)
* Get the unique flag of the application attribute.
*
* @throws AoException
* with the following possible error codes:
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE

```

```

*
* @return The unique flag of the application attribute.
*/
T_BOOLEAN isUnique ()
raises (AoException) ;
/** (4008)
* Set the base attribute of the application attribute. This allows the
* client to declare the application attribute (new or existing)
* additional to a base attribute. The application attribute will
* become the derived attribute of the given base attribute.
*
* It is allowed to modify the application attribute 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 data type of the base attribute is not the
* same as the data type 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.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_DUPLICATE_BASE_ATTRIBUTE
* AO_IMPLEMENTATION_PROBLEM
* AO_INVALID_DATATYPE
* AO_MISSING_VALUE
* AO_NOT_IMPLEMENTED
* AO_NOT_UNIQUE
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param baseAttr The base attribute.
*/
void setBaseAttribute (
in BaseAttribute baseAttr)
raises (AoException) ;
/** (4009)
* Set the data type of the application attribute.
*
* It is allowed to modify the application attribute outside a
* transaction but it is recommended to activate a transaction.
*
* It is not allowed to set the data type of application attributes
* that represent base attributes. An attempt to set the data type 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 data type of the base attribute is not the
* same as the data type of the instantiated attributes.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_HAS_BASE_ATTRIBUTE
* AO_IMPLEMENTATION_PROBLEM
* AO_INVALID_DATATYPE
* AO_IS_BASE_ATTRIBUTE
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param aaDataType The data type.
*/
void setDataType (
in DataType aaDataType)
raises (AoException) ;
/** (4010)
* Set the obligatory flag of the application attribute.
*
* It is allowed to modify the application attribute outside a
* transaction but it is recommended to activate a transaction.
*
* It is not allowed to set the obligatory flag of application
* attributes
* that represent base attributes. An attempt to set the obligatory

```



```

* flag of such an application attribute will result in the exception
* AO_IS_BASE_ATTRIBUTE.
*
* For performance and flexibility reasons this set-method should be
* used before the new application attribute is committed the first
* time.
*
* If this method is called before the first commit it will not throw
* the following exception:
* AO_MISSING_VALUE
*
* After the first commit, there may be instances of the application
* attribute. These instances may cause the following problem:
*
* AO_MISSING_VALUE: The obligatory flag of the base attribute is set
* but there are one or more empty values in the instances.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_HAS_BASE_ATTRIBUTE
* AO_IMPLEMENTATION_PROBLEM
* AO_IS_BASE_ATTRIBUTE
* AO_MISSING_VALUE
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param aaIsObligatory The obligatory flag.
*/
void setIsObligatory (
in T_BOOLEAN aaIsObligatory)
raises (AoException) ;
/** (4011)
* Set the unique flag of the application attribute.
*

```

- * It is allowed to modify the application attribute outside a transaction but it is recommended to activate a transaction.
- *
- * The server will check if the values of the instance attributes are unique. If this flag is set and the values of an attribute are not unique when using the method setValue an exception is thrown. If instances of the application element already exist that contain non-unique values and the flag shall be set this method throws an exception.
- *
- * It is not allowed to set the unique flag of application attributes that represent base attributes. An attempt to set the unique flag of such an application attribute will result in the exception AO_IS_BASE_ATTRIBUTE.
- *
- * If the unique flag is set to TRUE the obligatory flag is also set to TRUE. The previous values of both flag do not matter in this case. Setting the unique flag to FALSE does not affect the obligatory flag.
- *
- * For performance and flexibility reasons this set-method should be used before the new application attribute is committed the first time.
- *
- * If this method is called before the first commit it will not throw the following exception:
 - * AO_MISSING_VALUE
 - * AO_NOT_UNIQUE
- *
- * After the first commit, there may be instances of the application attribute. These instances may cause the following problem:
 - * AO_MISSING_VALUE: The obligatory flag of the base attribute is set but there are one or more empty values in the instances.
 - *
 - * AO_NOT_UNIQUE: The unique flag of the base attribute is set but the values of the instances are not unique.

```

*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_HAS_BASE_ATTRIBUTE
* AO_IMPLEMENTATION_PROBLEM
* AO_IS_BASE_ATTRIBUTE
* AO_MISSING_VALUE
* AO_NOT_IMPLEMENTED
* AO_NOT_UNIQUE
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param aaIsUnique The unique flag.
*/
void setIsUnique (
in T_BOOLEAN aaIsUnique)
raises (AoException) ;
/** (4012)
* Set the maximum allowed length of the application attribute. The
* maximum value accepted by the ASAM ODS server for this length
* information depends on the underlying physical storage. For the
* physical storage specified in chapter 3 the maximum allowed length
* will be stored in the AFLEN value in SVCATTR, and it is restricted
* by the maximum length of string type database fields. If a length is
* specified that exceeds the capabilities of the underlying physical
* storage the server will throw the exception AO_INVALID_LENGTH.
*
* It is allowed to modify the application attribute 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 only

```

* relevant for is ignored for all attribute values with the data type
 * other datatypes than DT_STRING, DS_STRING, DT_DATE, DS_DATE, all
 * components of DT_EXTERNALREFERENCE and of DS_EXTERNALREFERENCE, and
 * the header portion of DT_BLOB.

*

* For performance and flexibility reasons this set-method should be
 * used before the new application attribute is committed the first
 * time.

*

* If this method is called before the first commit it will not throw
 * the following exception:

* AO_HAS_INSTANCES

*

* After the first commit, there may be instances of the application
 * attribute. These instances may cause the exception AO_HAS_INSTANCES
 * if the instances of the application attribute are not empty.

*

* @throws AoException

* with the following possible error codes:

* AO_BAD_PARAMETER

* AO_CONNECTION_LOST

* AO_HAS_INSTANCES

* AO_IMPLEMENTATION_PROBLEM

* AO_INVALID_LENGTH

* AO_NOT_IMPLEMENTED

* AO_NO_MEMORY

* AO_SESSION_NOT_ACTIVE

*

* @param aaLength The maximum attribute length.

*/

```
void setLength (
in T_LONG aaLength)
raises (AoException) ;
```

```
/** (4013)
```

* Set the name of an application attribute.

*

* It is allowed to modify the application attribute 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.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_DUPLICATE_NAME
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param aaName The application attribute name.
*/
void setName (
in Name aaName)
raises (AoException) ;
/** (4014)
* Set the unit Id of an application attribute.
*
* It is allowed to modify the application attribute 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 data
* types:
* DT_BYTE
* DT_COMPLEX
* DT_DCOMPLEX
* DT_DOUBLE
* DT_FLOAT
* DT_LONG
* DT_LOGLONG
* DT_SHORT
* as well as for the corresponding sequence data types. For complex
* data types 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 recalculated to
* the unit of the application attribute. If there is no unit at the
* application attribute the unit at the attribute value is stored and
* reported on request at the instance.
*
* For performance and flexibility reasons this set-method should be
* used before the new application attribute is committed the first
* time.
*
* If this method is called before the first commit it will not throw
* the following exceptions:
* AO_INCOMPATIBLE_UNITS
* AO_MATH_ERROR
*
* After the first commit, there may be instances of the application
* attribute. These instances may cause the following problems:
*
* AO_INCOMPATIBLE_UNITS: No conversion rules is known to convert the

```

* unit.
*
* AO_MATH_ERROR: Converting the values to the new unit results in data
* overflow or underflow or a division by zero is detected.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_INCOMPATIBLE_UNITS
* AO_MATH_ERROR
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
* AO_UNKNOWN_UNIT
*
* @param aaUnit The unit Id.
*/
void setUnit (
in T_LONGLONG aaUnit)
raises (AoException);
/** (4015)
* The given user group the rights should be set for. <rights> defines
* the rights to set or to clear. If the parameter <set> is set to
* 'set', the rights in <rights> are set all others are cleared. If the
* parameter <set> is set to 'add', the rights in <rights> are added to
* the existing rights. If the parameter <set> is set to 'remove', the
* rights in <rights> are removed from the existing rights.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY

```

```

* AO_SESSION_NOT_ACTIVE
* AO_TRANSACTION_NOT_ACTIVE
*
* @param usergroup The user group for which the rights will be
* modified.
*
* @param rights The new right for the user group. The rights
* constants are defined in the interface
* SecurityRights. The interface definition language
* IDL does not allow to set the values of enumerations
* thus the constant definitions had to be done in an
* interface.
*
* @param set What to do with the new right.
*/
void setRights (
in InstanceElement usergroup,
in T_LONG rights,
in RightsSet set)
raises (AoException) ;
/** (4016)
* Retrieve access control list information of the given object.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @return The access control list entries of the given application
* element.
*/
ACLSequence getRights ()
raises (AoException) ;

```



```

/** (4017)
 * Return the application element to which the attribute belongs.
 *
 * @throws AoException
 * with the following possible error codes:
 * AO_BAD_PARAMETER
 * AO_CONNECTION_LOST
 * AO_IMPLEMENTATION_PROBLEM
 * AO_NOT_IMPLEMENTED
 * AO_NO_MEMORY
 * AO_SESSION_NOT_ACTIVE
 *
 * @return The application element of the attribute.
 */
ApplicationElement getApplicationElement ()
raises (AoException) ;
/** (4018)
 * Get the auto generate flag of the application attribute.
 *
 * @throws AoException
 * with the following possible error codes:
 * AO_CONNECTION_LOST
 * AO_IMPLEMENTATION_PROBLEM
 * AO_NOT_IMPLEMENTED
 * AO_NO_MEMORY
 * AO_SESSION_NOT_ACTIVE
 *
 * @return The auto generate flag of the application attribute.
 */
T_BOOLEAN isAutogenerated ()
raises (AoException) ;
/** (4019)
 * Set the auto generate flag of the application attribute.
 *
 * It is allowed to modify the application attribute outside a
 * transaction but it is recommended to activate a transaction.
 *

```

* For performance and flexibility reasons this set-method should be
 * used before the new application attribute is committed the first
 * time.

*

* @throws AoException
 * with the following possible error codes:

* AO_BAD_PARAMETER
 * AO_CONNECTION_LOST
 * AO_HAS_BASE_ATTRIBUTE
 * AO_IMPLEMENTATION_PROBLEM
 * AO_IS_BASE_ATTRIBUTE
 * AO_MISSING_VALUE
 * AO_NOT_IMPLEMENTED
 * AO_NO_MEMORY
 * AO_SESSION_NOT_ACTIVE

*

* @param isAutogenerated The auto generate flag.

*/

```
void setIsAutogenerated (
in T_BOOLEAN isAutogenerated)
raises (AoException) ;
/** (4020)
* Get the definition of the enumeration.
```

*

* @throws AoException
 * with the following possible error codes:

* AO_CONNECTION_LOST
 * AO_IMPLEMENTATION_PROBLEM
 * AO_NOT_IMPLEMENTED
 * AO_NO_MEMORY
 * AO_SESSION_NOT_ACTIVE
 * AO_INVALID_DATATYPE

*

* @return The ASAM ODS enumeration.

*/

```
EnumerationDefinition getEnumerationDefinition ()
raises (AoException) ;
```

```

/** (4021)
 * Set the definition of the enumeration. This method modifies the
 * application model, only the super user can use this method.
 *
 * The AoException AO_INVALID_DATATYPE is thrown when the data type of
 * the attribute is not DT_ENUM.
 *
 * @throws AoException
 * with the following possible error codes:
 * AO_BAD_PARAMETER
 * AO_CONNECTION_LOST
 * AO_IMPLEMENTATION_PROBLEM
 * AO_NOT_IMPLEMENTED
 * AO_NO_MEMORY
 * AO_SESSION_NOT_ACTIVE
 * AO_TRANSACTION_NOT_ACTIVE
 * AO_ACCESS_DENIED
 * AO_INVALID_DATATYPE
 *
 * @param enumDef The new enumeration definition.
 */
void setEnumerationDefinition (
in EnumerationDefinition enumDef)
raises (AoException);
/** (4022)
 * Has the attribute an unit. If this flag is set, all the attributes
 * of the instances derived from this attribute will has an unit.
 *
 * @throws AoException
 * with the following possible error codes:
 * AO_BAD_PARAMETER
 * AO_CONNECTION_LOST
 * AO_IMPLEMENTATION_PROBLEM
 * AO_NOT_IMPLEMENTED
 * AO_NO_MEMORY
 * AO_SESSION_NOT_ACTIVE
 *

```

```

* @return The flag if the attribute has an unit.
*/
T_BOOLEAN hasUnit ()
raises (AoException) ;
/** (4023)
* Set whether the attribute becomes an unit or not. A call to the
* method setUnit () will automatically set the withUnit (TRUE) .
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
* AO_TRANSACTION_NOT_ACTIVE
*
* @param withUnit The flag if the attribute becomes an unit.
*/
void withUnit (
in T_BOOLEAN withUnit)
raises (AoException) ;
/** (4024)
* Deprecated, not used any more since ASAM ODS 5.2.0.
* 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.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY

```

```

* AO_SESSION_NOT_ACTIVE
*
* @return The flag if the attribute has a value flag.
*/
T_BOOLEAN hasValueFlag ()
raises (AoException) ;
/** (4025)
* Deprecated, not used any more since ASAM ODS 5.2.0.
* 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.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
* AO_TRANSACTION_NOT_ACTIVE
*
* @param withValueFlag The flag if the attribute becomes a value
* flag.
*/
void withValueFlag (
in T_BOOLEAN withValueFlag)
raises (AoException) ;
}; // Interface ApplicationAttribute.
/**
* The ASAM ODS application element interface.
*/
interface ApplicationElement {
/** (5001)
* Create a new application attribute on the server.
*
* It is allowed to modify the application element 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.
*
* The exception AO_DUPLICATE_NAME name occurs if there is already
* another application attribute with the name "AUTOGEN".
*
* @throws AoException
* with the following possible error codes:
* AO_CONNECTION_LOST
* AO_DUPLICATE_NAME
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
* AO_TRANSACTION_NOT_ACTIVE
*
* @return The new application attribute.
*/
ApplicationAttribute createAttribute ()

```

```

raises (AoException) ;
/** (5002)
 * Create an instance of the application element.
 *
 * It is allowed to create an instance outside a transaction but it is
 * recommended to activate a transaction.
 *
 * The instance gets permanent when the transaction is committed. All
 * attributes connected to the application element are automatically
 * created and connected to the instance. The values of the attributes
 * can be set by the method setValue of the interface InstanceElement.
 *
 * @throws AoException
 * with the following possible error codes:
 * AO_BAD_PARAMETER
 * AO_CONNECTION_LOST
 * AO_IMPLEMENTATION_PROBLEM
 * AO_NOT_IMPLEMENTED
 * AO_NO_MEMORY
 * AO_SESSION_NOT_ACTIVE
 * AO_TRANSACTION_NOT_ACTIVE
 *
 * @param ieName The instance name.
 *
 * @return The new instance.
 */
InstanceElement createInstance (
in Name ieName)
raises (AoException) ;
/** (5003)
 * Get a list of all related application elements connected to this
 * application element.
 *
 * @throws AoException
 * with the following possible error codes:
 * AO_CONNECTION_LOST
 * AO_IMPLEMENTATION_PROBLEM

```

```

* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @return The related application elements.
*/
ApplicationElementSequence getAllRelatedElements ()
raises (AoException) ;
/** (5004)
* Get a list of all application relations connected to this
* application element. The inverse relation of relations connected to
* other application elements pointing to the given application
* elements are not returned.
*
* @throws AoException
* with the following possible error codes:
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @return The application relations of the application element.
*/
ApplicationRelationSequence getAllRelations ()
raises (AoException) ;
/** (5005)
* Get the application attribute of an application element which is
* inherited from the base attribute with the given name. The base name
* is case insensitive and may not contain wildcard characters.
*
* Note: The base model is case blind, e.g. Id, ID and id is all the
* same base attribute.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER

```



```

* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param baName The base attribute name.
*
* @return The application attribute.
*/
ApplicationAttribute getAttributeByBaseName (
in Name baName)
raises (AoException) ;
/** (5006)
* Get the application attribute of an application element which has
* the given name. The name is case sensitive and may not contain
* wildcard characters.
*
* Note: The application model is case sensitive, e.g. Id and ID are
* different application attributes, don't use this misleading
* attribute name.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param aaName The application attribute name.
*
* @return The application attribute.
*/
ApplicationAttribute getAttributeByName (
in Name aaName)

```

```

    raises (AoException) ;
    /** (5007)
    * Get a list of the application attributes of an application element.
    * The reference attributes are not returned.
    *
    * @throws AoException
    * with the following possible error codes:
    * AO_BAD_PARAMETER
    * AO_CONNECTION_LOST
    * AO_IMPLEMENTATION_PROBLEM
    * AO_NOT_IMPLEMENTED
    * AO_NO_MEMORY
    * AO_SESSION_NOT_ACTIVE
    *
    * @param aaPattern The name or the search pattern for the
    * application attribute name.
    *
    * @return The application attributes.
    */
    ApplicationAttributeSequence getAttributes (
    in Pattern aaPattern)
    raises (AoException) ;
    /** (5008)
    * Get the base element of an application element.
    *
    * @throws AoException
    * with the following possible error codes:
    * AO_CONNECTION_LOST
    * AO_IMPLEMENTATION_PROBLEM
    * AO_NOT_IMPLEMENTED
    * AO_NO_MEMORY
    * AO_SESSION_NOT_ACTIVE
    *
    * @return The base element.
    */
    BaseElement getBaseElement ()
    raises (AoException) ;

```

```

/** (5009)
 * Get the Id of an application element.
 *
 * @throws AoException
 * with the following possible error codes:
 * AO_CONNECTION_LOST
 * AO_IMPLEMENTATION_PROBLEM
 * AO_NOT_IMPLEMENTED
 * AO_NO_MEMORY
 * AO_SESSION_NOT_ACTIVE
 *
 * @return The Id of the application element.
 */
T_LONGLONG getId ()
raises (AoException) ;
/** (5010)
 * Get the instance element specified by the given Id. If the Id of the
 * instance is not unique an exception is thrown.
 *
 * @throws AoException
 * with the following possible error codes:
 * AO_BAD_PARAMETER
 * AO_CONNECTION_LOST
 * AO_IMPLEMENTATION_PROBLEM
 * AO_NOT_IMPLEMENTED
 * AO_NO_MEMORY
 * AO_SESSION_NOT_ACTIVE
 *
 * @param ieId The instance element Id.
 *
 * @return The instance element.
 */
InstanceElement getInstanceById (
in T_LONGLONG ieId)
raises (AoException) ;
/** (5011)
 * Get the instance element specified by the given name. If the name of

```

```

* the instance is not unique an exception is thrown.
*
* This is a 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.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_DUPLICATE_NAME
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param ieName The instance element name.
*
* @return The instance element.
*/
InstanceElement getInstanceByName (
in Name ieName)
raises (AoException) ;
/** (5012)
* Get the instances whose names match the pattern. The pattern is case
* sensitive and may contain wildcard characters.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*

```

```

* @param iePattern The name or the search pattern for the instance
* element name.
*
* @return The instance elements.
*/
InstanceElementIterator getInstances (
in Pattern iePattern)
raises (AoException) ;
/** (5013)
* Get the name of an application element.
*
* @throws AoException
* with the following possible error codes:
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @return The name of the application element.
*/
Name getName ()
raises (AoException) ;
/** (5014)
* Get related application elements connected via the specified
* relationship.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_INVALID_RELATIONSHIP
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*

```

```

* @param aeRelationship The requested relationship.
*
* @return The related application elements.
*/
ApplicationElementSequence getRelatedElementsByRelationship (
in Relationship aeRelationship)
raises (AoException) ;
/** (5015)
* Get application relations of the requested type connected from this
* application element. The inverse relations are not returned.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_INVALID_RELATION_TYPE
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param aeRelationType The requested relation type.
*
* @return The application relations.
*/
ApplicationRelationSequence getRelationsByType (
in RelationType aeRelationType)
raises (AoException) ;
/** (5016)
* Get the names of all related application elements.
*
* @throws AoException
* with the following possible error codes:
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY

```

```

* AO_SESSION_NOT_ACTIVE
*
* @return The names of the related application elements.
*/
NameSequence listAllRelatedElements ()
raises (AoException) ;
/** (5017)
* Get the application attribute names of the application element.
* There are no attribute names returned in the result list that
* contain a reference to another application element.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param aaPattern The name or the search pattern for the
* application attribute name.
*
* @return The names of the application attributes.
*/
NameSequence listAttributes (
in Pattern aaPattern)
raises (AoException) ;
/** (5018)
* Get the names of the instances whose names match the pattern. The
* pattern is case sensitive and may contain wildcard characters.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM

```

```

* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param aaPattern The name or the search pattern for the
* application attribute name.
*
* @return The names of the instances.
*/
NameIterator listInstances (
in Pattern aaPattern)
raises (AoException) ;
/** (5019)
* Get the names of related application elements connected via the
* specified relationship.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_INVALID_RELATIONSHIP
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param aeRelationship The requested relationship.
*
* @return The names of the related application elements.
*/
NameSequence listRelatedElementsByRelationship (
in Relationship aeRelationship)
raises (AoException) ;
/** (5020)
* Remove an application attribute from an application element. If
* there are instances of the application element the attribute of the
* existing instances change from application to instance attributes.

```



```

*
*
* It is allowed to modify the application element outside a
* transaction but it is recommended to activate a transaction.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_FOUND
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
* AO_TRANSACTION_NOT_ACTIVE
*
* @param applAttr The application attribute to remove.
*/
void removeAttribute (
in ApplicationAttribute applAttr)
raises (AoException) ;
/** (5021)
* Remove an instance from the application element.
*
* It is allowed to remove an instance outside a transaction but it is
* recommended to activate a transaction.
*
* The instance is removed from the server when the transaction is
* committed. If the recursive flag is set all children of the instance
* are also deleted. Removing instances is allowed only if there are no
* references (relations) to this instance. If the recursive flag is set
* a reference to one of the children is not allowed and will cause an
* exception.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER

```

```

* AO_CONNECTION_LOST
* AO_HAS_REFERENCES
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_FOUND
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
* AO_TRANSACTION_NOT_ACTIVE
*
* @param ieId The instance Id.
*
* @param recursive The recursive flag.
*/
void removeInstance (
in T_LONGLONG ieId,
in T_BOOLEAN recursive)
raises (AoException) ;
/** (5022)
* Set the base element of the application element.
*
* It is allowed to modify the application element outside a
* transaction but it is recommended to activate a transaction.
*
* The assignment to the current base element is overwritten. If there
* are instances of the application element or references to the
* application element an exception is thrown.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_HAS_INSTANCES
* AO_HAS_REFERENCES
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE

```

```

* AO_TRANSACTION_NOT_ACTIVE
*
* @param baseElem The base element.
*/
void setBaseElement (
in BaseElement baseElem)
raises (AoException) ;
/** (5023)
* Set the name of the application element.
*
* It is allowed to modify the application element 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.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_DUPLICATE_NAME
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
* AO_TRANSACTION_NOT_ACTIVE
*
* @param aeName The application element name.
*/
void setName (
in Name aeName)
raises (AoException) ;
/** (5024)
* The given user group the rights should be set for. <rights> defines

```

* the rights to set or to clear. If the parameter <set> is set to
 * 'set', the rights in <rights> are set all others are cleared. If the
 * parameter <set> is set to 'add', the rights in <rights> are added to
 * the existing rights. If the parameter <set> is set to 'remove', the
 * rights in <rights> are removed from the existing rights.

*

* @throws AoException

* with the following possible error codes:

* AO_BAD_PARAMETER

* AO_CONNECTION_LOST

* AO_IMPLEMENTATION_PROBLEM

* AO_NOT_IMPLEMENTED

* AO_NO_MEMORY

* AO_SESSION_NOT_ACTIVE

* AO_TRANSACTION_NOT_ACTIVE

*

* @param usergroup The user group for which the rights will be
 * modified.

*

* @param rights The new right for the user group. The rights
 * constants are defined in the interface
 * SecurityRights. The interface definition language
 * IDL does not allow to set the values of enumerations
 * thus the constant definitions had to be done in an
 * interface.

*

* @param set What to do with the new right.

*/

```
void setRights (
in InstanceElement usergroup,
in T_LONG rights,
in RightsSet set)
raises (AoException) ;
```

/** (5025)

* Retrieve access control list information of the given object.

*

* @throws AoException

```

* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @return The access control list entries of the given application
* element.
*/
ACLSequence getRights ()
raises (AoException) ;
/** (5026)
* Retrieve access control list information for the initial rights of
* the given object.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @return The access control list entries with the initial rights of
* the given application element.
*/
InitialRightSequence getInitialRights ()
raises (AoException) ;
/** (5027)
* The given user group the initial rights should be set for. <rights>
* defines the rights to set or to clear. If the parameter <set> is set
* to 'set', the rights in <rights> are set all others are cleared. If
* the parameter <set> is set to 'add', the rights in <rights> are
* added to the existing rights. If the parameter <set> is set to

```

```

* 'remove', the rights in <rights> are removed from the existing
* rights.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
* AO_TRANSACTION_NOT_ACTIVE
*
* @param usergroup The user group for which the initial rights will
* be modified.
*
* @param rights The new initial rights for the user group. The
* rights constants are defined in the interface
* SecurityRights. The interface definition language
* IDL does not allow to set the values of enumerations
* thus the constant definitions had to be done in an
* interface.
*
* @param refAid The Id of referencing application element for which
* the initial rights will be used. If no refAid is set
* the initial rights will be used for each new
* instance element independent of the application
* element.
*
* @param set What to do with the new initial rights.
*/
void setInitialRights (
in InstanceElement usergroup,
in T_LONG rights,
in T_LONGLONG refAid,
in RightsSet set)
raises (AoException) ;

```

```

/** (5028)
 * Set for the given application element, which relation will be used
 * to determine the initial rights for the new created instances.
 *
 * @throws AoException
 * with the following possible error codes:
 * AO_BAD_PARAMETER
 * AO_CONNECTION_LOST
 * AO_IMPLEMENTATION_PROBLEM
 * AO_NOT_IMPLEMENTED
 * AO_NO_MEMORY
 * AO_SESSION_NOT_ACTIVE
 * AO_TRANSACTION_NOT_ACTIVE
 *
 * @param applRel The application relation which will be used to
 * determine the initial rights. The relation range of
 * the application relation must be [1:1] otherwise
 * the server can not find an unique instance element
 * to retrieve the initial rights.
 *
 * @param set Set or remove the relation for the initial rights. If
 * this parameter is true the relation will be set
 * otherwise removed.
 */
void setInitialRightRelation (
in ApplicationRelation applRel,
in T_BOOLEAN set)
raises (AoException) ;
/** (5029)
 * Get all relations which are used to retrieve the instances to create
 * the initial rights of the new created instance element. If there are
 * more then one application relation the initial rights of each
 * related instance are 'ored' to the list of the initial rights.
 *
 * @throws AoException
 * with the following possible error codes:
 * AO_BAD_PARAMETER

```

```

* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @return The sequence with the application relations which will be
* used to create the initial rights of the new created
* instance element.
*/
ApplicationRelationSequence getInitialRightRelations ()
raises (AoException) ;
/** (5030)
* Get the security level of the application element. The security
* level tells if there is a security check for both application
* element and instance elements or only for the application
* attributes, the instance elements or none at all.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @return The current security level. The security level constants
* are defined in the interface SecurityLevel. The interface
* definition language IDL does not allow to set the values of
* enumerations thus the constant definitions had to be done
* in an interface.
*/
T_LONG getSecurityLevel ()
raises (AoException) ;
/** (5031)
* Set the security level for the given application element. If the

```



```

* security level is added the client is responsible for the access
* control list entries of the existing objects.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
* AO_TRANSACTION_NOT_ACTIVE
*
* @param secLevel The new security level. The security level
* constants are defined in the interface
* SecurityLevel. The interface definition language
* IDL does not allow to set the values of
* enumerations thus the constant definitions had to
* be done in an interface.
*
* @param set What to do with the new security level.
*/
void setSecurityLevel (
in T_LONG secLevel,
in RightsSet set)
raises (AoException) ;
/** (5032)
* Get the application model to which the application element belongs
* by returning an object with the interface ApplicationStructure. The
* application model provided is the same as that provided from the
* method getApplicationStructure of the Interface AoSession. This
* method guarantees that the client software is able to return to the
* session in case the session object is not available.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER

```

```

* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @return The application structure to which the application element
* belongs.
*/
ApplicationStructure getApplicationStructure ()
raises (AoException) ;
/** (5033)
* Create a list with instances. The attribute are given with the name
* of the sequence. The values of the attributes are given in the value
* sequence. The index in the different value sequences match for one
* instance element. The index in the instance element sequence of the
* related instances match for the instance with the same index in the
* value sequence.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
* AO_TRANSACTION_NOT_ACTIVE
* AO_INVALID_REQUEST
*
* @param attributes The attributes of the new created instances.
*
* @param relatedInstances The list with related instances for
* different application relations.
*
* @return The list with the new created instances.
*/

```

```

InstanceElementSequence createInstances (
in NameValueSeqUnitSequence attributes,
in ApplicationRelationInstanceElementSeqSequence relatedInstances)
raises (AoException);
/** (5034)
* 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.
*
* @throws AoException
* with the following possible error codes:
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param baseRelName The name of the base relation.
*
* @return The application relation sequence, if no relation is found
* an empty sequence is returned.
*/
ApplicationRelationSequence getRelationsByBaseName (
in Name baseRelName)
raises (AoException);
}; // Interface ApplicationElement.
/**
* The ASAM ODS application relation interface.
*
* A relation is the connection between two ASAM ODS elements; it may be
* navigated in both
* directions. The get- and set- methods are defined from the first
* element, the getInverse- and
* setInverse- methods work from the second element.

```

```

*/
interface ApplicationRelation {
/** (6001)
* Get the base relation of the application relation.
*
* @throws AoException
* with the following possible error codes:
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @return The base relation of the application relation. A 'null' is
* returned if the application relation has no base relation.
*/
BaseRelation getBaseRelation ()
raises (AoException) ;
/** (6002)
* Get the first application element of the application relation.
*
* @throws AoException
* with the following possible error codes:
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @return The first application element of the application relation.
*/
ApplicationElement getElem1 ()
raises (AoException) ;
/** (6003)
* Get the second application element of the application relation.
*
* @throws AoException

```

```

* with the following possible error codes:
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @return The second application element of the application relation.
*/
ApplicationElement getElem2 ()
raises (AoException) ;
/** (6004)
* Get the inverse relation range of the application relation.
*
* @throws AoException
* with the following possible error codes:
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @return The inverse relation range of the application relation.
*/
RelationRange getInverseRelationRange ()
raises (AoException) ;
/** (6005)
* Get the inverse relationship of the application relation.
*
* @throws AoException
* with the following possible error codes:
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*

```

```

    * @return The inverse relationship of the application relation.
    */
    Relationship getInverseRelationship ()
    raises (AoException) ;
    /** (6006)
    * Get the name of the application relation.
    *
    * @throws AoException
    * with the following possible error codes:
    * AO_CONNECTION_LOST
    * AO_IMPLEMENTATION_PROBLEM
    * AO_NOT_IMPLEMENTED
    * AO_NO_MEMORY
    * AO_SESSION_NOT_ACTIVE
    *
    * @return The name of the application relation.
    */
    Name getRelationName ()
    raises (AoException) ;
    /** (6007)
    * Get the relation range of the application relation.
    *
    * @throws AoException
    * with the following possible error codes:
    * AO_CONNECTION_LOST
    * AO_IMPLEMENTATION_PROBLEM
    * AO_NOT_IMPLEMENTED
    * AO_NO_MEMORY
    * AO_SESSION_NOT_ACTIVE
    *
    * @return The relation range of the application relation.
    */
    RelationRange getRelationRange ()
    raises (AoException) ;
    /** (6008)
    * Get the relationship of the application relation.
    *

```

```

* @throws AoException
* with the following possible error codes:
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @return The relationship of the application relation.
*/
Relationship getRelationship ()
raises (AoException) ;
/** (6009)
* Get the relation type of the application relation.
*
* @throws AoException
* with the following possible error codes:
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @return The relation type of the application relation.
*/
RelationType getRelationType ()
raises (AoException) ;
/** (6010)
* Set the base relation of the application relation.
*
* It is allowed to modify the application relation outside a
* transaction but it is recommended to activate a transaction.
*
* The relation type and relation range is copied from the base
* relation. The previous values get lost.
*
* @throws AoException

```

```

* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_INVALID_RELATION
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
* AO_TRANSACTION_NOT_ACTIVE
*
* @param baseRel The base relation.
*/
void setBaseRelation (
in BaseRelation baseRel)
raises (AoException) ;
/** (6011)
* Set the first application element of the application relation.
*
* It is allowed to modify the application relation outside a
* transaction but it is recommended to activate a transaction.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_INVALID_ELEMENT
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
* AO_TRANSACTION_NOT_ACTIVE
*
* @param applElem The application element.
*/
void setElem1 (
in ApplicationElement applElem)
raises (AoException) ;

```



```

/** (6012)
 * Set the second application element of the application relation.
 *
 * It is allowed to modify the application relation outside a
 * transaction but it is recommended to activate a transaction.
 *
 * @throws AoException
 * with the following possible error codes:
 * AO_BAD_PARAMETER
 * AO_CONNECTION_LOST
 * AO_IMPLEMENTATION_PROBLEM
 * AO_INVALID_ELEMENT
 * AO_NOT_IMPLEMENTED
 * AO_NO_MEMORY
 * AO_SESSION_NOT_ACTIVE
 * AO_TRANSACTION_NOT_ACTIVE
 *
 * @param applElem The application element.
 */
void setElem2 (
in ApplicationElement applElem)
raises (AoException) ;
/** (6013)
 * Set the relation range of an application relation.
 *
 * It is allowed to modify the application relation outside a
 * transaction but it is recommended to activate a transaction.
 *
 * It is only allowed to set the relation type if no base relation is
 * defined.
 *
 * @throws AoException
 * with the following possible error codes:
 * AO_BAD_PARAMETER
 * AO_CONNECTION_LOST
 * AO_HAS_BASE_RELATION
 * AO_IMPLEMENTATION_PROBLEM

```

```

* AO_INVALID_RELATION_RANGE
* AO_IS_BASE_RELATION
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param arRelationRange The inverse relation range.
*/
void setInverseRelationRange (
in RelationRange arRelationRange)
raises (AoException) ;
/** (6014)
* Set the name of an application relation.
*
* It is allowed to modify the application relation 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.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param arName The application relation name.
*/
void setRelationName (
in Name arName)
raises (AoException) ;
/** (6015)
* Set the relation range of an application relation.

```

```

*
* It is allowed to modify the application relation outside a
* transaction but it is recommended to activate a transaction.
*
* It is only allowed to set the relation type if no base relation is
* defined.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_HAS_BASE_RELATION
* AO_IMPLEMENTATION_PROBLEM
* AO_INVALID_RELATION_RANGE
* AO_IS_BASE_RELATION
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
* AO_TRANSACTION_NOT_ACTIVE
*
* @param arRelationRange The relation range.
*/
void setRelationRange (
in RelationRange arRelationRange)
raises (AoException) ;
/** (6016)
* Set the relation type of an application relation.
*
* It is allowed to modify the application relation 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 relation types
 * 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
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_HAS_BASE_RELATION
* AO_IMPLEMENTATION_PROBLEM
* AO_INVALID_RELATION_TYPE
* AO_IS_BASE_RELATION
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
* AO_TRANSACTION_NOT_ACTIVE
*
* @param arRelationType The relation type.
*/
void setRelationType (
in RelationType arRelationType)
raises (AoException) ;
/** (6017)
* Get the inverse name of the application relation. The inverse name
* of an application relation is the name of the relation seen from the
* other application element.
*
* @throws AoException
* with the following possible error codes:
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE

```

```

*
* @return The inverse name of the application relation.
*/
Name getInverseRelationName ()
raises (AoException) ;
/** (6018)
* Set the name of an application relation.
*
* It is allowed to modify the application relation outside a
* transaction but it is recommended to activate a transaction.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
* AO_TRANSACTION_NOT_ACTIVE
*
* @param arInvName The inverse application relation name.
*/
void setInverseRelationName (
in Name arInvName)
raises (AoException) ;
}; // Interface ApplicationRelation.
/**
* The ASAM ODS application structure interface.
*/
interface ApplicationStructure {
/** (7001)
* Check the application model for ASAM ODS conformity. The first error
* found is reported by an exception. The following checks are
* performed:
*
* - Each application element must be derived from a valid base

```

```

* element.
* - An application attribute is derived from one base attribute. It
* is not allowed to derive more the one application attribute from the
* same base attribute. It is allowed that application attributes are
* not derived from any base attribute.
* - All application elements must have at least the mandatory
* attributes.
* - Each application elements must be identified by a unique Asam
* path. No "floating" application elements are allowed.
* - All relations required by the base model must be present.
*
* @throws AoException
* with the following possible error codes:
* AO_CONNECTION_LOST
* AO_DUPLICATE_BASE_ATTRIBUTE
* AO_IMPLEMENTATION_PROBLEM
* AO_INVALID_RELATION
* AO_MISSING_ATTRIBUTE
* AO_MISSING_RELATION
* AO_MISSING_APPLICATION_ELEMENT
* AO_NOT_IMPLEMENTED
* AO_NO_PATH_TO_ELEMENT
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*/
void check ()
raises (AoException) ;
/** (7002)
* Create a new application element in the application model.
*
* It is allowed to modify the application model outside a transaction
* but it is recommended to activate a transaction.
*
* The information whether or not the new application element is a top
* level element is taken from the specified base element. The Id of
* the application element is set automatically. The mandatory base
* attributes are created automatically. Optional attributes have to be

```

```

* created by the calling program. The application attribute interface
* methods may be used to modify the attributes.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
* AO_TRANSACTION_NOT_ACTIVE
*
* @param baseElem The base element from which the application
* element is derived.
*
* @return The new application element.
*/
ApplicationElement createElement (
in BaseElement baseElem)
raises (AoException) ;
/** (7003)
* Create a new relation.
*
* It is allowed to modify the application model outside a transaction
* but it is recommended to activate a transaction.
*
* The relation is part of the application model. The application
* relation interface methods may be used to modify the relation.
*
* The default properties of a new application relation are:
* BaseRelation NULL
* Element1 NULL
* Element2 NULL
* Range -2, -2
* Name NULL
* Type INFO

```



```

* When element 1 or element 2 is set before the name of the relation
* is specified, the name of the application relation is set to
* "AUTOGEN".

```

```

*

```

```

* @throws AoException
* with the following possible error codes:
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
* AO_TRANSACTION_NOT_ACTIVE

```

```

*

```

```

* @return The new application relation.

```

```

*/

```

```

ApplicationRelation createRelation ()

```

```

raises (AoException) ;

```

```

/** (7004)

```

```

* Get the application element with the requested Id.

```

```

*

```

```

* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE

```

```

*

```

```

* @param aeId The Id of the requested application element.

```

```

*

```

```

* @return The requested application element.

```

```

*/

```

```

ApplicationElement getElementById (

```

```

in T_LONGLONG aeId)

```

```

raises (AoException) ;

```

```

/** (7005)

```

```

* Get the application element with the requested name.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param aeName The name of the requested application element.
*
* @return The requested application element.
*/
ApplicationElement getElementByName (
in Name aeName)
raises (AoException) ;
/** (7006)
* Get the application elements whose names match the pattern. The
* pattern is case sensitive and may contain wildcard characters.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param aePattern The name or the search pattern for the requested
* application elements.
*
* @return The requested application elements.
*/
ApplicationElementSequence getElements (

```

```

in Pattern aePattern)
raises (AoException);
/** (7007)
 * Get the names of application elements that are derived from the
 * specified base element.
 *
 * @throws AoException
 * with the following possible error codes:
 * AO_BAD_PARAMETER
 * AO_CONNECTION_LOST
 * AO_IMPLEMENTATION_PROBLEM
 * AO_INVALID_BASETYPE
 * AO_NOT_IMPLEMENTED
 * AO_NO_MEMORY
 * AO_SESSION_NOT_ACTIVE
 *
 * @param aeType The requested base element type. The base element
 * type can be a pattern.
 *
 * @return The requested application element names.
 */
ApplicationElementSequence getElementsByBaseType (
in BaseType aeType)
raises (AoException);
/** (7008)
 * Get the instance element specified by the ASAM path.
 *
 * @throws AoException
 * with the following possible error codes:
 * AO_BAD_PARAMETER
 * AO_CONNECTION_LOST
 * AO_IMPLEMENTATION_PROBLEM
 * AO_INVALID_ASAM_PATH
 * AO_NOT_IMPLEMENTED
 * AO_NO_MEMORY
 * AO_SESSION_NOT_ACTIVE
 *

```

```

* @param asamPath The ASAM path of the requested instance element.
*
* @return The requested instance element.
*/
InstanceElement getInstanceByAsamPath (
in Name asamPath)
raises (AoException) ;
/** (7009)
* Returns the relations between two application elements.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param applElem1 The first application element.
*
* @param applElem2 The second application element.
*
* @return The relations between the specified application elements.
*/
ApplicationRelationSequence getRelations (
in ApplicationElement applElem1,
in ApplicationElement applElem2)
raises (AoException) ;
/** (7010)
* Get the top level application elements which are 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.
*
* @throws AoException
* with the following possible error codes:

```

```

* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_INVALID_BASETYPE
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param aeType The requested base type. The base element type can
* be a pattern.
*
* @return The top level application elements.
*/
ApplicationElementSequence getTopLevelElements (
in BaseType aeType)
raises (AoException);
/** (7011)
* Get the names of the application elements that match the pattern.
* The pattern is case sensitive and may contain wildcard characters.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param aePattern The name or the search pattern for the requested
* base elements.
*
* @return The names of the application elements.
*/
NameSequence listElements (
in Pattern aePattern)
raises (AoException);

```

```

/** (7012)
 * Get the names of application elements that are derived from the
 * given base type.
 *
 * @throws AoException
 * with the following possible error codes:
 * AO_BAD_PARAMETER
 * AO_CONNECTION_LOST
 * AO_IMPLEMENTATION_PROBLEM
 * AO_INVALID_BASETYPE
 * AO_NOT_IMPLEMENTED
 * AO_NO_MEMORY
 * AO_SESSION_NOT_ACTIVE
 *
 * @param aeType The requested base type. The base element type can
 * be a pattern.
 *
 * @return The names of the application elements.
 */
NameSequence listElementsByBaseType (
in BaseType aeType)
raises (AoException) ;
/** (7013)
 * Get the names of the top level application elements that are derived
 * from the given base type. If the given base type is not a top level
 * base element an exception is thrown. A top level application element
 * is an application element without a father.
 *
 * @throws AoException
 * with the following possible error codes:
 * AO_BAD_PARAMETER
 * AO_CONNECTION_LOST
 * AO_IMPLEMENTATION_PROBLEM
 * AO_INVALID_BASETYPE
 * AO_NOT_IMPLEMENTED
 * AO_NO_MEMORY
 * AO_SESSION_NOT_ACTIVE

```

```

*
* @param aeType The requested base type.
*
* @return The names of the application elements.
*/
NameSequence listTopLevelElements (
in BaseType aeType)
raises (AoException) ;
/** (7014)
* Remove an application element from the application model.
*
* It is allowed to modify the application model outside a transaction
* but it is recommended to activate a transaction.
*
* - Only allowed:
* - if the application element is empty
*   (has no instances) .
* - no relations with other application elements.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_HAS_INSTANCES
* AO_HAS_REFERENCES
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_FOUND
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
* AO_TRANSACTION_NOT_ACTIVE
*
* @param applElem The application element to be removed.
*/
void removeElement (
in ApplicationElement applElem)
raises (AoException) ;

```

```

/** (7015)
 * This method removes an application relation from the model.
 *
 * It is allowed to modify the application model outside a transaction
 * but it is recommended to activate a transaction.
 *
 * The elements of the relation are still part of the application
 * model. If there are instances of the relation they are also removed.
 *
 * @throws AoException
 * with the following possible error codes:
 * AO_BAD_PARAMETER
 * AO_CONNECTION_LOST
 * AO_HAS_INSTANCES
 * AO_IMPLEMENTATION_PROBLEM
 * AO_NOT_FOUND
 * AO_NOT_IMPLEMENTED
 * AO_NO_MEMORY
 * AO_SESSION_NOT_ACTIVE
 * AO_TRANSACTION_NOT_ACTIVE
 *
 * @param applRel The application relation to be removed.
 */
void removeRelation (
in ApplicationRelation applRel)
raises (AoException) ;
/** (7016)
 * Get the instance elements specified by the element id.
 *
 * @throws AoException
 * with the following possible error codes:
 * AO_BAD_PARAMETER
 * AO_CONNECTION_LOST
 * AO_IMPLEMENTATION_PROBLEM
 * AO_INVALID_ASAM_PATH
 * AO_NOT_IMPLEMENTED
 * AO_NO_MEMORY

```



```

* AO_SESSION_NOT_ACTIVE
*
* @param ieIds The sequence with the element id.
*
* @return The requested instance element sequence.
*/
InstanceElementSequence getInstancesById (
in ElemIdSequence ieIds)
raises (AoException) ;
/** (7017)
* Get the current client session in which the application model is
* created.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @return The current client session.
*/
AoSession getSession ()
raises (AoException) ;
/** (7018)
* Create a new enumeration definition. This method modifies the
* application model and is only allowed for the super user.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY

```

```

* AO_SESSION_NOT_ACTIVE
* AO_TRANSACTION_NOT_ACTIVE
* AO_ACCESS_DENIED
*
* @param enumName Name of the enumeration
*
* @return The new created enumeration
*/
EnumerationDefinition createEnumerationDefinition (
in T_STRING enumName)
raises (AoException) ;
/** (7019)
* Remove the enumeration definition. The server checks if the
* enumeration is still in use by one of the attributes. This method
* modifies the application model and is only allowed for the super
* user.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
* AO_TRANSACTION_NOT_ACTIVE
* AO_ACCESS_DENIED
*
* @param enumName Name of the enumeration to remove.
*/
void removeEnumerationDefinition (
in T_STRING enumName)
raises (AoException) ;
/** (7020)
* Get the list of all enumeration names.
*
* @throws AoException

```

```

* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @return List with all enumeration names.
*/
NameSequence listEnumerations ()
raises (AoException) ;
/** (7021)
* Get the specified enumeration definition.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param enumName Name of the requested enumeration.
*
* @return The enumeration definition.
*/
EnumerationDefinition getEnumerationDefinition (
in T_STRING enumName)
raises (AoException) ;
/** (7022)
* Create the relation between a list of instances. The number of
* instances in both list must be identical. The application element of
* the instances in each list must be identical. The application
* elements must match the application elements of the application
* relation. The index in the list of the instances defines related

```

```

* instances.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
* AO_TRANSACTION_NOT_ACTIVE
* AO_INVALID_REQUEST
*
* @param applRel The application relation.
*
* @param elemList1 The list with the instances of one application
* element for which the relation will be created.
*
* @param elemList2 The list with the related instances.
*/
void createInstanceRelations (
in ApplicationRelation applRel,
in InstanceElementSequence elemList1,
in InstanceElementSequence elemList2)
raises (AoException) ;
}; // Interface ApplicationStructure.
/**
* The ASAM ODS base attribute interface.
*/
interface BaseAttribute {
/** (8001)
* Get the data type of the base attribute.
*
* @throws AoException
* with the following possible error codes:
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM

```

```

* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @return The data type of the base attribute.
*/
DataType getDataType ()
raises (AoException) ;
/** (8002)
* Get the name of the base attribute.
*
* @throws AoException
* with the following possible error codes:
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @return The name of the base attribute.
*/
Name getName ()
raises (AoException) ;
/** (8003)
* Get the obligatory flag of the base attribute.
*
* @throws AoException
* with the following possible error codes:
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @return The obligatory flag of the base attribute.
*/
T_BOOLEAN isObligatory ()

```

```

    raises (AoException) ;
    /** (8004)
    * Get the unique flag of the base attribute.
    *
    * @throws AoException
    * with the following possible error codes:
    * AO_CONNECTION_LOST
    * AO_IMPLEMENTATION_PROBLEM
    * AO_NOT_IMPLEMENTED
    * AO_NO_MEMORY
    * AO_SESSION_NOT_ACTIVE
    *
    * @return The unique flag of the base attribute.
    */
    T_BOOLEAN isUnique ()
    raises (AoException) ;
    /** (8005)
    * Return the base element to which the attribute belongs..
    *
    * @throws AoException
    * with the following possible error codes:
    * AO_BAD_PARAMETER
    * AO_CONNECTION_LOST
    * AO_IMPLEMENTATION_PROBLEM
    * AO_NOT_IMPLEMENTED
    * AO_NO_MEMORY
    * AO_SESSION_NOT_ACTIVE
    *
    * @return The base element of the attribute.
    */
    BaseElement getBaseElement ()
    raises (AoException) ;
    /** (8006)
    * Get the definition of the enumeration of the base attribute.
    *
    * @throws AoException
    * with the following possible error codes:

```

```

* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
* AO_INVALID_DATATYPE
*
* @return The ASAM ODS enumeration.
*/
EnumerationDefinition getEnumerationDefinition ()
raises (AoException) ;
}; // Interface BaseAttribute.
/**
* The ASAM ODS base element interface.
*/
interface BaseElement {
/** (9001)
* Get all known relations of the base element.
*
* @throws AoException
* with the following possible error codes:
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @return All known relations of the base element.
*/
BaseRelationSequence getAllRelations ()
raises (AoException) ;
/** (9002)
* Get attributes of the base element.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER

```

```

* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param baPattern The name or the search pattern for the requested
* base attributes.
*
* @return The requested attributes of the base element.
*/
BaseAttributeSequence getAttributes (
in Pattern baPattern)
raises (AoException) ;
/** (9003)
* Get the related elements of a base element defined by the
* relationship.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_INVALID_RELATIONSHIP
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param brRelationship The requested relationship.
*
* @return The related elements of a base element.
*/
BaseElementSequence getRelatedElementsByRelationship (
in Relationship brRelationship)
raises (AoException) ;
/** (9004)
* Get the base element's relations of the requested relation type.

```



```

*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_INVALID_RELATION_TYPE
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param brRelationType The requested relation type.
*
* @return The base element's relations of the requested type.
*/
BaseRelationSequence getRelationsByType (
in RelationType brRelationType)
raises (AoException) ;
/** (9005)
* Get the type of the base element. The type of the base element is
* identical with the name of the base element. The type of the base
* element is a string.
*
* @throws AoException
* with the following possible error codes:
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @return The type of the base element.
*/
BaseType getType ()
raises (AoException) ;
/** (9006)
* Get whether or not the base element is a top level element. Top

```

```

* level elements are elements without a father.
*
* @throws AoException
* with the following possible error codes:
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @return Boolean whether or not the base element is a top level
* element.
*/
T_BOOLEAN isTopLevel ()
raises (AoException) ;
/** (9007)
* Get attribute names of the base element.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param baPattern The name or the search pattern for the requested
* base attribute names.
*
* @return The requested attribute names of the base element.
*/
NameSequence listAttributes (
in Pattern baPattern)
raises (AoException) ;
/** (9008)
* Get the related element names of the base element defined by the

```

```

* relationship.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_INVALID_RELATIONSHIP
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param brRelationship The requested relationship.
*
* @return The related element names of the base element.
*/
BaseTypeSequence listRelatedElementsByRelationship (
in Relationship brRelationship)
raises (AoException) ;
}; // Interface BaseElement.
/**
* The ASAM ODS base relation interface.
*/
interface BaseRelation {
/** (10001)
* Get the first base element of the base relation.
*
* @throws AoException
* with the following possible error codes:
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @return The first base element of the base relation.
*/

```

```

BaseElement getElem1 ()
raises (AoException) ;
/** (10002)
 * Get the second base element of the base relation.
 *
 * @throws AoException
 * with the following possible error codes:
 * AO_CONNECTION_LOST
 * AO_IMPLEMENTATION_PROBLEM
 * AO_NOT_IMPLEMENTED
 * AO_NO_MEMORY
 * AO_SESSION_NOT_ACTIVE
 *
 * @return The second base element of the base relation.
 */
BaseElement getElem2 ()
raises (AoException) ;
/** (10003)
 * Get the inverse relation range of the base relation.
 *
 * @throws AoException
 * with the following possible error codes:
 * AO_CONNECTION_LOST
 * AO_IMPLEMENTATION_PROBLEM
 * AO_NOT_IMPLEMENTED
 * AO_NO_MEMORY
 * AO_SESSION_NOT_ACTIVE
 *
 * @return The inverse relation range of the base relation.
 */
RelationRange getInverseRelationRange ()
raises (AoException) ;
/** (10004)
 * Get the inverse relationship of the base relation.
 *
 * @throws AoException
 * with the following possible error codes:

```

```

* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @return The inverse relationship of the base relation.
*/
Relationship getInverseRelationship ()
raises (AoException) ;
/** (10005)
* Get the relation name of the base relation.
*
* @throws AoException
* with the following possible error codes:
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @return The relation name of the base relation.
*/
Name getRelationName ()
raises (AoException) ;
/** (10006)
* Get the relation range of the base relation.
*
* @throws AoException
* with the following possible error codes:
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @return The relation range of the base relation.

```

```

*/
RelationRange getRelationRange ()
raises (AoException) ;
/** (10007)
* Get the relationship of the base relation.
*
* @throws AoException
* with the following possible error codes:
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @return The relationship of the base relation.
*/
Relationship getRelationship ()
raises (AoException) ;
/** (10008)
* Get the relation type of the base relation.
*
* @throws AoException
* with the following possible error codes:
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @return The relation type of the base relation.
*/
RelationType getRelationType ()
raises (AoException) ;
/** (10009)
* Return the inverse name of the base relation.
*
* @throws AoException

```

```

* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @return The name of the inverse relation.
*/
Name getInverseRelationName ()
raises (AoException) ;
}; // Interface BaseRelation.
/**
* The ASAM ODS base structure interface.
*/
interface BaseStructure {
/** (11001)
* Get the base element that matches the requested type. The type of a
* base element is identical with the name of the base element.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_INVALID_BASETYPE
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param beType The name of the requested base element.
*
* @return The requested base element.
*/
BaseElement getElementByType (
in BaseType beType)

```

```

    raises (AoException) ;
    /** (11002)
    * Get the base elements that match the pattern. The pattern is case
    * sensitive and may contain wildcard characters.
    *
    * @throws AoException
    * with the following possible error codes:
    * AO_BAD_PARAMETER
    * AO_CONNECTION_LOST
    * AO_IMPLEMENTATION_PROBLEM
    * AO_NOT_IMPLEMENTED
    * AO_NO_MEMORY
    * AO_SESSION_NOT_ACTIVE
    *
    * @param bePattern The name or the search pattern for the requested
    * base elements.
    *
    * @return The requested base elements.
    */
    BaseElementSequence getElements (
    in Pattern bePattern)
    raises (AoException) ;
    /** (11003)
    * Get the base relation between two base elements. The base relation
    * is given from the first element (parameter elem1) to the second
    * element (parameter elem2), the inverse relation is not given.
    *
    * @throws AoException
    * with the following possible error codes:
    * AO_BAD_PARAMETER
    * AO_CONNECTION_LOST
    * AO_IMPLEMENTATION_PROBLEM
    * AO_NOT_IMPLEMENTED
    * AO_NO_MEMORY
    * AO_SESSION_NOT_ACTIVE
    *
    * @param elem1 The base element from which the relation starts.

```



```

*
* @param elem2 The base element to which the relation points.
*
* @return The base relation between the two base elements.
*/
BaseRelation getRelation (
in BaseElement elem1,
in BaseElement elem2)
raises (AoException) ;
/** (11004)
* Get the top level base elements that match the pattern. The pattern
* is case sensitive and may contain wildcard characters. A top level
* base element is a base element without a father.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param bePattern The name or the search pattern for the requested
* top level base elements.
*
* @return The requested top level base elements.
*/
BaseElementSequence getTopLevelElements (
in Pattern bePattern)
raises (AoException) ;
/** (11005)
* Get the version of the base model. The version of the base model is
* the version of the ASAM ODS base model.
*
* @throws AoException
* with the following possible error codes:

```

```

* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @return The version of the ASAM ODS base model.
*/
T_STRING getVersion ()
raises (AoException) ;
/** (11006)
* Get the base element names that match the pattern. The pattern is
* case sensitive and may contain wildcard characters.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param bePattern The name or the search pattern for the requested
* base element names.
*
* @return The requested base element names.
*/
BaseTypeSequence listElements (
in Pattern bePattern)
raises (AoException) ;
/** (11007)
* Get the top level base element names that match the pattern. The
* pattern is case sensitive and may contain wildcard characters. A top
* level base element is a base element without a father.
*
* @throws AoException

```

```

* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param bePattern The name or the search pattern for the requested
* top level base element names.
*
* @return The requested top level base element names.
*/
BaseTypeSequence listTopLevelElements (
in Pattern bePattern)
raises (AoException);
}; // Interface BaseStructure.
/**
* The ASAM ODS blob interface.
*/
interface Blob {
/** (12001)
* Append a byte sequence to the binary large object.
*
* It is allowed to modify the binary large object outside a
* transaction but it is recommended to activate a transaction.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param value The byte sequence.

```

```

*/
void append (
in S_BYTE value)
raises (AoException) ;
/** (12002)
* Compares the content of the binary large object. The headers are not
* compared.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param aBlob The blob to compare.
*
* @return A flag whether or not the compared blobs are equal.
*/
T_BOOLEAN compare (
in T_BLOB aBlob)
raises (AoException) ;
/** (12003)
* Get a part of the binary large object.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param offset The starting position of the data in the blob.

```

```

*
* @param length The number of bytes requested from the blob.
*
* @return The request part of the blob data.
*/
S_BYTE get (
in T_LONG offset,
in T_LONG length)
raises (AoException) ;
/** (12004)
* Get the header of the binary large object.
*
* @throws AoException
* with the following possible error codes:
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @return The blob header.
*/
T_STRING getHeader ()
raises (AoException) ;
/** (12005)
* Get the length of the binary large object without loading it.
*
* @throws AoException
* with the following possible error codes:
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @return The blob length.
*/

```

```

T_LONG getLength ()
raises (AoException) ;
/** (12006)
* Clear the binary large object and set the new data.
*
* It is allowed to modify the binary large object outside a
* transaction but it is recommended to activate a transaction.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param value The new blob data.
*/
void set (
in S_BYTE value)
raises (AoException) ;
/** (12007)
* Set the header of a binary large object.
*
* It is allowed to modify the binary large object outside a
* transaction but it is recommended to activate a transaction.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*

```

```

* @param header The blob header.
*/
void setHeader (
in T_STRING header)
raises (AoException) ;
/** (12008)
* Destroy the object on the server. The destructor of the client, so
* the server knows this object is not used anymore by the client.
* Access to this object after the destroy method will lead to an
* exception.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*/
void destroy ()
raises (AoException) ;
}; // Interface Blob.
/**
* The ASAM ODS column interface. It is not inherited from
* InstanceElement. Via the method getColumn of the interface SubMatrix
* the column is accessed. The column is only used for read access. With
* the creation of instances at the application element of type
* AoLocalColumn and the relation to the instances of the application
* element of type AoSubMatrix a local column can be created. The column
* name is used for the SMatLink and the column is used to store a formula
* for the calculation of the values of the column. There is no definition
* of the formula language at the moment.
*/
interface Column {
/** (13001)
* Get the formula of the column.

```

```

*
* There is no formula defined in ASAM ODS, so there is nothing to
* return.
*
* @throws AoException
* with the following possible error codes:
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @return The formula of the column.
*/
T_STRING getFormula ()
raises (AoException) ;
/** (13002)
* Get the name of the column.
*
* @throws AoException
* with the following possible error codes:
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @return The name of the column.
*/
Name getName ()
raises (AoException) ;
/** (13003)
* Get the source measurement quantity.
*
* @throws AoException
* with the following possible error codes:
* AO_CONNECTION_LOST

```



```

* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @return The source measurement quantity.
*/
InstanceElement getSourceMQ ()
raises (AoException) ;
/** (13004)
* Get the unit of the column.
*
* The method returns the unit of the values according the current
* setting. Initially within a session this is the unit as given in the
* storage. After setUnit () is called, the unit as specified by
* setUnit () will be
* returned for the particular column. Thus the unit returned will be
* retrieved by the ODS server according to following strategy:
* 1) if a unit has already been set for the particular column by
* setUnit (...) prior; that unit is returned, else
* 2) if the related measurement quantity has a unit (i.e. it has a
* relation derived from the base relation 'unit' referencing an
* instance of base type AoUnit) , that unit's name is returned, else
* 3) if the related measurement quantity has a relation to a quantity
* (i.e. its relation derived from the base relation 'quantity' is
* referencing an instance of base type AoQuantity) , and this quantity
* has
* a default unit (i.e. it has a relation derived from the base
* relation 'default_unit' referencing an instance of base type
* AoUnit) , that unit's name is returned, else
* 4) an empty string is returned.
*
* @throws AoException
* with the following possible error codes:
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED

```

```

* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @return The unit of the column.
*/
T_STRING getUnit ()
raises (AoException) ;
/** (13005)
* Set the formula of the column.
*
* It is allowed to modify the column 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.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param formula The formula.
*/
void setFormula (
in T_STRING formula)
raises (AoException) ;
/** (13006)
* Set the unit of the column. This is only a temporary unit for
* getting the values of a column. This unit is not stored in the
* storage.
* When an empty string is given the unit is returned to the original
* setting.
* When the value matrix where the column belongs to is in the STORAGE
* mode this method have no influence on the values returned from the

```

```

* value matrix.
* To change the unit permanent in the storage the relation at the
* measurement quantity must be changed by the client.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param unit The physical unit.
*/
void setUnit (
in T_STRING unit)
raises (AoException) ;
/** (13007)
* Is the column an independent column
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @return The independent flag of the column.
*/
T_BOOLEAN isIndependent ()
raises (AoException) ;
/** (13008)
* Is the column an scaling column
*

```

```

* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @return Tells if the column is a scaling column.
*/
T_BOOLEAN isScaling ()
raises (AoException) ;
/** (13009)
* Set the column as an independent column.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
* AO_TRANSACTION_NOT_ACTIVE
*
* @param independent The new value of the independent flag.
*/
void setIndependent (
in T_BOOLEAN independent)
raises (AoException) ;
/** (13010)
* Set the column to a scaling column.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER

```

```

* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
* AO_TRANSACTION_NOT_ACTIVE
*
* @param scaling The new value of the scaling flag.
*/
void setScaling (
in T_BOOLEAN scaling)
raises (AoException) ;
/** (13011)
* Get the data type of the column.
* The data type of the measurement quantity is always returned,
* independent of the mode of the values matrix where the column
* belongs to neither the sequence representation of the column.
*
* This method always returns the data type of the measurement quantity
* to which this local column belongs, independent of
* - the value matrix mode of the value matrices this local column
* belongs to
* - the sequence representation of this local column
*
* @throws AoException
* with the following possible error codes:
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @return The data type of the column.
*/
DataType getDataType ()
raises (AoException) ;
/** (13012)

```

```

* Destroy the object on the server. The destructor of the client so
* the server knows this object is not used anymore by the client.
* Access to this object after the destroy method will lead to an
* exception.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*/
void destroy ()
raises (AoException) ;
/** (13013)
* Get the sequence representation of the column.
*
* When the value matrix to which the column belongs is in CALCULATED
* mode the sequence_representation is always explicit.
*
* @throws AoException
* with the following possible error codes:
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @return The sequence representation of the column. This is the
* integer value according the enumeration seq_rep_enum of the
* base model.
*/
T_LONG getSequenceRepresentation ()
raises (AoException) ;
/** (13014)

```

```

* Set the sequence representation of a new column.
*
* @throws AoException
* with the following possible error codes:
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
* AO_BAD_PARAMETER
* AO_TRANSACTION_NOT_ACTIVE
*
* @param sequenceRepresentation The sequence representation. This
* is the integer value according the
* enumeration seq_rep_enum of the base
* model.
*/
void setSequenceRepresentation (
in T_LONG sequenceRepresentation)
raises (AoException) ;
/** (13015)
* Get the generation parameters of the Column.
*
* If no generation parameters exist an empty sequence is returned, its
* type is double, its length is 0.
* When the value matrix to which the column belongs is in CALCULATED
* mode an empty sequence is returned, its type is double, its length
* is 0.
*
* The data type of the generation parameter for the implicit columns
* is the data type given at the measurement quantity. The data type of
* the generation parameters for raw data is always T_DOUBLE.
*
* @throws AoException
* with the following possible error codes:
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM

```

```

* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @return The generation parameters.
*/
TS_Union getGenerationParameters ()
raises (AoException) ;
/** (13016)
* Set the generation parameters.
*
* The data type of the generation parameter for the implicit columns
* must be the data type given at the measurement quantity. The data
* type of the generation parameters for raw data must be always
* T_DOUBLE.
*
* @throws AoException
* with the following possible error codes:
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
* AO_BAD_PARAMETER
* AO_TRANSACTION_NOT_ACTIVE
*
* @param generationParameters The generation parameters.
*/
void setGenerationParameters (
in TS_Union generationParameters)
raises (AoException) ;
/** (13017)
* Get the data type of the raw values.
*
* When the column has no raw values the data type of the measurement
* quantity is returned.
* When the value matrix to which the column belongs is in CALCULATED

```



```

* mode the data type of the measurement quantity returned.
*
* @throws AoException
* with the following possible error codes:
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @return The data type of the raw values.
*/
DataType getRawDataType ()
raises (AoException) ;
}; // Interface Column.
/**
* The ASAM ODS instance element interface. It is allowed to modify the
* instances outside a transaction but recommended to activate a
* transaction before the instances are modified. The modifications to
* instances get permanent when the transaction is committed.
*/
interface InstanceElement {
/** (14001)
* Add an instance attribute to the instance. The instance attribute is
* built as a Name/Value/Unit tuple on the client. This method has to
* copy the data from the client to the server. The name of the
* attribute must be unique.
*
* It is allowed to add an instance attribute outside a transaction but
* it is recommended to activate a transaction.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED

```

```

* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param instAttr The instance attribute to be added.
*/
void addInstanceAttribute (
in NameValueUnit instAttr)
raises (AoException) ;
/** (14002)
* Create a relation between the current and the given instance. Check
* if the application elements of the relation matches the application
* elements of the instances.
*
* It is allowed to modify the instance element outside a transaction
* but it is recommended to activate a transaction.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_INVALID_RELATION
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param relation The application relation.
*
* @param instElem The instance element.
*/
void createRelation (
in ApplicationRelation relation,
in InstanceElement instElem)
raises (AoException) ;
/** (14003)
* Get the application element of the instance element.
*

```

```

* @throws AoException
* with the following possible error codes:
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @return The application element from which the instance element is
* derived.
*/
ApplicationElement getApplicationElement ()
raises (AoException) ;
/** (14004)
* Get the ASAM-Path of the instance element.
*
* @throws AoException
* with the following possible error codes:
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @return The ASAM path to the instance element.
*/
Name getAsamPath ()
raises (AoException) ;
/** (14005)
* Get the Id of the instance element.
*
* @throws AoException
* with the following possible error codes:
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY

```

```

* AO_SESSION_NOT_ACTIVE
*
* @return The Id of the instance element.
*/
T_LONGLONG getId ()
raises (AoException) ;
/** (14006)
* Get the name of the instance element.
*
* @throws AoException
* with the following possible error codes:
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @return The name of the instance element.
*/
Name getName ()
raises (AoException) ;
/** (14007)
* Get the related instances. The application relation and the name of
* the related instances specify the listed instances. The pattern is
* case sensitive and may contain wildcard characters.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_INVALID_RELATION
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param applRel The application relation.

```

```

*
* @param iePattern The name or the search pattern for the related
* instance names.
*
* @return The related instances.
*/
InstanceElementIterator getRelatedInstances (
in ApplicationRelation applRel,
in Pattern iePattern)
raises (AoException) ;
/** (14008)
* Get the list of related instances. The relationship and the name of
* the related instances specify the listed instances. The pattern is
* case sensitive and may contain wildcard characters.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_INVALID_RELATIONSHIP
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param ieRelationship The requested relationship.
*
* @param iePattern The name or the search pattern for the related
* instance names.
*
* @return The related instances.
*/
InstanceElementIterator getRelatedInstancesByRelationship (
in Relationship ieRelationship,
in Pattern iePattern)
raises (AoException) ;
/** (14009)

```

```

* Get the attribute value (name, value and unit) of the given
* attribute of the instance element. This method will not return the
* value of relation attributes, use the method getRelatedInstances.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param attrName The name of the requested attribute.
*
* @return The attribute value.
*/
NameValueUnit getValue (
in Name attrName)
raises (AoException) ;
/** (14010)
* Get the attribute value (value and unit) of the attribute inherited
* from the given base attribute of the instance element. The base name
* is case insensitive and may not contain wildcard characters.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param baseAttrName The base name of the requested attribute.
*
* @return The attribute value.

```

```

*/
NameValueUnit getValueByBaseName (
in Name baseAttrName)
raises (AoException) ;
/** (14011)
* Get the attribute names from the instance element. The attributes
* reserved for a relation are not listed.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_INVALID_ATTRIBUTE_TYPE
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param iaPattern The name or the search pattern for the attribute
* names.
*
* @param aType The requested attribute type.
*
* @return The names of the attributes.
*/
NameSequence listAttributes (
in Pattern iaPattern,
in AttrType aType)
raises (AoException) ;
/** (14012)
* Get the names of the related instances. The application relation and
* the name of the related instances specifies the listed names. The
* pattern is case sensitive and may contain wildcard characters.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER

```

```

* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param ieRelation The application relation.
*
* @param iePattern The name or the search pattern for the related
* instance names.
*
* @return The names of the related instances.
*/
NameIterator listRelatedInstances (
in ApplicationRelation ieRelation,
in Pattern iePattern)
raises (AoException) ;
/** (14013)
* Get the names of the related instances. The relationship and the
* name of the related instances specify the listed names. The pattern
* is case sensitive and may contain wildcard characters.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_INVALID_RELATIONSHIP
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param ieRelationship The requested relationship.
*
* @param iePattern The name or the search pattern for the related
* instance names.
*

```



```

* @return The names of the related instances.
*/
NameIterator listRelatedInstancesByRelationship (
in Relationship ieRelationship,
in Pattern iePattern)
raises (AoException);
/** (14014)
* Remove an instance attribute.
*
* It is allowed to remove the instance attribute outside a transaction
* but it is recommended to activate a transaction.
*
* The application attributes can't be removed.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_FOUND
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param attrName The name of the attribute to be removed.
*/
void removeInstanceAttribute (
in Name attrName)
raises (AoException);
/** (14015)
* Remove the relation between the current instance and the given
* instance. It is necessary to specify the instance element in case of
* n:m relations if not all relations shall be deleted.
*
* It is allowed to modify the instance element outside a transaction
* but it is recommended to activate a transaction.
*

```

```

* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_INVALID_RELATION
* AO_NOT_FOUND
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param applRel The relation to be removed.
*
* @param instElem_nm The instance element for specific delete from
* n:m relations.
*/
void removeRelation (
in ApplicationRelation applRel,
in InstanceElement instElem_nm)
raises (AoException) ;
/** (14016)
* Rename the instance attribute. The application attributes can't be
* renamed.
*
* It is allowed to rename the instance attribute outside a transaction
* but it is recommended to activate a transaction.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_DUPLICATE_NAME
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_FOUND
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE

```

```

*
* @param oldName The old instance attribute name.
*
* @param newName The new instance attribute name.
*/
void renameInstanceAttribute (
in Name oldName,
in Name newName)
raises (AoException) ;
/** (14017)
* Set the name of an instance element.
*
* It is allowed to modify the name of the instance element outside a
* transaction but it is recommended to activate a transaction.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param iaName The instance attribute name.
*/
void setName (
in Name iaName)
raises (AoException) ;
/** (14018)
* Set the value of an application attribute or of an existing instance
* attribute, but it doesn't create a new instance attribute; for this
* purpose use the method addInstancesAttribute of this interface.
*
* It is allowed to modify the attribute value 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 of this interface .
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_DUPLICATE_VALUE
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
* AO_TRANSACTION_NOT_ACTIVE
*
* @param value The value to be set in the instance element.
*/
void setValue (
in NameValueUnit value)
raises (AoException) ;
/** (14019)
* Cast an instance element to a measurement. There are some
* object-oriented languages which do not allow this cast.
*
* @throws AoException
* with the following possible error codes:
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
* AO_INVALID_BASETYPE
*
* @return The instance of type measurement.

```

```

*/
Measurement upcastMeasurement ()
raises (AoException) ;
/** (14020)
 * Cast an instance element to a submatrix. There are some
 * object-oriented languages which do not allow this cast.
 *
 * @throws AoException
 * with the following possible error codes:
 * AO_CONNECTION_LOST
 * AO_IMPLEMENTATION_PROBLEM
 * AO_NOT_IMPLEMENTED
 * AO_NO_MEMORY
 * AO_SESSION_NOT_ACTIVE
 * AO_INVALID_BASETYPE
 *
 * @return The instance of type submatrix.
 */
SubMatrix upcastSubMatrix ()
raises (AoException) ;
/** (14021)
 * Get the attribute value (name, value and unit) of the given
 * attribute of the instance element.
 *
 * @throws AoException
 * with the following possible error codes:
 * AO_BAD_PARAMETER
 * AO_CONNECTION_LOST
 * AO_INCOMPATIBLE_UNITS
 * AO_IMPLEMENTATION_PROBLEM
 * AO_NOT_IMPLEMENTED
 * AO_NO_MEMORY
 * AO_SESSION_NOT_ACTIVE
 *
 * @param attr The name of the requested attribute and the unit of
 * the attribute value.
 *

```

```

* @return The attribute value, value converted to the requested unit.
*/
NameValueUnit getValueInUnit (
in NameUnit attr)
raises (AoException) ;
/** (14022)
* Set a sequences of values of an application attributes but no
* instances attributes.
*
* It is allowed to modify the values of the attributes outside a
* transaction but it is recommended to activate a transaction.
*
* The name of the attribute is specified by the name of the
* NameValueUnit tuple. If the application attribute flag unique is
* set, the uniqueness of the new value is checked.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_DUPLICATE_VALUE
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
* AO_TRANSACTION_NOT_ACTIVE
*
* @param values The sequence of the values to be set at the instance
* element.
*/
void setValueSeq (
in NameValueUnitSequence values)
raises (AoException) ;
/** (14023)
* The given user group the rights should be set for. <rights> defines
* the rights to set or to clear. If the parameter <set> is set to
* 'set', the rights in <rights> are set all others are cleared. If the

```

* parameter <set> is set to 'add', the rights in <rights> are added to
 * the existing rights. If the parameter <set> is set to 'remove', the
 * rights in <rights> are removed from the existing rights.

*

* @throws AoException

* with the following possible error codes:

* AO_BAD_PARAMETER

* AO_CONNECTION_LOST

* AO_IMPLEMENTATION_PROBLEM

* AO_NOT_IMPLEMENTED

* AO_NO_MEMORY

* AO_SESSION_NOT_ACTIVE

* AO_TRANSACTION_NOT_ACTIVE

*

* @param usergroup The user group for which the rights will be
 * modified.

*

* @param rights The new right for the user group. The rights
 * constants are defined in the interface
 * SecurityRights. The interface definition language
 * IDL does not allow to set the values of enumerations
 * thus the constant definitions had to be done in an
 * interface.

*

* @param set What to do with the new right.

*/

```
void setRights (
in InstanceElement usergroup,
in T_LONG rights,
in RightsSet set)
raises (AoException);
```

/** (14024)

* Retrieve access control list information of the given object.

*

* @throws AoException

* with the following possible error codes:

* AO_BAD_PARAMETER

```

* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @return The access control list entries of the given application
* element.
*/
ACLSequence getRights ()
raises (AoException) ;
/** (14025)
* Retrieve access control list information for the initial rights of
* the given object.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @return The access control list entries with the initial rights of
* the given application element.
*/
InitialRightSequence getInitialRights ()
raises (AoException) ;
/** (14026)
* The given user group the initial rights should be set for. <rights>
* defines the rights to set or to clear. If the parameter <set> is set
* to 'set', the rights in <rights> are set all others are cleared. If
* the parameter <set> is set to 'add', the rights in <rights> are
* added to the existing rights. If the parameter <set> is set to
* 'remove', the rights in <rights> are removed from the existing
* rights.

```



```

*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
* AO_TRANSACTION_NOT_ACTIVE
*
* @param usergroup The user group for which the initial rights will
* be modified.
*
* @param rights The new initial rights for the user group. The
* rights constants are defined in the interface
* SecurityRights. The interface definition language
* IDL does not allow to set the values of enumerations
* thus the constant definitions have to be done in the
* interface.
*
* @param refAid The Id of referencing application element for which
* the initial rights will be used. If no refAid is set
* the initial rights will be used for each new
* instance element independent of the application
* element.
*
* @param set What to do with the new initial rights.
*/
void setInitialRights (
in InstanceElement usergroup,
in T_LONG rights,
in T_LONGLONG refAid,
in RightsSet set)
raises (AoException) ;
/** (14027)
* Provides an easy-to-use and effective copy mechanism for instance

```

* elements inside the server. The new instance elements gets a copy of
 * all attribute values and informational relations that are available
 * in the original instance element. The new instance element has the
 * same parent as the original instance element but it does not have
 * references to any children of the original instance element.

*

* @throws AoException

* with the following possible error codes:

* AO_BAD_PARAMETER

* AO_CONNECTION_LOST

* AO_IMPLEMENTATION_PROBLEM

* AO_NOT_IMPLEMENTED

* AO_NO_MEMORY

* AO_SESSION_NOT_ACTIVE

* AO_TRANSACTION_NOT_ACTIVE

*

* @param newName The name of the new instance element. If a new
 * version shall be created this parameter may be NULL
 * to use the same name for the copy. In this case a
 * new version must be provided.

*

* @param newVersion The version of the new instance element. This
 * parameter may be NULL if a new name is provided.

*

* @return The reference to the copied instance element.

*/

InstanceElement shallowCopy (

in T_STRING newName,

in T_STRING newVersion)

raises (AoException) ;

/** (14028)

* Provides an easy-to-use and effective copy 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.

*

* @throws AoException

* with the following possible error codes:

* AO_BAD_PARAMETER

* AO_CONNECTION_LOST

* AO_IMPLEMENTATION_PROBLEM

* AO_NOT_IMPLEMENTED

* AO_NO_MEMORY

* AO_SESSION_NOT_ACTIVE

* AO_TRANSACTION_NOT_ACTIVE

*

* @param newName The name of the new instance element. If a new
 * version shall be created this parameter may be NULL
 * to use the same name for the copy. In this case a
 * new version must be provided.

*

* @param newVersion The version of the new instance element. This
 * parameter may be NULL if a new name is provided.

*

* @return The reference to the copied instance element hierarchy.

*/

InstanceElement deepCopy (

in T_STRING newName,

in T_STRING newVersion)

raises (AoException);

/** (14029)

* Get the sequence of the values of the application or instance
 * attributes, specified by their names. The name sequence can use a
 * pattern (*) for all attributes of the instance element. This means
 * that application as well as instance attributes will be delivered.

*

```

* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param attrNames The names of the attributes to be reported.
*
* @return The sequence of the attribute values.
*/
NameValueUnitSequence getValueSeq (
in NameSequence attrNames)
raises (AoException) ;
/** (14030)
* Destroy the object on the server. The destructor of the client so
* the server knows this object is not used anymore by the client.
* Access to this object after the destroy method will lead to an
* exception.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*/
void destroy ()
raises (AoException) ;
/** (14031)
* 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.

```

```

*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param compIeObj The instance element to compare with.
*
* @return The difference of the Id's. Meaning:
* diff < 0 ElemId of instance is smaller then instance to
* compare with.
* diff == 0 ElemId is identical.
* diff > 0 ElemId of instance is greater then instance to
* compare with.
*/
T_LONGLONG compare (
in InstanceElement compIeObj)
raises (AoException);
/** (14032)
* Create a list with instances which are related to the actual
* instance element. The attribute are given with the name of the
* sequence. The values of the attributes are given in the value
* sequence. The index in the different value sequences match for one
* instance element. The index in the instance element sequence of the
* related instances match for the instance with the same index in the
* value sequence.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED

```

```

* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
* AO_TRANSACTION_NOT_ACTIVE
* AO_INVALID_REQUEST
*
* @param applRel The application relation for witch the related
* instances will be created.
*
* @param attributes The attributes of the new created instances.
*
* @param relatedInstances The list with related instances for
* different application relations.
*
* @return The list with the new created instances.
*/
InstanceElementSequence createRelatedInstances (
in ApplicationRelation applRel,
in NameValueSeqUnitSequence attributes,
in ApplicationRelationInstanceElementSeqSequence relatedInstances)
raises (AoException) ;
}; // Interface InstanceElement.
/**
* The ASAM ODS instance element iterator interface.
*/
interface InstanceElementIterator {
/** (15001)
* Destroy the iterator and free the associated memory.
*
* @throws AoException
* with the following possible error codes:
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*/
void destroy ()

```

```

raises (AoException) ;
/** (15002)
 * Get the total number of elements accessible by the iterator.
 *
 * @throws AoException
 * with the following possible error codes:
 * AO_CONNECTION_LOST
 * AO_IMPLEMENTATION_PROBLEM
 * AO_NOT_IMPLEMENTED
 * AO_NO_MEMORY
 * AO_SESSION_NOT_ACTIVE
 *
 * @return The number of elements accessible by the iterator.
 */
T_LONG getCount ()
raises (AoException) ;
/** (15003)
 * Get the next n elements from the sequence.
 *
 * @throws AoException
 * with the following possible error codes:
 * AO_BAD_PARAMETER
 * AO_CONNECTION_LOST
 * AO_IMPLEMENTATION_PROBLEM
 * AO_NOT_IMPLEMENTED
 * AO_NO_MEMORY
 * AO_SESSION_NOT_ACTIVE
 *
 * @param how_many The number of requested elements.
 *
 * @return The next n instance elements from the instance sequence.
 */
InstanceElementSequence nextN (
in T_LONG how_many)
raises (AoException) ;
/** (15004)
 * Get the next element from the sequence.

```

```

*
* @throws AoException
* with the following possible error codes:
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @return The next instance element from the instance sequence.
*/
InstanceElement nextOne ()
raises (AoException) ;
/** (15005)
* Reset the pointer in the element sequence to the first element.
*
* @throws AoException
* with the following possible error codes:
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*/
void reset ()
raises (AoException) ;
}; // Interface InstanceElementIterator.
/**
* The ASAM ODS measurement interface. The interface Measurement extends
* the interface InstanceElement. Thus all methods of the interface
* InstanceElement do also work here.
*/
interface Measurement : InstanceElement {
/** (16001)
* Create a submatrix link. The submatrix link is only valid in the
* current session. When the session is closed the submatrix link will
* be destroyed.

```



```

*
* @throws AoException
* with the following possible error codes:
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @return The new submatrix link.
*/
SMatLink createSMatLink ()
raises (AoException) ;
/** (16002)
* Get the list of the submatrix links .
*
* @throws AoException
* with the following possible error codes:
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @return The available submatrix links.
*/
SMatLinkSequence getSMatLinks ()
raises (AoException) ;
/** (16003)
* Get the value matrix of a measurement.
* By calling this method, the value matrix mode of the returned value
* matrix will be taken from the context variable 'VALUEMATRIX_MODE'.
* For more information especially on different modes of the value
* matrix, see description at the method getValueMatrixInMode () of this
* interface.
*
* @throws AoException

```

```

* with the following possible error codes:
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
* AO_INVALID_VALUEMATRIX_STRUCTURE
*
* @return The value matrix.
*/
ValueMatrix getValueMatrix ()
raises (AoException) ;
/** (16004)
* Remove a submatrix link.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_FOUND
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param smLink The sub matrix link to be removed.
*/
void removeSMatLink (
in SMatLink smLink)
raises (AoException) ;
/** (16005)
* Get the value matrix of a measurement in the requested mode.
*
* 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 value matrix due to the data of the
* measurement
* - if there are no independent column
* - if there are different independent columns
* - if there are sub matrices 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) .
*
* In the CALCULATED mode the server returns the calculated values for
* implicit and raw values. The server returns at each measurement
* point the explicit value as the value for the implicit channels. The
* raw values are calculated to the explicit values.
*
* In the STORAGE mode the server handles the values of the localcolumn
* as they are in the storage. Write is only allowed when the value
* matrix is in the STROAGE mode.
*
* @throws AoException
* with the following possible error codes:
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
* AO_INVALID_VALUEMATRIX_STRUCTURE
*
* @param vmMode The requested mode of the value matrix.
*
* @return The value matrix.
*/

```

```

ValueMatrix getValueMatrixInMode (
in ValueMatrixMode vmMode)
raises (AoException) ;
}; // Interface Measurement.
/**
* The ASAM ODS name iterator interface.
*/
interface NameIterator {
/** (17001)
* Destroy the iterator and free the associated memory.
*
* @throws AoException
* with the following possible error codes:
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*/
void destroy ()
raises (AoException) ;
/** (17002)
* Get the total number of elements accessible by the iterator.
*
* @throws AoException
* with the following possible error codes:
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @return The number of elements accessible by the iterator.
*/
T_LONG getCount ()
raises (AoException) ;
/** (17003)

```

```

* Get the next n elements from the sequence.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param how_many The number of requested elements.
*
* @return The next n names from the name sequence.
*/
NameSequence nextN (
in T_LONG how_many)
raises (AoException) ;
/** (17004)
* Get the next element from the sequence.
*
* @throws AoException
* with the following possible error codes:
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @return The next name from the name sequence.
*/
Name nextOne ()
raises (AoException) ;
/** (17005)
* Reset the pointer in the element sequence to the first element.
*
* @throws AoException

```

```

* with the following possible error codes:
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*/

void reset ()
raises (AoException) ;
}; // Interface NameIterator.
/**
* The ASAM ODS name-value iterator interface.
*/
interface NameValueIterator {
/** (18001)
* Destroy the iterator and free the associated memory.
*
* @throws AoException
* with the following possible error codes:
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*/
void destroy ()
raises (AoException) ;
/** (18002)
* Get the total number of elements accessible by the iterator.
*
* @throws AoException
* with the following possible error codes:
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE

```

```

*
* @return The number of elements accessible by the iterator.
*/
T_LONG getCount ()
raises (AoException) ;
/** (18003)
* Get the next n elements from the sequence.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param how_many The number of requested elements.
*
* @return The next n name-value pairs from the name-value pair
* sequence.
*/
NameValueSequence nextN (
in T_LONG how_many)
raises (AoException) ;
/** (18004)
* Get the next element from the sequence.
*
* @throws AoException
* with the following possible error codes:
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @return The next name-value pair from the name-value pair sequence.

```

```

*/
NameValue nextOne ()
raises (AoException) ;
/** (18005)
* Reset the pointer in the element sequence to the first element.
*
* @throws AoException
* with the following possible error codes:
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*/
void reset ()
raises (AoException) ;
}; // Interface NameValueIterator.
/**
* The ASAM ODS name-value-unit iterator interface. This interface is
* identical with the NameValueUnitIdIterator, except the unit is given as
* a string instead of an Id.
*/
interface NameValueUnitIterator {
/** (19001)
* Destroy the iterator and free the associated memory.
*
* @throws AoException
* with the following possible error codes:
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*/
void destroy ()
raises (AoException) ;
/** (19002)

```



```

* Get the total number of elements accessible by the iterator.
*
* @throws AoException
* with the following possible error codes:
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @return The number of elements accessible by the iterator.
*/
T_LONG getCount ()
raises (AoException) ;
/** (19003)
* Get the next n elements from the sequence.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param how_many The number of requested elements.
*
* @return The next n name-value-unit tuples from the name-value-unit
* tuple sequence.
*/
NameValueUnitSequence nextN (
in T_LONG how_many)
raises (AoException) ;
/** (19004)
* Get the next element from the sequence.
*

```

```

* @throws AoException
* with the following possible error codes:
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @return The next name-value-unit tuple from the name-value-unit
* tuple sequence.
*/
NameValueUnit nextOne ()
raises (AoException) ;
/** (19005)
* Reset the pointer in the element sequence to the first element.
*
* @throws AoException
* with the following possible error codes:
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*/
void reset ()
raises (AoException) ;
}; // Interface NameValueUnitIterator.
/**
* The ASAM ODS submatrix link interface.
*/
interface SMatLink {
/** (20001)
* Get the link or build type.
*
* @throws AoException
* with the following possible error codes:
* AO_CONNECTION_LOST

```

```

* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @return The link type.
*/
BuildUpFunction getLinkType ()
raises (AoException) ;
/** (20002)
* Get the ordinal or sequence number
*
* @throws AoException
* with the following possible error codes:
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @return The sequence number.
*/
T_LONG getOrdinalNumber ()
raises (AoException) ;
/** (20003)
* Get the first submatrix of the link.
*
* @throws AoException
* with the following possible error codes:
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @return The first submatrix of the link.
*/

```

```

SubMatrix getSMat1 ()
raises (AoException) ;
/** (20004)
 * Get the bind columns of the first submatrix used in the link (e.g.
 * Time) .
 *
 * @throws AoException
 * with the following possible error codes:
 * AO_CONNECTION_LOST
 * AO_IMPLEMENTATION_PROBLEM
 * AO_NOT_IMPLEMENTED
 * AO_NO_MEMORY
 * AO_SESSION_NOT_ACTIVE
 *
 * @return The columns of the first submatrix.
 */
ColumnSequence getSMat1Columns ()
raises (AoException) ;
/** (20005)
 * Get the second submatrix of the link.
 *
 * @throws AoException
 * with the following possible error codes:
 * AO_CONNECTION_LOST
 * AO_IMPLEMENTATION_PROBLEM
 * AO_NOT_IMPLEMENTED
 * AO_NO_MEMORY
 * AO_SESSION_NOT_ACTIVE
 *
 * @return The second submatrix of the link.
 */
SubMatrix getSMat2 ()
raises (AoException) ;
/** (20006)
 * Get the bind columns of the second submatrix used in the link (e.g.
 * Time) .
 *
```

```

* @throws AoException
* with the following possible error codes:
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @return The columns of the second submatrix.
*/
ColumnSequence getSMat2Columns ()
raises (AoException) ;
/** (20004)
* Get the bind columns of the first submatrix used in the link (e.g.
* Time) .
*
* @throws AoException
* with the following possible error codes:
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @return The columns of the first submatrix.
*/
ColumnSequence getSMat1Columns ()
raises (AoException) ;
/** (20005)
* Get the second submatrix of the link.
*
* @throws AoException
* with the following possible error codes:
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY

```

```

* AO_SESSION_NOT_ACTIVE
*
* @return The second submatrix of the link.
*/
SubMatrix getSMat2 ()
raises (AoException) ;
/** (20006)
* Get the bind columns of the second submatrix used in the link (e.g.
* Time) .
*
* @throws AoException
* with the following possible error codes:
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @return The columns of the second submatrix.
*/
ColumnSequence getSMat2Columns ()
raises (AoException) ;
/** (20007)
* Set the build or link type.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_INVALID_BUILDUP_FUNCTION
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param linkType The requested build-up function.
*/

```

```

void setLinkType (
in BuildUpFunction linkType)
raises (AoException) ;
/** (20008)
* Set the ordinal or sequence number.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_INVALID_ORDINALNUMBER
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param ordinalNumber The sequence number.
*/
void setOrdinalNumber (
in T_LONG ordinalNumber)
raises (AoException) ;
/** (20009)
* Set the first submatrix of the link.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_INVALID_SUBMATRIX
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param subMat1 The first sub matrix of the sub matrix link.
*/
void setSMat1 (

```

```

in SubMatrix subMat1)
raises (AoException) ;
/** (20010)
* Set the bind columns of the first submatrix used in the link (e.g.
* Time) .
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_INVALID_COLUMN
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param columns The column sequence of the sub matrix.
*/
void setSMat1Columns (
in ColumnSequence columns)
raises (AoException) ;
/** (20011)
* Set the second submatrix of the link.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_INVALID_SUBMATRIX
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param subMat2 The second sub matrix of the sub matrix link.
*/
void setSMat2 (

```



```

in SubMatrix subMat2)
raises (AoException);
/** (20012)
* Set the bind columns of the second submatrix used in the link (e.g.
* Time). If there is more than one column bound the column sequence
* must be identical with the column sequence of the first submatrix.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_INVALID_COLUMN
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param columns The column sequence of the sub matrix.
*/
void setSMat2Columns (
in ColumnSequence columns)
raises (AoException);
}; // Interface SMatLink.
/**
* The ASAM ODS submatrix interface. The instances of the submatrix can be
* accessed via the method getRelatedInstances of the interface
* Measurement. The interface SubMatrix extends the interface
* InstanceElement. Thus all methods of the interface InstanceElement do
* also work here.
*/
interface SubMatrix : InstanceElement {
/** (21001)
* Get the columns of the submatrix. The column is not inherited from
* the InstanceElement interface. This is the only way to get a column.
* The columns are used in the SMatLink interface to build the value
* matrix. The pattern is case sensitive and may contain wildcard
* characters.

```

```

*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param colPattern The name or the search pattern for the column
* names.
*
* @return The columns of the submatrix.
*/
ColumnSequence getColumns (
in Pattern colPattern)
raises (AoException) ;
/** (21002)
* Get a value matrix of the submatrix.
* By calling this method, the value matrix mode of the returned value
* matrix will be taken from the context variable 'VALUEMATRIX_MODE'.
* For more information especially on different modes of the value
* matrix, see description at the method getValueMatrixInMode () of this
* interface.
*
* @throws AoException
* with the following possible error codes:
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_INVALID_SMATLINK
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @return The value matrix.
*/

```

```

ValueMatrix getValueMatrix ()
raises (AoException) ;
/** (21003)
* Get the names of the columns of the submatrix. The name sequence is
* identical with the names of the related instances. The pattern is
* case sensitive and may contain wildcard characters.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param colPattern The name or the search pattern for the column
* names.
*
* @return The column names of the submatrix.
*/
NameSequence listColumns (
in Pattern colPattern)
raises (AoException) ;
/** (21004)
* Get the value matrix of a submatrix in the requested mode.
*
* The server doesn't check any independent columns. The columns are
* only the local columns from this submatrix. The flags in the
* TS_ValueSeq are the flags of the localcolumn. If no flags are given
* there are no flags available, the client have to check the
* global_flag.
*
* In the CALCULATED mode the server returns the calculated values for
* implicit and raw values. The server returns at each measurement
* point the explicit value as the value for the implicit channels. The
* raw values are calculated to the explicit values.

```

```

*
* In the STORAGE mode the server handles the values of the localcolumn
* as they are in the storage. Write is only allowed when the value
* matrix is in the STROAGE mode. The values are exactly from this
* submatrix. The raw / implicit values are returned.
*
* @throws AoException
* with the following possible error codes:
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param vmMode The requested mode of the value matrix.
*
* @return The value matrix.
*/
ValueMatrix getValueMatrixInMode (
in ValueMatrixMode vmMode)
raises (AoException) ;
}; // Interface SubMatrix.
/**
* The ASAM ODS value matrix interface. Value matrix is an interface used
* by Measurement and SubMatrix to handle vectors of values.
*/
interface ValueMatrix {
/** (22001)
* Get the columns of the value matrix no matter whether the column is
* dependent or independent. The pattern is case sensitive and may
* contain wildcard characters.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM

```

```

* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param colPattern The name or the search pattern for the column
* names.
*
* @return The columns of the value matrix, no matter whether the
* column is dependent, independent or scaling
*/
ColumnSequence getColumns (
in Pattern colPattern)
raises (AoException) ;
/** (22002)
* Get the column count of the value matrix.
*
* @throws AoException
* with the following possible error codes:
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @return The number of columns of the value matrix.
*/
T_LONG getColumnCount ()
raises (AoException) ;
/** (22003)
* Get the independent columns of the value matrix. The independent
* columns are the columns used to build the value matrix.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM

```

```

* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param colPattern The name or the search pattern for the
* independent column name.
*
* @return The independent column of the value matrix.
*/
ColumnSequence getIndependentColumns (
in Pattern colPattern)
raises (AoException) ;
/** (22004)
* Get the row count of the value matrix.
*
* @throws AoException
* with the following possible error codes:
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @return The number of rows of the value matrix.
*/
T_LONG getRowCount ()
raises (AoException) ;
/** (22005)
* Get a measurement point of the value matrix. The parameter meaPoint
* specifies the row of the matrix. The iterator allows to access all
* elements in the row.
*
* The server behavior depends on the mode of the value matrix.
* Value matrix mode 'CALCULATED':
* In case 'sequence_representation' of the corresponding local column
* is one of the entries 'raw_linear', 'raw_polynomial',
* 'raw_linear_external', 'raw_polynomial_external',

```

```

* 'raw_linear_calibrated', or 'raw_linear_calibrated_external', the
* server will first calculate the physical values from raw values and
* generation parameters, before it returns them to the requesting
* client.
* Value matrix mode 'STORAGE':
* In case 'sequence_representation' of the corresponding local column
* is one of the entries 'raw_linear', 'raw_polynomial',
* 'raw_linear_external', 'raw_polynomial_external',
* 'raw_linear_calibrated', or 'raw_linear_calibrated_external', the
* server will return the raw values of the local column.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param meaPoint The measurement point.
*
* @return The requested measurement point.
*/
NameValueUnitIterator getValueMeaPoint (
in T_LONG meaPoint)
raises (AoException);
/** (22006)
* Get the values or a part of values of the column from the value
* matrix. The parameter column specifies from which column the values
* will be returned. The startPoint and count specify the part of the
* vector. A startPoint = 0 and count = rowCount will return the entire
* vector. When startPoint >= rowCount an exception is thrown. If
* startPoint + count > 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.
*
* The server behavior depends on the mode of the value matrix.
* Value matrix mode 'CALCULATED':
* In case 'sequence_representation' of the corresponding local column
* is one of the entries 'raw_linear', 'raw_polynomial',
* 'raw_linear_external', 'raw_polynomial_external',
* 'raw_linear_calibrated', or 'raw_linear_calibrated_external', the
* server will first calculate the physical values from raw values and
* generation parameters, before it returns them to the requesting
* client.
* Value matrix mode 'STORAGE':
* In case 'sequence_representation' of the corresponding local column
* is one of the entries 'raw_linear', 'raw_polynomial',
* 'raw_linear_external', 'raw_polynomial_external',
* 'raw_linear_calibrated', or 'raw_linear_calibrated_external', the
* server will return the raw values of the local column.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_INVALID_COLUMN
* AO_INVALID_COUNT
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param col The column to retrieve the values from.
*
* @param startPoint The starting point in the column.
*
* @param count The number of points to be retrieved.
*
* @return The requested column values of the value matrix.

```



```

*/
TS_ValueSeq getValueVector (
in Column col,
in T_LONG startPoint,
in T_LONG count)
raises (AoException);
/** (22007)
* Get the names of the columns of the value matrix no matter whether
* the column is dependent or independent. The pattern is case
* sensitive and may contain wildcard characters.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param colPattern The name or the search pattern for the column
* names.
*
* @return The column names of the value matrix, no matter whether the
* column is dependent, independent or scaled by another one.
*/
NameSequence listColumns (
in Pattern colPattern)
raises (AoException);
/** (22008)
* Get the names of the independent columns of the value matrix. The
* independent columns are the columns used to build the value matrix.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST

```

```

* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param colPattern The name or the search pattern for the
* independent column name.
*
* @return The names of the independent columns of the value matrix.
*/
NameSequence listIndependentColumns (
in Pattern colPattern)
raises (AoException) ;
/** (22009)
* Remove the values of the columns at a given measurement point.
* Remove the number of points of the given column. If the count is 0
* all points until the end of the column are removed.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_INVALID_COUNT
* AO_NOT_FOUND
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param columnNames The columns from which the measurement points
* are to be removed.
*
* @param meaPoint The measurement point to be removed.
*
* @param count The number of points to be removed from each column.
*/
void removeValueMeaPoint (

```

```

in NameSequence columnNames,
in T_LONG meaPoint,
in T_LONG count)
raises (AoException);
/** (22010)
* Remove the values from a value vector. Beginning at startPoint the
* number of values specified in count are removed. If count is 0 all
* values from the startPoint until the end of the vector are removed.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_INVALID_COLUMN
* AO_INVALID_COUNT
* AO_NOT_FOUND
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param col The column from which the values are to be removed.
*
* @param startPoint The starting point for the value removal.
*
* @param count The number of points to be removed from the column.
*/
void removeValueVector (
in Column col,
in T_LONG startPoint,
in T_LONG count)
raises (AoException);
/** (22011)
* Create or modify a measurement point of a value matrix.
* The sequence of name values specifies the names of the column with
* the new values.
* The meaning of the parameter setType is:

```

```

*
* INSERT Insert the values at meaPoint, the current values at
* meaPoint are moved to the end of the new inserted values.
*
* APPEND The value of meaPoint is ignored, the values are
* appended at the end of the current values.
*
* UPDATE Update or modify the values at meaPoint, the current
* values are overwritten. If meaPoint is bigger than the number of
* values in the vector, the measurement point is
* automatically appended.
*
* The names of the columns have to exist.
*
* When a client creates a ValueMatrix based on AoMeasurement this
* methods behaves as follows:
* The server checks the sub matrices and creates all necessary
* instances of AoSubmatrix, AoLocalColumn and AoMeasurementQuantity.
* The values of the name attribute of AoSubmatrix must be generated by
* the server. The value will be equal to the value of the attribute ID
* (converted to DT_STRING) . Missing instances of AoMeasurementQuantity
* will be created by the server, too.
*
* The mandatory attributes will get the following default values:
* Name supplied by client
* Datatype copied from AoQuantity.default_datatype
* Typesize copied from AoQuantity.default_typesize
* Interpolation no interpolation
* Rank copied from AoQuantity.default_rank
* Dimension copied from AoQuantity.default_dimension
*
* The server takes the value of the channel (supplied by client) and
* looks up the corresponding instance in AoQuantity using the
* attribute default_meq_name.
*
* @throws AoException
* with the following possible error codes:

```

```

* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_IS_MEASUREMENT_MATRIX
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param set The set type.
*
* @param meaPoint The measurement point.
*
* @param value The values to be inserted.
*/
void setValueMeaPoint (
in SetType set,
in T_LONG meaPoint,
in NameValueSequence value)
raises (AoException) ;
/** (22012)
* Create or modify a value vector in a value matrix.
*
* It is allowed to modify the values of the value matrix 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 sub matrices and creates all necessary
- * instances of AoSubmatrix, AoLocalColumn and AoMeasurementQuantity.
- * The values of the name attribute of AoSubmatrix must be generated by
- * the server. The value will be equal to the value of the attribute ID
- * (converted to DT_STRING) . Missing instances of AoMeasurementQuantity
- * will be created by the server,
- * too.
- *
- * The mandatory attributes will get the following default values:
- * Name supplied by client
- * Datatype copied from AoQuantity.default_datatype
- * Typesize copied from AoQuantity.default_typesize
- * Interpolation no interpolation
- * Rank copied from AoQuantity.default_rank
- * Dimension copied from AoQuantity.default_dimension
- *
- * The server takes the value of the channel (supplied by client) and
- * looks up the corresponding instance in AoQuantity using the
- * attribute default_meq_name.
- *
- * @throws AoException
- * with the following possible error codes:
- * AO_BAD_PARAMETER
- * AO_CONNECTION_LOST
- * AO_IMPLEMENTATION_PROBLEM
- * AO_INVALID_COLUMN
- * AO_INVALID_SET_TYPE

```

* AO_IS_MEASUREMENT_MATRIX
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param col The column whose values are to be set.
*
* @param set The set type.
*
* @param startPoint The starting point for the new values.
*
* @param value The values to be inserted.
*/
void setValueVector (
in Column col,
in SetType set,
in T_LONG startPoint,
in TS_ValueSeq value)
raises (AoException) ;
/** (22013)
* Create or modify a number of value vectors in a value matrix.
*
* It is allowed to modify the values of the value matrix 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 sub matrices 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 data type identical with the
* raw_datatype, any other data type will cause an AO_INVALID_DATATYPE
* exception.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_INVALID_COLUMN
* AO_INVALID_SET_TYPE
* AO_INVALID_DATATYPE
* AO_IS_MEASUREMENT_MATRIX
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param set The set type.
*
* @param startPoint The measurement point.
*
* @param value The values to be inserted.
*/
void setValue (
in SetType set,
in T_LONG startPoint,
in NameValueSeqUnitSequence value)
raises (AoException);
/** (22014)
* Add a column to the value matrix. It is only allowed to create a
* value vector if the value matrix is created from a submatrix.

```

```

*
* It is allowed to add a column outside a transaction but it is
* recommended to activate a transaction.
*
* The data is made permanent on transaction commit. Until the
* transaction is committed or until the access to the measurement
* point of the value matrix it is allowed to have different number of
* values in the different value vectors. After the new column is
* added, it is possible to set the values of the column.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_INVALID_NAME
* AO_INVALID_SET_TYPE
* AO_IS_MEASUREMENT_MATRIX
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
* AO_TRANSACTION_NOT_ACTIVE
*
* @param newColumn The name and unit of the column to add.
*
* @return The new column.
*/
Column addColumn (
in NameUnit newColumn)
raises (AoException) ;
/** (22015)
* Get the names of the scaling columns of the value matrix.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST

```

```

* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param colPattern The name or the search pattern for the scaling
* column name.
*
* @return The names of the scaling columns of the value matrix.
*/
NameSequence listScalingColumns (
in Pattern colPattern)
raises (AoException) ;
/** (22016)
* Get the scaling column of the value matrix.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param colPattern The name or the search pattern for the scaling
* column name.
*
* @return The scaling columns of the value matrix.
*/
ColumnSequence getScalingColumns (
in Pattern colPattern)
raises (AoException) ;
/** (22017)
* List the names of the columns, which are scaled by the given column.
*
* @throws AoException

```

```

* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
* AO_NO_SCALING_COLUMN
*
* @param scalingColumn The scaling column.
*
* @return The names of the columns which are scaled by the given
* input column.
*/
NameSequence listColumnsScaledBy (
in Column scalingColumn)
raises (AoException) ;
/** (22018)
* Get the columns which are scaled by the given column.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
* AO_NO_SCALING_COLUMN
*
* @param scalingColumn The scaling column.
*
* @return The columns which are scaled by the given input column.
*/
ColumnSequence getColumnsScaledBy (
in Column scalingColumn)
raises (AoException) ;

```

```

/** (22019)
 * Add a column to the value matrix. It is only allowed to create a
 * value vector if the value matrix is created from a submatrix. The
 * column will be scaled by the given scaling column.
 *
 * It is allowed to add a column outside a transaction but it is
 * recommended to activate a transaction.
 *
 * The data is made permanent on transaction commit. Until the
 * transaction is committed or until the access to the measurement
 * point of the value matrix it is allowed to have different number of
 * values in the different value vectors. After the new column is
 * added, it is possible to set the values of the column.
 *
 * @throws AoException
 * with the following possible error codes:
 * AO_BAD_PARAMETER
 * AO_CONNECTION_LOST
 * AO_IMPLEMENTATION_PROBLEM
 * AO_INVALID_NAME
 * AO_INVALID_SET_TYPE
 * AO_IS_MEASUREMENT_MATRIX
 * AO_NOT_IMPLEMENTED
 * AO_NO_MEMORY
 * AO_SESSION_NOT_ACTIVE
 * AO_NO_SCALING_COLUMN
 * AO_TRANSACTION_NOT_ACTIVE
 *
 * @param newColumn The name and unit of the column to add.
 *
 * @param scalingColumn The scaling column.
 *
 * @return The new column.
 */
Column addColumnScaledBy (
in NameUnit newColumn,
in Column scalingColumn)

```

```

raises (AoException) ;
/** (22020)
* Destroy the object on the server. The destructor of the client, so
* the server knows this object is not used anymore by the client.
* Access to this object after the destroy method will lead to an
* exception.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
* AO_TRANSACTION_NOT_ACTIVE
*/
void destroy ()
raises (AoException) ;
/** (22021)
* Get the values of different columns of the value matrix.
*
* The server behavior depends on the mode of the value matrix.
* Value matrix mode 'CALCULATED':
* In case 'sequence_representation' of the corresponding local column
* is one of the entries 'raw_linear', 'raw_polynomial',
* 'raw_linear_external', 'raw_polynomial_external',
* 'raw_linear_calibrated', or 'raw_linear_calibrated_external', the
* server will first calculate the physical values from raw values and
* generation parameters, before it returns them to the requesting
* client.
* Value matrix mode 'STORAGE':
* In case 'sequence_representation' of the corresponding local column
* is one of the entries 'raw_linear', 'raw_polynomial',
* 'raw_linear_external', 'raw_polynomial_external',
* 'raw_linear_calibrated', or 'raw_linear_calibrated_external', the
* server will return the raw values of the local column.

```

```

*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param columns The requested columns.
*
* @param startPoint The starting point in the column.
*
* @param count The number of points to be retrieved. 0 mean until
* end of column.
*
* @return The values of the different columns. The name of the return
* structure corresponds with the name of the column. The unit
* corresponds with the unit of the column. The order of the
* results might not match the order in the requested
* sequence.
*/
NameValueSeqUnitSequence getValue (
in ColumnSequence columns,
in T_LONG startPoint,
in T_LONG count)
raises (AoException) ;
/** (22022)
* Get the current mode of the value matrix.
*
* @throws AoException
* with the following possible error codes:
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY

```

```

* AO_SESSION_NOT_ACTIVE
*
* @return The mode of the value matrix
*/
ValueMatrixMode getMode ()
raises (AoException) ;
}; // Interface ValueMatrix.
/**
* The ASAM ODS name-value-unitId iterator interface. This interface is
* identical with the NameValueUnitIterator, except the unit is given as
* an Id instead of a string.
*/
interface NameValueUnitIdIterator {
/** (23001)
* Destroy the iterator and free the associated memory.
*
* @throws AoException
* with the following possible error codes:
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*/
void destroy ()
raises (AoException) ;
/** (23002)
* Get the total number of elements accessible by the iterator.
*
* @throws AoException
* with the following possible error codes:
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*

```



```

* @return The number of elements accessible by the iterator.
*/
T_LONG getCount ()
raises (AoException) ;
/** (23003)
* Get the next n elements from the sequence.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param how_many The number of requested elements.
*
* @return The next n name-value-unit tuples from the name-value-unit
* tuple sequence.
*/
NameValueSeqUnitId nextN (
in T_LONG how_many)
raises (AoException) ;
/** (23004)
* Get the next element from the sequence.
*
* @throws AoException
* with the following possible error codes:
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @return The next name-value-unitId tuple from the name-value-unitId
* tuple sequence.

```

```

*/
NameValueUnitId nextOne ()
raises (AoException) ;
/** (23005)
* Reset the pointer in the element sequence to the first element.
*
* @throws AoException
* with the following possible error codes:
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*/
void reset ()
raises (AoException) ;
}; // Interface NameValueUnitIdIterator.
/**
* The application element access. The ApplElemAccess interface (RPC API
* GetVal, PutVal, GetInstRef, ...) is responsible to retrieve instances
* of application elements. Currently available query methods are:
* - query with attribute and value
* - query with attribute and values (not) in set
* - sorted results
* - grouped results including having condition
*/
interface ApplElemAccess {
/** (24001)
* Perform the Query.
*
* The number of different application elements which are requested are
* exactly defined by the definition of the query given in the field
* anuSeq of the QueryStructure. The number of attributes for each
* application element is also given in the definition. The number of
* matching instances (their attributes) is not defined by the query
* and can be a large amount. Therefore only one iterator for the
* attribute values are defined.

```

*
 * 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.
 *
 * In case an attribute of blob type is among the requested attributes,
 * the content of the return structure will depend on the number of
 * instances that are returned:
 * - In case attributes of only one instance are returned (either
 * because 'how_many' was set to 1 or there is only one instance
 * available at the server), the return structure contains the blob
 * itself.
 * -In case attributes of more than one instance are returned, the
 * return structure does not contain the blobs; instead it contains the
 * headers of the respective blobs.
 *
 * @throws AoException
 * with the following possible error codes:
 * AO_BAD_PARAMETER
 * AO_CONNECTION_LOST
 * AO_IMPLEMENTATION_PROBLEM
 * AO_NOT_IMPLEMENTED
 * AO_NO_MEMORY
 * AO_SESSION_NOT_ACTIVE
 *
 * @param aoq The query definition.
 *

```

* @param how_many Maximum number of values in each result set. Valid
* arguments are:
* how_many = 0 : report all found values,
* how_many > 0 : report a maximum number of values.
*
* @return The result set with the requested attribute values.
*/
ElemResultSetSequence getInstances (
in QueryStructure aoq,
in T_LONG how_many)
raises (AoException) ;
/** (24002)
* Get related instances (Id) . This method returns a sequence of
* related instances.
*
* The relation name specifies the relation given in the
* ApplStructValue. The aid of the ElemId and the relName defines the
* unique relation and the target application element.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param elem Original instance.
*
* At the RPC API this information was stored in the
* field elemId of the structure GetInstRefReq and the
* request AOP_GetInstRef.
*
* @param relName The relation name.
*
* At the RPC API this information was stored in the

```

```

* field refName of the structure GetInstRefReq and
* the request AOP_GetInstRef.
*
* @return The list of the Id of the related instances.
*/
S_LONGLONG getRelInst (
in ElemId elem,
in Name relName)
raises (AoException) ;
/** (24003)
* Set the instance reference.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param elem 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.
*
* @param relName 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.
*
* @param instIds 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 then one relation.
*
* @param type The type of the modification, insert, update or
* remove.
*
* At the RPC API this information was stored in the
* field on off of the structure SetInstRefReq and the
* request AOP_SetInstRef.
*/
void setRelInst (
in ElemId elem,
in Name relName,
in S_LONGLONG instIds,
in SetType type)
raises (AoException) ;
/** (24004)
* Create instance elements of an application element.
* The application element is specified by the AID of input structure.
*
* The attribute names are specified by the name of the input
* structure.
* The values of one instance element are specified in the valueseq of
* the input structure.
* You can create several instance elements in one call by filling the
* valueseq of the input structure.
* The same index in the valueseq corresponds to the attributes values
* of one instance element. This method returns a sequence of Id's, the
* order is related to the order of instance element specified in the
* input structure. In case of inheritance, the method supports only
* instances of same subtype per call. The returned Id's are the Id's
* of the related super type instances.
*
* The client must supply all mandatory attributes and references
* within one single method call; otherwise the object cannot be made
* persistent by the server in the database without the risk of
* violating any database constraint.

```

```

*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param val 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.
*
* @return List with the Id's of the new created instances.
*/
ElemIdSequence insertInstances (
in AIDNameValueSeqUnitIdSequence val)
raises (AoException) ;
/** (24005)
* Update 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 super type 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.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param val The sequence of attributes and their values. At least
* one of the attribute values sequence must be a sequence
* with the Id.
*
* At the RPC API this information was stored in the
* fields elemId and nvSeq of the structure PutValReq and
* the request AOP_PutValReq.
*/
void updateInstances (
in AIDNameValueSeqUnitIdSequence val)
raises (AoException) ;
/** (24006)
* Delete instances from an application element. In case of inherited
* application elements the Id of the super type has to be specified.
*
* This method can be used to delete several instances of the same
* application element, the method removeInstances remove one instance
* of an application element with the children of the instance element.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST

```



```

* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param aid The application element Id.
*
* @param instIds 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.
*/
void deleteInstances (
in T_LONGLONG aid,
in S_LONGLONG instIds)
raises (AoException) ;
/** (24007)
* Get the value matrix of a measurement or a submatrix. If the value
* matrix will be built up with special submatrix link, use the
* interface Measurement.
* By calling this method, the value matrix mode of the returned value
* matrix will be taken from the context variable 'VALUEMATRIX_MODE'.
* For more information especially on different modes of the value
* matrix, see description at the method getValueMatrixInMode () of the
* interfaces 'Measurement' and 'SubMatrix'.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
* AO_INVALID_BASETYPE
*

```

```

* @param elem The element id. The aid has to be the application
* element Id of the AoMeasurement or AoSubmatrix.
*
* @return The value matrix
*/
ValueMatrix getValueMatrix (
in ElemId elem)
raises (AoException) ;
/** (24008)
* Set the ACL information on some application element-attribute
* defined by <aid> and <attr_name>. The <usergroup_id> defines the
* user group the rights should be set for. <rights> defines the rights
* to set or to clear. If the parameter <set> is set to 'set', the
* rights in <rights> are set all others are cleared. If the parameter
* <set> is set to 'add', the rights in <rights> are added to the
* existing rights. If the parameter <set> is set to 'remove', the
* rights in <rights> are removed from the existing rights. Restriction
* for the model: only one application element of the type AoUserGroup
* is allowed.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
* AO_TRANSACTION_NOT_ACTIVE
*
* @param aid The Id of the application element.
*
* @param attrName The name of the attribute.
*
* @param usergroupId The user group to set the rights for.
*
* @param rights The new right for the user group.

```

```

*
* @param set What to do with the new right.
*/
void setAttributeRights (
in T_LONGLONG aid,
in T_STRING attrName,
in T_LONGLONG usergroupId,
in T_LONG rights,
in RightsSet set)
raises (AoException) ;
/** (24009)
* Set the ACL information on some application element defined by
* <aid>. The <usergroup_id> defines the user group the rights should
* be set for. <rights> defines the rights to set or to clear. If the
* parameter <set> is set to 'set', the rights in <rights> are set all
* others are cleared. If the parameter <set> is set to 'add', the
* rights in <rights> are added to the existing rights. If the
* parameter <set> is set to 'remove', the rights in <rights> are
* removed from the existing rights. Restriction for the model: only
* one application element of the type AoUserGroup is allowed.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param aid The Id of the application element.
*
* @param usergroupId The user group to set the rights for.
*
* @param rights The new rights for the user group. The rights
* constants are defined in the interface
* SecurityRights. The interface definition language

```

```

* IDL does not allow to set the values of enumerations
* thus the constant definitions had to be done in an
* interface.
*
* @param set What to do with the new right.
*/
void setElementRights (
in T_LONGLONG aid,
in T_LONGLONG usergroupId,
in T_LONG rights,
in RightsSet set)
raises (AoException) ;
/** (24010)
* Set the ACL information on some instance defined by the application
* element id <aid> and a sequence of instance defined by the id <iid>.
* The <usergroup_id> defines the user group the rights should be set
* for. <rights> defines the rights to set or to clear. If the
* parameter <set> is set to 'set', the rights in <rights> are set all
* others are cleared. If the parameter <set> is set to 'add', the
* rights in <rights> are added to the existing rights. If the
* parameter <set> is set to 'remove', the rights in <rights> are
* removed from the existing rights. Restriction for the model: only
* one application element of the type AoUserGroup is allowed.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param aid The Id of the application element.
*
* @param instIds The sequence of instance Id's.
*

```

```

* @param usergroupId The user group to set the rights for.
*
* @param rights The new right for the user group.
*
* @param set What to do with the new right.
*/
void setInstanceRights (
in T_LONGLONG aid,
in S_LONGLONG instIds,
in T_LONGLONG usergroupId,
in T_LONG rights,
in RightsSet set)
raises (AoException) ;
/** (24011)
* Retrieve access control list information for the given application
* attribute <aid>/<attr_name>.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param aid The Id of the application element.
*
* @param attrName The name of the attribute.
*
* @return The access control list entries of the give application
* attribute.
*/
ACLSequence getAttributeRights (
in T_LONGLONG aid,
in T_STRING attrName)
raises (AoException) ;

```

```

/** (24012)
 * Retrieve access control list information for the requested
 * application element <aid>.
 *
 * @throws AoException
 * with the following possible error codes:
 * AO_BAD_PARAMETER
 * AO_CONNECTION_LOST
 * AO_IMPLEMENTATION_PROBLEM
 * AO_NOT_IMPLEMENTED
 * AO_NO_MEMORY
 * AO_SESSION_NOT_ACTIVE
 *
 * @param aid The Id of the application element.
 *
 * @return The access control list entries of the given application
 * element.
 */
ACLSequence getElementRights (
in T_LONGLONG aid)
raises (AoException) ;
/** (24013)
 * Retrieve access control list information for the requested instance
 * <aid>/<iid>.
 *
 * @throws AoException
 * with the following possible error codes:
 * AO_BAD_PARAMETER
 * AO_CONNECTION_LOST
 * AO_IMPLEMENTATION_PROBLEM
 * AO_NOT_IMPLEMENTED
 * AO_NO_MEMORY
 * AO_SESSION_NOT_ACTIVE
 *
 * @param aid The Id of the application element.
 *
 * @param iid The Id of the instance.

```

```

*
* @return The access control list entries of the given instance.
*/
ACLSequence getInstanceRights (
in T_LONGLONG aid,
in T_LONGLONG iid)
raises (AoException) ;
/** (24014)
* Set the access control list information for the initial rights on
* some application element defined by <aid>. The <usergroup_id>
* defines the user group the rights should be set for. <rights>
* defines the rights to set or to clear. If the parameter <set> is set
* to 'set', the rights in <rights> are set all others are cleared. If
* the parameter <set> is set to 'add', the rights in <rights> are
* added to the existing rights. If the parameter <set> is set to
* 'remove', the rights in <rights> are removed from the existing
* rights. Restriction for the model: only one application element of
* the type AoUserGroup is allowed.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param aid The Id of the application element.
*
* @param usergroupId The user group to set the initial rights for.
*
* @param rights The new initial rights for the user group. The
* rights constants are defined in the interface
* SecurityRights. The interface definition language
* IDL does not allow to set the values of enumerations
* thus the constant definitions had to be done in an

```

```

* interface.
*
* @param refAid The Id of referencing application element for which
* the initial rights will be used. If no refAid is set
* the initial rights will be used for each new
* instance element independent of the application
* element.
*
* @param set What to do with the new initial rights.
*/
void setElementInitialRights (
in T_LONGLONG aid,
in T_LONGLONG usergroupId,
in T_LONG rights,
in T_LONGLONG refAid,
in RightsSet set)
raises (AoException) ;
/** (24015)
* Set the access control list information for the initial rights on
* some instance defined by the application element id <aid> and a
* sequence of instance defined by the id <iid>. The <usergroup_id>
* defines the user group the rights should be set for. <rights>
* defines the rights to set or to clear. If the parameter <set> is set
* to 'set', the rights in <rights> are set all others are cleared. If
* the parameter <set> is set to 'add', the rights in <rights> are
* added to the existing rights. If the parameter <set> is set to
* 'remove', the rights in <rights> are removed from the existing
* rights. Restriction for the model: only one application element of
* the type AoUserGroup is allowed.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY

```



```

* AO_SESSION_NOT_ACTIVE
*
* @param aid The Id of the application element.
*
* @param instIds The sequence of instance Id's.
*
* @param usergroupId The user group to set the initial rights for.
*
* @param rights The new initial right for the user group.
*
* @param refAid The Id of referencing application element for which
* the initial rights will be used. If no refAid is set
* the initial rights will be used for each new
* instance element independent of the application
* element.
*
* @param set What to do with the new initial rights.
*/
void setInstanceInitialRights (
in T_LONGLONG aid,
in S_LONGLONG instIds,
in T_LONGLONG usergroupId,
in T_LONG rights,
in T_LONGLONG refAid,
in RightsSet set)
raises (AoException) ;
/** (24016)
* Set the flag <set> in svcattr, if this reference will be used (or
* not) to retrieve the Initial Rights. If more than one reference is
* set to true, the union (or-function) of all rights are used.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED

```

```

* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
* AO_TRANSACTION_NOT_ACTIVE
*
* @param aid The application element Id.
*
* @param refName The name of the reference.
*
* @param set What to do with the given reference.
*/
void setInitialRightReference (
in T_LONGLONG aid,
in T_STRING refName,
in RightsSet set)
raises (AoException) ;
/** (24017)
* Get all attribute names (references) which are used to retrieve the
* Initial Rights.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param aid The application element Id.
*
* @return The names of the references which will be used for the
* initial rights determination.
*/
NameSequence getInitialRightReference (
in T_LONGLONG aid)
raises (AoException) ;
/** (24018)

```

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

*

* @throws AoException

* with the following possible error codes:

* AO_BAD_PARAMETER

* AO_CONNECTION_LOST

* AO_IMPLEMENTATION_PROBLEM

* AO_NOT_IMPLEMENTED

* AO_NO_MEMORY

* AO_SESSION_NOT_ACTIVE

*

* @param aid The Id of the application element.

*

* @return The access control list entries of the given application
 * element for the initial rights.

*/

InitialRightSequence getElementInitialRights (
 in T_LONGLONG aid)

raises (AoException) ;

/** (24019)

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

*

* @throws AoException

* with the following possible error codes:

* AO_BAD_PARAMETER

* AO_CONNECTION_LOST

* AO_IMPLEMENTATION_PROBLEM

* AO_NOT_IMPLEMENTED

* AO_NO_MEMORY

* AO_SESSION_NOT_ACTIVE

*

* @param aid The Id of the application element.

*

* @param iid The Id of the instance.

*

```

* @return The access control list entries of the given instance for
* the initial rights.
*/
InitialRightSequence getInstanceInitialRights (
in T_LONGLONG aid,
in T_LONGLONG iid)
raises (AoException) ;
/** (24020)
* Perform the Query. This method can be used for a more powerful query
* compared to the method getInstances of this interface, with join
* definitions and aggregate functions.
*
* The number of different application elements which are requested are
* exactly defined by the definition of the query given in the field
* anuSeq of the QueryStructureExt. The number of attributes for each
* application element is also given in the definition. The number of
* matching instances (their attributes) is not defined by the query
* and can be a large amount. Therefore only one iterator for the
* attribute values are defined.
*
* 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 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.

*

* In case an attribute of blob type is among the requested attributes,
 * the content of the return structure will depend on the number of
 * instances that are returned:

- * - In case attributes of only one instance are returned (either
 * because 'how_many' was set to 1 or there is only one instance

```

* available at the server) , the return structure contains the blob
* itself.
* - In case attributes of more than one instance are returned, the
* return structure does not contain the blobs; instead it contains the
* headers of the respective blobs.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param aoq The query definition.
*
* @param how_many Maximum number of values in each result set. Valid
* arguments are:
* how_many = 0 : report all found values,
* how_many > 0 : report a maximum number of values.
*
* @return The result set with the requested attribute values.
*/
ResultSetExtSequence getInstancesExt (
in QueryStructureExt aoq,
in T_LONG how_many)
raises (AoException) ;
/** (24021)
* 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.
* The behavior of the ASAM ODS server depends on whether the
* referenced element is a submatrix or a measurement. For more
* information on the server behavior see description at the method
* getValueMatrixInMode () of the interfaces 'Measurement' and
* 'SubMatrix'.

```

```

*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
* AO_INVALID_BASETYPE
* AO_INVALID_VALUEMATRIX_STRUCTURE
*
* @param elem The element id. The aid has to be the application
* element Id of the AoMeasurement or AoSubmatrix.
*
* @param vmMode The requested mode of the value matrix.
*
* @return The value matrix.
*/
ValueMatrix getValueMatrixInMode (
in ElemId elem,
in ValueMatrixMode vmMode)
raises (AoException) ;
}; // Interface ApplElemAccess.
/**
* The ASAM ODS query evaluator interface allows to perform queries in
* synchronous mode and to create Query objects for asynchronous
* execution.
*/
interface QueryEvaluator {
/** (25000)
* Evaluate a query in synchronous mode.
*
* @throws AoException
* with the following possible error codes:
* AO_QUERY_TYPE_INVALID
* AO_QUERY_INVALID

```

```

* AO_QUERY_PROCESSING_ERROR
* AO_QUERY_TIMEOUT_EXCEEDED
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param queryStr The query string.
*
* @param params Sequence of parameter names and values. The
* following parameters should be passed:
*
* Name: "MaxDuration";
* Type: T_LONG
* Comment: Can be used to restrict the processing
* time. The time is given in milliseconds,
* Default value: 0 (no restriction)
*
* @return The result of the query as an instance element iterator.
*/
InstanceElementIterator getInstances (
in T_STRING queryStr,
in NameValueSequence params)
raises (AoException) ;
/** (25001)
* Evaluate a query in synchronous mode.
*
* @throws AoException
* with the following possible error codes:
* AO_QUERY_TYPE_INVALID
* AO_QUERY_INVALID
* AO_QUERY_PROCESSING_ERROR
* AO_QUERY_TIMEOUT_EXCEEDED
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST

```



```

* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param queryStr The query string.
*
* @param params Sequence of parameter names and values. The
* following parameters should be passed:
*
* Name: "MaxDuration";
* Type: T_LONG
* Comment: Can be used to restrict the processing
* time. The time is given in milliseconds,
* Default value: 0 (no restriction)
*
* @return The result of the query as a name value unit sequence
* iterator. Each name value unit tuple is one cell of the
* table. The name value unit sequence is one row of the
* table.
*/
NameValueUnitSequenceIterator getTableRows (
in T_STRING queryStr,
in NameValueSequence params)
raises (AoException) ;
/** (25002)
* Evaluate a query in synchronous mode.
*
* @throws AoException
* with the following possible error codes:
* AO_QUERY_TYPE_INVALID
* AO_QUERY_INVALID
* AO_QUERY_PROCESSING_ERROR
* AO_QUERY_TIMEOUT_EXCEEDED
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM

```

```

* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param queryStr The query string.
*
* @param params Sequence of parameter names and values. The
* following parameters should be passed:
*
* Name: "MaxDuration";
* Type: T_LONG
* Comment: Can be used to restrict the processing
* time. The time is given in milliseconds,
* Default value: 0 (no restriction)
*
* @return The result of the query as a name value sequence unit
* sequence. Each name value 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.
*/
NameValueSeqUnitSequence getTable (
in T_STRING queryStr,
in NameValueSequence params)
raises (AoException) ;
/** (25003)
* Create a query object to execute it in asynchronous mode.
*
* @throws AoException
* with the following possible error codes:
* AO_QUERY_TYPE_INVALID
* AO_QUERY_INVALID
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE

```

```

*
* @param queryStr The query string
*
* @param params Sequence of parameter names and values. The
* following parameters should be passed:
*
* Name: "QueryResultType";
* Type: ResultType.
* Comment: Specifies what kind of result is expected
* by the client.
* Default value: INSTELEM_ITERATOR_AS_RESULT
*
* Name: "MaxDuration";
* Type: T_LONG
* Comment: Can be used to restrict the processing
* time. The time is given in milliseconds,
* Default value: 0 (no restriction)
*
* @return The query object.
*/
Query createQuery (
in T_STRING queryStr,
in NameValueSequence params)
raises (AoException) ;
}; // Interface QueryEvaluator.
/**
* The ASAM ODS interface Query.
*/
interface Query {
/** (26000)
* Get the QueryEvaluator object which is responsible for this query.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM

```

```

* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @return The QueryEvaluator object which is responsible for this
* query.
*/
QueryEvaluator getQueryEvaluator ()
raises (AoException) ;
/** (26001)
* Do the query pre-processing (optimization, etc.) Call can be
* omitted by the client. In this case the functionality is executed on
* the call of executeQuery.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
* AO_QUERY_PROCESSING_ERROR
* AO_QUERY_INVALID_RESULTTYPE
*
* @param params Sequence of parameter names and values. The
* following parameters should be passed:
*
* Name: "QueryResultType";
* Type: ResultType.
* Comment: Specifies what kind of result is expected
* by the client, this parameter is required if the
* parameters isn't given at the method createQuery of
* the interface QueryEvaluator.
* Default value: INSTELEM_ITERATOR_AS_RESULT
*/
void prepareQuery (

```

```

in NameValueSequence params)
raises (AoException);
/** (26002)
 * Execute query in asynchronous mode.
 *
 * @throws AoException
 * with the following possible error codes:
 * AO_BAD_PARAMETER
 * AO_CONNECTION_LOST
 * AO_IMPLEMENTATION_PROBLEM
 * AO_NOT_IMPLEMENTED
 * AO_NO_MEMORY
 * AO_SESSION_NOT_ACTIVE
 * AO_QUERY_PROCESSING_ERROR
 * AO_QUERY_INVALID_RESULTTYPE
 *
 * @param params Sequence of parameter names and values. The
 * following parameter should be passed:
 *
 * Name: "QueryResultType";
 * Type: ResultType.
 * Comment: Specifies what kind of result is expected
 * by the client, this parameter is required if the
 * parameters isn't given at the method prepareQuery or
 * the method createQuery of the interface
 * QueryEvaluator.
 * Default value: INSTELEM_ITERATOR_AS_RESULT
 *
 * Name: "Synchronous";
 * Type: T_BOOLEAN
 * Comment: In case of "true" guarantees synchronous
 * execution.
 * Default value: "false"
 */
void executeQuery (
in NameValueSequence params)
raises (AoException);

```

```

/** (26003)
 * Return query status.
 *
 * Returns INCOMPLETE if the query is still executing.
 *
 * Returns COMPLETE if the query finished execution or if the query
 * execution stopped because of an error or because the timeout was
 * exceeded.
 *
 * @throws AoException
 * with the following possible error codes:
 * AO_BAD_PARAMETER
 * AO_CONNECTION_LOST
 * AO_IMPLEMENTATION_PROBLEM
 * AO_NOT_IMPLEMENTED
 * AO_NO_MEMORY
 * AO_SESSION_NOT_ACTIVE
 *
 * @return The status of the query.
 */
QueryStatus getStatus ()
raises (AoException) ;
/** (26004)
 * Get the query result. This method should only be called after the
 * query has been executed. It returns an iterator on the instances
 * that were found by the query.
 *
 * @throws AoException
 * with the following possible error codes:
 * AO_BAD_PARAMETER
 * AO_CONNECTION_LOST
 * AO_IMPLEMENTATION_PROBLEM
 * AO_NOT_IMPLEMENTED
 * AO_NO_MEMORY
 * AO_SESSION_NOT_ACTIVE
 * AO_QUERY_PROCESSING_ERROR
 * AO_QUERY_TIMEOUT_EXCEEDED

```

```

* AO_QUERY_INCOMPLETE
* AO_QUERY_INVALID_RESULTTYPE
*
* @return The result of the query as an instance element iterator.
*/
InstanceElementIterator getInstances ()
raises (AoException) ;
/** (26005)
* Get the query result. This method should only be called after the
* query has been executed. It returns an iterator on the name value
* unit sequence.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
* AO_QUERY_PROCESSING_ERROR
* AO_QUERY_TIMEOUT_EXCEEDED
* AO_QUERY_INCOMPLETE
* AO_QUERY_INVALID_RESULTTYPE
*
* @return The result of the query as a name value unit sequence
* iterator. Each name value unit tuple is one cell of the
* table. The name value unit sequence is one row of the
* table.
*/
NameValueUnitSequenceIterator getTableRows ()
raises (AoException) ;
/** (26006)
* Get the query result. This method should only be called after the
* query has been executed. It returns the result as an structure.
*
* @throws AoException

```

```

* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
* AO_QUERY_PROCESSING_ERROR
* AO_QUERY_TIMEOUT_EXCEEDED
* AO_QUERY_INCOMPLETE
* AO_QUERY_INVALID_RESULTTYPE
*
* @return The result of the query as a name value sequence unit
* sequence. Each name value 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.
*/
NameValueSeqUnitSequence getTable ()
raises (AoException) ;
}; // Interface Query.
/**
* The name value unit sequence iterator. The table iterator as query
* result.
*/
interface NameValueUnitSequenceIterator {
/** (27001)
* Destroy the iterator and free the associated memory.
*
* @throws AoException
* with the following possible error codes:
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*/
void destroy ()

```



```

raises (AoException) ;
/** (27002)
 * Get the total number of elements accessible by the iterator.
 *
 * @throws AoException
 * with the following possible error codes:
 * AO_CONNECTION_LOST
 * AO_IMPLEMENTATION_PROBLEM
 * AO_NOT_IMPLEMENTED
 * AO_NO_MEMORY
 * AO_SESSION_NOT_ACTIVE
 *
 * @return The number of elements accessible by the iterator.
 */
T_LONG getCount ()
raises (AoException) ;
/** (27003)
 * Get the next n elements from the sequence.
 *
 * @throws AoException
 * with the following possible error codes:
 * AO_BAD_PARAMETER
 * AO_CONNECTION_LOST
 * AO_IMPLEMENTATION_PROBLEM
 * AO_NOT_IMPLEMENTED
 * AO_NO_MEMORY
 * AO_SESSION_NOT_ACTIVE
 *
 * @param how_many The number of requested elements.
 *
 * @return The next n values of the name-value-unit sequence. For
 * each NameValueUnit the next n values are stored in the value
 * sequence.
 */
NameValueSeqUnitSequence nextN (
in T_LONG how_many)
raises (AoException) ;

```

```

/** (27004)
 * Get the next element from the iterator.
 *
 * @throws AoException
 * with the following possible error codes:
 * AO_CONNECTION_LOST
 * AO_IMPLEMENTATION_PROBLEM
 * AO_NOT_IMPLEMENTED
 * AO_NO_MEMORY
 * AO_SESSION_NOT_ACTIVE
 *
 * @return The next name-value-unit tuple sequence from the
 * name-value-unit tuple.
 */
NameValueSeqUnit nextOne ()
raises (AoException) ;
/** (27005)
 * Reset the pointer in the element iterator to the first element.
 *
 * @throws AoException
 * with the following possible error codes:
 * AO_CONNECTION_LOST
 * AO_IMPLEMENTATION_PROBLEM
 * AO_NOT_IMPLEMENTED
 * AO_NO_MEMORY
 * AO_SESSION_NOT_ACTIVE
 */
void reset ()
raises (AoException) ;
}; // Interface NameValueUnitSequenceIterator.
/**
 * The ASAM ODS enumeration interface.
 */
interface EnumerationDefinition {
/** (28000)
 * List the possible names of the enumeration. The sort order of the
 * list is the value of the item. The first item has the value zero.

```

```

*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @return List with all possible names of the enumeration items.
*/
NameSequence listItemNames ()
raises (AoException) ;
/** (28001)
* Get the value of an item.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
* AO_NOT_FOUND
*
* @param itemName The name of the item.
*
* @return The number of the item.
*/
T_LONG getItem (
in T_STRING itemName)
raises (AoException) ;
/** (28002)
* Get the name of an item.
*

```

```

* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
* AO_NOT_FOUND
*
* @param item The value of the item.
*
* @return The name of the item.
*/
T_STRING getItemName (
in T_LONG item)
raises (AoException) ;
/** (28003)
* Add a new item to the enumeration. This method modifies the
* application model and is only allowed for the super user.
*
* 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.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
* AO_TRANSACTION_NOT_ACTIVE
* AO_ACCESS_DENIED
* AO_DUPLICATE_NAME
* AO_DUPLICATE_VALUE

```

```

*
* @param itemName The name of the new item.
*/
void addItem (
in T_STRING itemName)
raises (AoException);
/** (28004)
* Rename the item of the enumeration. This method modifies the
* application model and is only allowed for the super user.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
* AO_TRANSACTION_NOT_ACTIVE
* AO_NOT_FOUND
*
* @param oldItemName The existing name of the item.
*
* @param newItemName the new name of the item.
*/
void renameItem (
in T_STRING oldItemName,
in T_STRING newItemName)
raises (AoException);
/** (28005)
* Get the name of the enumeration.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM

```

```

* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @return Name of the enumeration.
*/
T_STRING getName ()
raises (AoException) ;
/** (28006)
* Set the name of the enumeration. This method modifies the
* application model and is only allowed for the super user.
*
* 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.
*
* @throws AoException
* with the following possible error codes:
* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
* AO_TRANSACTION_NOT_ACTIVE
* AO_ACCESS_DENIED
*
* @param enumName Name of the enumeration.
*/
void setName (
in T_STRING enumName)
raises (AoException) ;
/** (28007)
* Get the index of the enumeration.
*
* @throws AoException
* with the following possible error codes:

```

```

* AO_BAD_PARAMETER
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @return The index of the enumeration.
*/
T_LONG getIndex ()
raises (AoException) ;
}; // Interface EnumerationDefinition.
/**
* For iteration through the result elements, there is also a new iterator
* necessary called ElemResultSetExtSeqIterator. It's functionality is
* still the same as all other iterators and needs therefore no further
* explanation.
* There is only one difference to the other iterations. The iteration is
* done on the lowest level in that case on the TS_UnionSeq.
*/
interface ElemResultSetExtSeqIterator {
/** (29000)
* Destroy the iterator and free the associated memory.
*
* @throws AoException
* with the following possible error codes:
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*/
void destroy ()
raises (AoException) ;
/** (29001)
* Get the total number of elements accessible by the iterator.
*

```

```

* @throws AoException
* with the following possible error codes:
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @return The number of elements accessible by the iterator.
*/
T_LONG getCount ()
raises (AoException) ;
/** (29002)
* Get the next n elements from the sequence.
*
* @throws AoException
* with the following possible error codes:
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @param how_many The number of requested elements.
*
* @return The next n attribute values from the element result set
* sequence.
*/
ElemResultSetExtSequence nextN (
in T_LONG how_many)
raises (AoException) ;
/** (29003)
* Get the next element from the sequence.
*
* @throws AoException
* with the following possible error codes:
* AO_CONNECTION_LOST

```



```

* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*
* @return The next attribute values from the element result set
* sequence.
*/
ElemResultSetExt nextOne ()
raises (AoException) ;
/** (29004)
* Reset the pointer in the element sequence to the first element.
*
* @throws AoException
* with the following possible error codes:
* AO_CONNECTION_LOST
* AO_IMPLEMENTATION_PROBLEM
* AO_NOT_IMPLEMENTED
* AO_NO_MEMORY
* AO_SESSION_NOT_ACTIVE
*/
void reset ()
raises (AoException) ;
}; // Interface ElemResultSetExtSeqIterator.
}; // Module ods.
}; // Module asam.
}; // Module org.
#endif
/**
* The ASAM ODS error codes.
*
* AO_ACCESS_DENIED
* The remote server denied the access. If this error occurred it was not
* even possible to present the authentication information. This means the
* authentication information might be correct but the server refused the
* access already at a lower level.
*

```

* AO_BAD_OPERATION

* The BAD_OPERATION error code is used when a method is invalid in a marshalling operation.

*

* AO_BAD_PARAMETER

* A parameter of the wrong type was passed to the method. The minorCode tells which parameter (1, 2, 3 or 4) is bad. If more than one parameter is bad, only the first one is reported. This error can occur only in non-typesave language bindings. Those language bindings also impose the problem that not all parameter errors are automatically detectable.

*

* AO_CONNECTION_LOST

* Due to a hardware or network software problem the connection to the server was lost.

*

* AO_CONNECT_FAILED

* The connect to a server failed. This error may occur if the server is down or currently unreachable.

*

* AO_CONNECT_REFUSED

* The connection was refused by the server. This error may occur if the presented authentication information is either incorrect or incomplete. This error shall not occur if the server does not accept any more sessions due to overload problems. See AO_SESSION_LIMIT_REACHED for this case.

*

* AO_DUPLICATE_BASE_ATTRIBUTE

* Any application element may have only one base attribute of a certain type. This means it may have only one attribute of base attribute type NAME, one ID, one VERSION and so on.

*

* AO_DUPLICATE_NAME

* The implicit or explicit specified name is already in use but it is required to be unique.

*

* AO_DUPLICATE_VALUE

* The attribute is marked unique in the application model. Thus duplicate
 * values are not allowed.
 *

* AO_HAS_BASE_ATTRIBUTE
 * Base attribute found. It is not allowed to modify the datatype, unique-
 or obligatory flag .
 *

* AO_HAS_BASE_RELATION
 * Base relation found. It is not allowed to modify the relationtype,
 * -range or -ship of an application relation derived from a base
 * relation.
 *

* AO_HAS_INSTANCES
 * The operation is not allowed for elements that have instances.
 *

* AO_HAS_REFERENCES
 * The requested operation is not permitted because the target element has
 * references.
 *

* AO_IMPLEMENTATION_PROBLEM
 * This error is reserved for the reporting of implementation specific
 * problems that are not properly handled by the standard error
 * definitions. An application should not crash if this error occurs but
 * there is no way to react to this error other than reporting and
 * ignoring
 * it. The intend of this error is not to leave implementation specific
 * errors unreported.
 *

* AO_INCOMPATIBLE_UNITS
 * The units are incompatible. No conversion rule is known.
 *

* AO_INVALID_ASAM_PATH
 * The specified Asam path is invalid.
 *

* AO_INVALID_ATTRIBUTE_TYPE
 * The requested attribute type is invalid.
 *

* AO_INVALID_BASETYPE

* The specified base type is invalid. The following basetypes are
* allowed:

* AoAny

* AoAttributeMap

* AoEnvironment

* AoLocalColumn

* AoLog

* AoMeasurement

* AoMeasurementQuantity

* AoNameMap

* AoParameter

* AoParameterSet

* AoPhysicalDimension

* AoQuantity

* AoQuantityGroup

* AoSubmatrix

* AoSubTest

* AoTest

* AoTestDevice

* AoTestEquipment

* AoTestEquipmentPart

* AoTestSequence

* AoTestSequencePart

* AoUnit

* AoUnitGroup

* AoUnitUnderTest

* AoUnitUnderTestPart

* AoUser

* AoUserGroup

*

* AO_INVALID_BASE_ELEMENT

* The base element is invalid in this context. If this is an element of
* type measurement, another element of this type may already exist.

*

* AO_INVALID_BUILDUP_FUNCTION

* The specified build-up function is invalid.

*
* AO_INVALID_COLUMN
* The specified column is invalid.
*
* AO_INVALID_COUNT
* The specified number of points is invalid (probably negative) .
*
* AO_INVALID_DATATYPE
* The datatype is not allowed in the given context or it conflicts with
* an existing datatype definition.
*
* This error may also occur in non-typesave language bindings. To avoid
* this error in all language bindings it is recommended to use always the
* definitions of the enumeration "DataType".
*
* AO_INVALID_ELEMENT
* The element is invalid in this context.
*
* AO_INVALID_LENGTH
* The given length is invalid. Negative length values are not allowed.
*
* AO_INVALID_ORDINALNUMBER
* The ordinal number is either already used or less than zero.
*
* AO_INVALID_RELATION
* The relation is invalid. The related elements and the base relation do
* not fit.
*
* AO_INVALID_RELATIONSHIP
* This error may occur only in non-typesave language bindings. To avoid
* this error in all language bindings it is recommended to use always the
* definitions of the enumeration "Relationship".
*
* AO_INVALID_RELATION_RANGE
* The specified relation range is invalid.
*
* AO_INVALID_RELATION_TYPE

* This error may occur only in non-typesave language bindings. To avoid
 * this error in all language bindings it is recommended to use always the
 * definitions of the enumeration "RelationType".

*

* AO_INVALID_SET_TYPE

* The specified set-type is invalid.

*

* AO_INVALID_SMATLINK

* The submatrix link is invalid. Either submatrix 1 or 2 is not specified
 * or the ordinal number is missing when there is more than one SMatLink.

*

* AO_INVALID_SUBMATRIX

* The specified submatrix is invalid.

*

* AO_INVALID_VALUEMATRIX_STRUCTURE

* The server 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.

* - ..

*

* AO_IS_BASE_ATTRIBUTE

* The application attribute is already of a base attribute type. It can
 * not be changed. If this is required, the application attribute has to
 * be removed from its application element and re-created. This error may
 * occur if an application attribute derived from a base attribute

*

* a. shall be overwritten by another base
 * attribute type.

* b. shall receive another datatype.

* c. shall receive another unique-flag.

* d. shall receive another obligatory-flag.

*

* AO_IS_BASE_RELATION

* Properties of base relations may not be changed.

*

* AO_IS_MEASUREMENT_MATRIX

* The matrix is a complex, generated matrix from a measurement not just a simple submatrix. It is only allowed to modify submatrices but not the composed measurement matrices.

*

* AO_MATH_ERROR

* A computation error occurred. This can be an overflow, an underflow or a division by zero.

*

* AO_MISSING_APPLICATION_ELEMENT

* A required application element is missing.

*

* AO_MISSING_ATTRIBUTE

* A required (obligatory) attribute is missing.

*

* AO_MISSING_RELATION

* A required relation is missing.

*

* AO_MISSING_VALUE

* An obligatory value is missing.

*

* AO_NOT_FOUND

* The requested element was not found. This error occurs only in remove or rename operations if the subject of the operation is not found. All get- and list-methods return an empty list if the requested item is not found.

*

* AO_NOT_IMPLEMENTED

* The requested method is not yet implemented. This error is not allowed to occur in a certified implementation. It is intended to allow partial operational tests. during the development process.

*

* AO_NOT_UNIQUE

* This error occurs if the instances of a property are required to be unique.

- *
 - * AO_NO_MEMORY
 - * No more volatile memory available.
- *
 - * AO_NO_PATH_TO_ELEMENT
 - * A free-floating element was detected. No navigation path leads to this element.
- *
 - * AO_NO_SCALING_COLUMN
 - * The column is no scaling column.
- *
 - * AO_OPEN_MODE_NOT_SUPPORTED
 - * The requested open mode is not supported. Valid open modes are "read" and "write". Anything else is rejected with this error and no session is created.
- *
 - * AO_QUERY_INCOMPLETE
 - * The execution of the query was not yet completed.
- *
 - * AO_QUERY_INVALID
 - * Some error in the query string or some inconsistency between the return type of the query string and the result type specified by parameter "QueryResultType".
- *
 - * AO_QUERY_INVALID_RESULTTYPE
 - * The requested result type of the query do no metch with the previous definition of the result type.
- *
 - * AO_QUERY_PROCESSING_ERROR
 - * Some error occured during the execution of the query.
- *
 - * AO_QUERY_TIMEOUT_EXCEEDED
 - * It was not possible to execute the query within the time limit set by parameter "MaxDuration".
- *
 - * AO_QUERY_TYPE_INVALID
 - * The server does not support the specified query language type.


```

*
* AO_SESSION_LIMIT_REACHED
* The server does not accept any new connections. This error may occur if
* the server reached the session limit for a distinct user or the total
* number of sessions allowed.
*
* AO_SESSION_NOT_ACTIVE
* The session is no longer active. This error occurs if an attempt is
* made to call a method of a closed session. This error shall not be
* confused with the error AO_CONNECTION_LOST.
*
* AO_TRANSACTION_ALREADY_ACTIVE
* There may be only one active transaction at one time. If this error
* occurs there is already an active transaction. That transaction remains
* active in case of this error.
*
* AO_TRANSACTION_NOT_ACTIVE
* Write operation have to be done always in the context of a transaction.
* This error occurs if no transaction is active during a write operation.
*
* AO_UNKNOWN_ERROR
* Use the zero as unknown error to avoid confusing error messages if no
* error code has been set.
*
* AO_UNKNOWN_UNIT
* The unit is unknown.
*/

```

13.6 使用 ASAM 协调一致的数据类型

ASAM ODS 在规范的早期阶段就定义了数据类型，它们的前缀是 `T_resp.S_`，它们被用在各种实现中，同时 ASAM 对所制定的所有标准进行了数据类型协调工作，5.5 节描述了 ASAM 协调一致的数据类型和传统的 ODS 数据类型之间的关系。由于基于 ASAM ODS 数据类型的实现有很多，因此 OO-API 中保留了它们。但也有基于协调一致数据类型开发的实现。本节提供的 IDL 文件描述了协调一致数据类型。应当注意 ASAM ODS 数据类型枚举是标准化 ASAM 数据类型的组成部分，ASAM 官方发布的枚举项名称（前缀为 `DT_resp.DS_`）可以无限制使用。

13.6.1 映射文件 A_TYPES.IDL

下列 `a_types.idl` 文件是为了映射协调一致的数据类型。这里仅是为了提供参考信息，实际的文件由 ASAM 协会维护并可向其获取。

```
// *****
// *** ASAM Data Type Specification ***
// *** File : a_types.idl ***
// *** Version : 2.0 ***
// *** Date : November 18, 2005 ***
// *****
/*
-----

S 18.11.2005:
removed enumeration Data_Type
renamed: BYTE_ARRAY_256 as A_BYTE_ARRAY_256
BYTE_ARRAY as A_BYTE_ARRAY
added: A_SEQUENCE_BOOLEAN .. A_SEQUENCE_ASCIISTRING
-----*/

//----- Start of the a_types.IDL file -----
#ifndef A_TYPES_IDL
#define A_TYPES_IDL
// base types
typedef boolean A_BOOLEAN; // boolean
typedef char A_INT8; // 8 bit integer
typedef octet A_UINT8; // 8 bit unsigned integer
typedef short A_INT16; // 16 bit integer
typedef unsigned short A_UINT16; // 16 bit unsigned integer
typedef long A_INT32; // 32 bit integer
typedef unsigned long A_UINT32; // 32 bit unsigned integer
typedef float A_FLOAT32; // 32 bit float
typedef double A_FLOAT64; // 64 bit float
typedef A_UINT8 A_BCD; // BCD number
typedef A_INT32 A_ENUM; // Enumeration
//-----//

structures describing numeric values
// 64 bit integer
struct A_INT64 {
A_INT32 n32High;
```

```

A_UINT32 un32Low;
};
// 64 bit unsigned integer
struct A_UINT64 {
A_UINT32 un32High;
A_UINT32 un32Low;
};
// complex value (real and imaginary part are float values)
struct A_COMPLEX32 {
A_FLOAT32 fReal;
A_FLOAT32 fImaginary;
};
// complex value (real and imaginary part are double values)
struct A_COMPLEX64 {
A_FLOAT64 dReal;
A_FLOAT64 dImaginary;
};
//-----//
character types
typedef A_UINT8 A_ASCII; // semantically, one char 1..255
// or the termination character Hex00
typedef A_UINT16 A_UNICODE_2; // semantically: one 16 bit character defined
in
// ISO-10646 UCS-2 0x0001 - 0xFFFF
// or the termination character Hex0000
typedef A_INT32 A_UNICODE_4; // semantically: one 31 bit character defined in
// ISO-10646 UCS-4 0x00000001 - 0x7FFFFFFF
// or the termination character Hex00000000
//-----//      string
types
// ASCII string
typedef sequence <A_ASCII> A_ASCIISTRING; // max restricted to 4294967295
typedef sequence <A_ASCII, 64> A_ASCIISTRING_64;
typedef sequence <A_ASCII, 128> A_ASCIISTRING_128;
typedef sequence <A_ASCII, 255> A_ASCIISTRING_255;
typedef sequence <A_ASCII, 1024> A_ASCIISTRING_1024;
typedef sequence <A_ASCII, 2048> A_ASCIISTRING_2048;

```

```

typedef sequence <A_ASCII, 5100> A_ASCIISTRING_5100;
typedef sequence <A_ASCII, 2> A_ASCIISTRING_2;
// UCS-2 string
typedef sequence <A_UNICODE_2> A_UNICODE2STRING; // max restricted to
4294967295
// zero terminated with
// character Hex0000
// UCS-4 string
typedef sequence <A_UNICODE_4> A_UNICODE4STRING; // max restricted to
4294967295
// zero terminated with
// character Hex00000000
// character field types
// ASCII character field
struct A_ASCIIFIELD {
A_UINT32 un32Chars; // value range un32Chars: [1, 2^32-1]
A_ASCIISTRING szValue; // un32Chars contains the number of ASCII
// characters in the character field
};
// UCS-2 character field
struct A_UNICODE2FIELD {
A_UINT32 un32Chars; // value range un32Chars: [1, 2^32-1]
A_UNICODE2STRING szValue; // un32Chars contains the number of UCS-2
// characters in the character field
};
// UCS-4 character field
struct A_UNICODE4FIELD {
A_UINT32 un32Chars; // value range un32Chars: [1, 2^32-1]
A_UNICODE4STRING szValue; // un32Chars contains the number of UCS-4
// characters in the character field
};
//-----// bit and
byte fields
typedef sequence <A_UINT8, 256> A_BYTEARRAY_256;
struct A_BITFIELD {
A_INT16 n16NumberOfBits; // number of valid bits in bitfield
// [1 .. 2048]

```

```

A_BYTEARRAY_256 aun8Field; // array of bytes,
// length of un8Field is calculated as
// (un32NumberOfBits + 7) / 8
// value range for number of bytes: [1..256]
};
// maximum restricted to 4294967295
typedef sequence<A_UINT8> A_BYTEARRAY;
struct ASAM_BYTEFIELD {
A_UINT32 un32Bytes; // value range un32Bytes:[1, 2^32-1]
A_BYTEARRAY aun8Field; // array of bytes
};
//-----//
sequences
typedef sequence<A_BOOLEAN> A_SEQUENCE_BOOLEAN;
typedef sequence<A_INT8> A_SEQUENCE_INT8;
typedef sequence<A_UINT8> A_SEQUENCE_UINT8;
typedef sequence<A_INT16> A_SEQUENCE_INT16;
typedef sequence<A_UINT16> A_SEQUENCE_UINT16;
typedef sequence<A_INT32> A_SEQUENCE_INT32;
typedef sequence<A_UINT32> A_SEQUENCE_UINT32;
typedef sequence<A_INT64> A_SEQUENCE_INT64;
typedef sequence<A_UINT64> A_SEQUENCE_UINT64;
typedef sequence<A_FLOAT32> A_SEQUENCE_FLOAT32;
typedef sequence<A_FLOAT64> A_SEQUENCE_FLOAT64;
typedef sequence<A_COMPLEX32> A_SEQUENCE_COMPLEX_32;
typedef sequence<A_COMPLEX64> A_SEQUENCE_COMPLEX_64;
typedef sequence<A_ENUM> A_SEQUENCE_ENUM;
typedef sequence<A_BCD> A_SEQUENCE_BCD;
typedef sequence<A_ASCIISTRING> A_SEQUENCE_ASCIISTRING;
#endif
//----- End of the a_types.IDL file -----

```

13.7 快速参考

本节提供了最重要的 OO-API 接口的快速参考及其相关方法。CORBA IDL 语法用来描述返回值和参数的独立于语言的表达。有关内容的详细描述和示例可参见 OO-API 的语言联编部分（见 13.2 节）。

注：本节的内容仅为了便于专业用户使用，普通用户从中也可全面了解 OO-API 的概貌（详见 13.2 节）。

13.7.1 接口 AOFACTORY

接口AOFACTORY函数如下:

```
— T_STRING getDescription ()
— T_STRING getInterfaceVersion ()
— T_STRING getName ()
— T_STRING getType ()
— AoSession newSession (T_STRING auth)
```

13.7.2 接口 AOSESSION

接口AOSESSION函数如下:

```
— 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 ()
— EnumerationAttributeStructureSequence getEnumerationAttributes ()
— EnumerationStructureSequence getEnumerationStructure ()
— LockMode 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_BOOLEANset)
— void setLockMode (LockMode lockMode)
```

```

— void setPassword ( T_STRING username, T_STRING oldPassword,
  T_STRING newPassword)
— void startTransaction ()

```

13.7.3 接口 APPLELEMACCESS

接口APPLELEMACCESS函数如下：

```

— void deleteInstances (T_LONGLONG aid, S_LONGLONG instIds)
— ACLSequence getAttributeRights (T_LONGLONG aid, T_STRING attrName)
— InitialRightSequence getElementInitialRights (T_LONGLONG aid)
— ACLSequence getElementRights (T_LONGLONG aid)
— NameSequence getInitialRightReference (T_LONGLONG aid)
— InitialRightSequence getInstanceInitialRights ( T_LONGLONG aid,
  T_LONGLONG instId)
— ACLSequence getInstanceRights (T_LONGLONG aid, T_LONGLONG instId)
— ElemResultSetSequence getInstances(QueryStructure aoq, T_LONG how_many)
— ResultSetExtSequence getInstancesExt ( QueryStructureExt aoq,
  T_LONG how_many)
— S_LONGLONG getRelInst (ElemId elem, Name relName)
— ValueMatrix getValueMatrix (ElemId elem)
— ValueMatrix getValueMatrixInMode (ElemId elem, ValueMatrixMode vmMode)
— 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)

```

13.7.4 接口 APPLICATIONATTRIBUTE

接口APPLICATIONATTRIBUTE函数如下:

```

— ApplicationElement getApplicationElement ()
— BaseAttribute getBaseAttribute ()
— DataType getDataType ()
— EnumerationDefinition getEnumerationDefinition ()
— T_LONG getLength ()
— Name getName ()
— ACLSequence getRights ()
— T_LONGLONG getUnit ()
— T_BOOLEAN hasUnit ()
— T_BOOLEAN hasValueFlag ()
— T_BOOLEAN isAutogenerated ()
— T_BOOLEAN isObligatory ()
— T_BOOLEAN isUnique ()
— void setBaseAttribute (BaseAttribute baseAttr)
— void setDataType (DataType dataType)
— void setEnumerationDefinition (EnumerationDefinition enumDef)
— void setIsAutogenerated (T_BOOLEAN isAutogenerated)
— void setIsObligatory (T_BOOLEAN isObligatory)
— void setIsUnique (T_BOOLEAN isUnique)
— void setLength (T_LONG length)
— void setName (Name name)
— void setRights (InstanceElement usergroup, T_LONG rights, RightsSet set)
— void setUnit (T_LONGLONG unitId)
— void withUnit (T_BOOLEAN withUnit)
— void withValueFlag (T_BOOLEAN withValueFlag)

```

13.7.5 接口 APPLICATIONELEMENT

接口APPLICATIONELEMENT函数如下:

```

— ApplicationAttribute createAttribute ()
— InstanceElement createInstance (Name ieName)
— InstanceElementSequence createInstances
  (
    NameValueSeqUnitSequenceattributes,
    ApplicationRelationInstanceElementSeqSequencerelatedInstances)
— 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
  (RelationshippaeRelationship)
— ApplicationRelationSequence getRelationsByType
  (RelationTypepaeRelationType)
— ApplicationRelationSequence getRelationsByBaseName (Name baseRelName)
— ACLSequence getRights ()
— SecurityLevel getSecurityLevel ()
— NameSequence listAllRelatedElements ()
— NameSequence listAttributes (Pattern aaPattern)
— NameIterator listInstances ( Pattern aaPattern, NameSequence
  listRelatedElementsByRelationship (RelationshippaaRelationship)
— 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)

```

13.7.6 接口 APPLICATIONRELATION

接口APPLICATIONRELATION函数如下:

```

— 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)

```

13.7.7 接口 APPLICATIONSTRUCTURE

接口APPLICATIONSTRUCTURE函数如下：

```

— 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)

13.7.8 接口 BASEATTRIBUTE

接口BASEATTRIBUTE函数如下:

- BaseElement getBaseElement ()
- DataType getDataType ()
- EnumerationDefinition getEnumerationDefinition ()
- Name getName ()
- T_BOOLEAN isObligatory ()
- T_BOOLEAN isUnique ()

13.7.9 接口 BASEELEMENT

接口BASEELEMENT函数如下:

- BaseRelationSequence getAllRelations ()
- BaseAttributeSequence getAttributes (Pattern baPattern)
- BaseElementSequence getRelatedElementsByRelationship (RelationshipbrRelationship)
- BaseRelationSequence getRelationsByType (RelationType brRelationType)
- BaseType getType ()
- T_BOOLEAN isTopLevel ()
- NameSequence listAttributes (Pattern baPattern)
- BaseTypeSequence listRelatedElementsByRelationship (RelationshipbrRelationship)

13.7.10 接口 BASERELATION

接口BASERELATION函数如下:

- BaseElement getElem1 ()
- BaseElement getElem2 ()
- Name getInverseRelationName ()
- RelationRange getInverseRelationRange ()
- Relationship getInverseRelationship ()
- Name getRelationName ()
- RelationRange getRelationRange ()

- Relationship getRelationship ()
- RelationType getRelationType ()

13.7.11 接口 BASESTRUCTURE

接口BASESTRUCTURE函数如下:

- 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)

13.7.12 接口 BLOB

接口BLOB函数如下:

- 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)

13.7.13 接口 COLUMN

接口COLUMN函数如下:

- void destroy ()
- DataType getDataType ()
- T_STRING getFormula ()
- TS_UNION getGenerationParameters ()
- Name getName ()
- DataType getRawDataType ()
- T_LONG getSequenceRepresentation ()
- InstanceElement getSourceMQ ()
- T_STRING getUnit ()
- T_BOOLEAN isIndependent ()
- T_BOOLEAN isScaling ()
- void setFormula (T_STRING formula)

```

— void setGenerationParameters (TS_UNION genPar)
— void setIndependent (T_BOOLEAN independent)
— void setScaling (T_BOOLEAN scaling)
— void setSequenceRepresentation (T_LONG seqRep)
— void setUnit (T_STRING unit)

```

13.7.14 接口 ELEMRESULTSETTEXTSEQITERATOR

接口ELEMRESULTSETTEXTSEQITERATOR函数如下:

```

— void destroy ()
— T_LONG getCount ()
— ElemResultSetExtSequence nextN (T_LONG how_many)
— ElemResultSetExt nextOne ()
— void reset ()

```

13.7.15 接口 ENUMERATIONDEFINITION

接口 ENUMERATIONDEFINITION 函数如下:

```

— 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)

```

13.7.16 接口 INSTANCEELEMENT

接口INSTANCEELEMENT 函数如下:

```

— void addInstanceAttribute (NameValueUnit instAttr)
— T_LONGLONG compare (InstanceElement compIeObj)
— InstanceElementSequence createRelatedInstances
  ( ApplicationRelationapplRel, NameValueSeqUnitSequence
    attributes,ApplicationRelationInstanceElementSeqSequence
    relatedInstances)
— void createRelation ( ApplicationRelation relation,
  InstanceElementinstElem)
— 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
  (RelationshipieRelationship, Pattern iePattern)
— ACLSequence getRights ()
— NameValueUnit getValue (Name attrName)
— NameValueUnit getValueByBaseName (Name baseAttrName)
— NameValueUnit getValueInUnit (NameUnit attr)
— NameValueUnitSequence getValueSeq (NameSequence attrNames)
— NameSequence listAttributes (Pattern iaPattern, AttrType aType)
— NameIterator listRelatedInstances ( ApplicationRelation
  ieRelation, Pattern iePattern)
— NameIterator listRelatedInstancesByRelationship ( in
  RelationshipieRelationship, Pattern, iePattern)
— void removeInstanceAttribute (Name attrName)
— void removeRelation ( ApplicationRelation applRel,
  InstanceElementinstElem_nm)
— void renameInstanceAttribute (Name oldName, Nme newName)
— void setInitialRights ( InstanceElement usergroup, T_LONG
  rights, T_LONGLONG refAid, RightsSet set)
— void setName (Name iaName)
— void setRights (InstanceElement usergroup, T_LONG rights, RightsSet set)
— void setValue (NameValueUnit value)
— void setValueSeq (NameValueUnitSequence values)
— InstanceElement shallowCopy (T_STRING newName, T_STRING newVersion)
— Measurement upcastMeasurement ()
— SubMatrix upcastSubMatrix ()

```

13.7.17 接口 INSTANCEELEMENTITERATOR

接口INSTANCEELEMENTITERATOR 函数如下:

```

— void destroy ()
— T_LONG getCount ()
— InstanceElementSequence nextN (T_LONG how_many)

```

- InstanceElement nextOne ()
- void reset ()

13.7.18 接口 MEASUREMENT

接口MEASUREMENT 函数如下:

该接口扩展了 InstanceElement.接口。

- SMatLink createSMatLink ()
- SMatLinkSequence getSMatLinks ()
- ValueMatrix getValueMatrix ()
- ValueMatrix getValueMatrixInMode (ValueMatrixMode vmMode)
- void removeSMatLink (SMatLink smLink)

13.7.19 接口 NAMEITERATOR

接口NAMEITERATOR 函数如下:

- void destroy ()
- T_LONG getCount ()
- NameSequence nextN (T_LONG how_many)
- Name nextOne ()
- void reset ()

13.7.20 接口 NAMEVALUEITERATOR

接口NAMEVALUEITERATOR 函数如下:

- void destroy ()
- T_LONG getCount ()
- NameValueSequence nextN (T_LONG how_many)
- NameValue nextOne ()
- void reset ()

13.7.21 接口 NAMEVALUEUNITIDITERATOR

接口NAMEVALUEUNITIDITERATOR 函数如下:

- void destroy ()
- T_LONG getCount ()
- NameValueUnitIdSequence nextN (T_LONG how_many)
- NameValueUnitId nextOne ()
- void reset ()

13.7.22 接口 NAMEVALUEUNITITERATOR

接口NAMEVALUEUNITITERATOR 函数如下:

- void destroy ()

- T_LONG getCount ()
- NameValueUnitSequence nextN (T_LONG how_many)
- NameValueUnit nextOne ()
- void reset ()

13.7.23 接口 NAMEVALUEUNITSEQUENCEITERATOR

接口NAMEVALUEUNITSEQUENCEITERATOR 函数如下:

- void destroy ()
- T_LONG getCount ()
- NameValueSeqUnitSequence nextN (T_LONG how_many)
- NameValueSeqUnit nextOne ()
- void reset ()

13.7.24 接口 QUERY

- void executeQuery (NameValueSequence params)
- InstanceElementIterator getInstances ()
- QueryEvaluator getQueryEvaluator ()
- QueryStatus getStatus ()
- NameValueSeqUnitSequence getTable ()
- NameValueUnitSequenceIterator getTableRows ()
- void prepareQuery (NameValueSequence params)

13.7.25 接口 QUERYEVALUATOR

接口QUERYEVALUATOR 函数如下:

- Query createQuery (T_STRING queryStr, NameValueSequence params)
- InstanceElementIterator getInstances (T_STRING queryStr,NameValueSequence params)
- NameValueSeqUnitSequence getTable (T_STRING queryStr, NameValueSequenceparams)
- NameValueUnitSequenceIterator getTableRows (T_STRING queryStr,NameValueSequence params)

13.7.26 接口 SMATLINK

接口SMATLINK 函数如下:

- BuildUpFunction getLinkType ()
- T_LONG getOrdinalNumber ()
- SubMatrix getSMat1 ()
- ColumnSequence getSMat1Columns ()
- SubMatrix getSMat2 ()

- ColumnSequence getSMat2Columns ()
- void setLinkType (BuildUpFunction linkType)
- void setOrdinalNumber (T_LONG ordinalNumber)
- void setSMat1 (SubMatrix subMat1)
- void setSMat1Columns (ColumnSequence columns)
- void setSMat2 (SubMatrix subMat2)
- void setSMat2Columns (in ColumnSequence columns)

13.7.27 接口 SUBMATRIX

该接口扩展了 InstanceElement.接口。函数如下：

- ColumnSequence getColumns (Pattern colPattern)
- ValueMatrix getValueMatrix ()
- ValueMatrix getValueMatrixInMode (ValueMatrixMode vmMode)
- NameSequence listColumns (Pattern colPattern)

13.7.28 接口 VALUEMATRIX

接口VALUEMATRIX 函数如下：

- 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)
- ValueMatrixMode getMode ()
- 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,
      NameValueSeqUnitSequencevalue)
— void      setValueMeaPoint      (      SetType      set,      T_LONG      meaPoint,
      NameValueSeqUnitSequencevalue)
— void      setValueVector      (      Column      col,      SetType      set,      T_LONG
      startPoint,TS_ValueSeq value)

```

14 TODS API (应用编程接口)

14.1 IMapleTRFactory (工厂)

IMapleTRFactory 作为所 TODS OO-API 的入口使用。所有的对象全部由此类完成，同时此类提供 TODS 全局性信息。

14.1.1 获取当前系统版本信息

```
ErrorCodeGetSystemVersion (out string version,out string ErrString)
```

功能：获取当前系统版本信息。服务器将从 TEnvironment 实例的基础属性“description”中调出描述信息。

输出参数： version 值的格式：主版本号.次版本号.内部版本号.修订版本号，例如：1.0.1.1

ErrString 错误信息。此参数输出此方法执行时的异常信息，若执行时未发生异常则此值为空字符串。

返回值：为 ErrorCode 的某个值见附录 A。

14.1.2 获取基础模型版本信息

```
ErrorCode GetBaseModelVersion (out string version,out string ErrString)
```

功能：获取基础模型版本信息

输出参数： version 值的格式:主版本号.次版本号.内部版本号.修订版本号，例如：1.0.1.1

ErrString 错误信息。此参数输出此方法执行时的异常信息，若执行时未发生异常则此值为空字符串。

返回值：为 ErrorCode 的某个值见附录 A。

14.1.3 获取应用模型版本信息

```
ErrorCode GetApplModelVersion (out string version,out string ErrString)
```

功 能：获取应用模型版本信息

输出参数： version 值的格式：主版本号.次版本号.内部版本号.修订版本号，例如：1.0.1.1

ErrMsg 错误信息。此参数输出此方法执行时的异常信息，若执行时未发生异常则此值为空字符串。

返回值：为 ErrorCode 的某个值见附录 A。

14.1.4 创建会话

ErrorCode NewSession (string username, string password, out IMapleTRSession, out string ErrMsg)

功能：创建会话实例

输入参数：username 表示用户登陆名。

password 表示用户登陆密码。

输出参数：ErrMsg 错误信息。此参数输出此方法执行时的异常信息，若执行时未发生异常则此值为空字符串。

返回值：为 ErrorCode 的某个值见附录 A。

14.2 IMapleTRSession （会话）

14.2.1 关闭会话

ErrorCode Close (out string ErrMsg)

功能：关闭会话，退出登陆

输出参数：ErrMsg 错误信息。此参数输出此方法执行时的异常信息，若执行时未发生异常则此值为空字符串。

返回值：为 ErrorCode 的某个值见附录 A。

14.2.2 获取 IMapleTRModelAccess

ErrorCode GetMapleTRModelAccess (out IMapleTRModelAccess, out string ErrMsg)

功能：获取数模型访问

输出参数：实例化有 IMapleTRModelAccess

ErrMsg 错误信息。此参数输出此方法执行时的异常信息，若执行时未发生异常则此值为空字符串。

返回值：为 ErrorCode 的某个值见附录 A。

14.2.3 获取 IMapleTRDBAccess

ErrorCode GetMapleTRModelAccess (out IMapleTRDBAccess, out string ErrMsg)

功能：获取 DB 数据访问

输出参数：实例化有 IMapleTRDBAccess

ErrString 错误信息。此参数输出此方法执行时的异常信息，若执行时未发生异常则此值为空字符串。

返回值：为 ErrorCode 的某个值见附录 A。

14.2.4 获取 IMapleTRFileAccess

ErrorCode GetMapleTRModelAccess (out IMapleTRFileAccess ,out string ErrString)

功能：获取文件访问

输出参数：IMapleTRFileAccess 实例化有 IMapleTRFileAccess

ErrString 错误信息。此参数输出此方法执行时的异常信息，若执行时未发生异常则此值为空字符串。

返回值：为 ErrorCode 的某个值见附录 A。

14.3 IMapleTRModelAccess（数据模型）

14.3.1 获取所有基础模型清单

ErrorCode GetBaseModelNameList (out string modelNameListPackage ,out string ErrString)

功 能：获取所有基础模型清单。

输出参数：ErrString 错误信息。此参数输出此方法执行时的异常信息，若执行时未发生异常则此值为空字符串。

返回值：为 ErrorCode 的某个值见附录 A。

14.3.2 获取所有基础模型信息

ErrorCode GetBaseModels (out string modelPackage ,out string ErrString)

功 能：获取所有基础模型表信息。

输出参数：ErrString 错误信息。此参数输出此方法执行时的异常信息，若执行时未发生异常则此值为空字符串。

14.3.3 获取制定模型信息

ErrorCode GetModelByName (string modelName,out string modelPackage,out string ErrString)

功 能：根据模型表名称获取基础模型信息。

输入参数: modelName

输出参数: ErrString 错误信息。此参数输出此方法执行时的异常信息,若执行时未发生异常则此值为空字符串。

返回值: 为 ErrorCode 的某个值见附录 A。

14.3.4 获取关联模型信息

ErrorCode GetModelRelationsByName (string modelName,out string modelPackage,out string ErrString)

功 能: 根据模型表名称获取关联模型信息。

输入参数: modelName

modelPackage

输出参数: ErrString 错误信息。此参数输出此方法执行时的异常信息,若执行时未发生异常则此值为空字符串。

返回值: 为 ErrorCode 的某个值见附录 A。

14.3.5 获取指定关联模型信息

ErrorCode GetModelRelationByMNRN (stringmodelName,string relationalName,out string modelPackage,out string ErrString)

功 能: 根据模型表名称和关系名称获取关联模型信息。

输入参数: modelName

relationalName

输出参数: ErrString 错误信息。此参数输出此方法执行时的异常信息,若执行时未发生异常则此值为空字符串。

返回值: 为 ErrorCode 的某个值见附录 A。

14.3.6 获取所有应用模型信息

ErrorCode GetAppModels (out string modelPackage,out string ErrString)

功 能: 获取所有应用模型表信息。

输入参数: modelPackage

输出参数: ErrString 错误信息。此参数输出此方法执行时的异常信息,若执行时未发生异常则此值为空字符串。

返回值：为 ErrorCode 的某个值见附录 A。

14.3.7 新建应用模型

ErrorCode AddAppModel (string modelPackage,out string ErrString)

功 能：新建应用模型表信息

输出参数：ErrString 错误信息。此参数输出此方法执行时的异常信息，若执行时未发生异常则此值为空字符串。

返回值：为 ErrorCode 的某个值见附录 A。

14.3.8 编辑应用模型信息

ErrorCode EditAppModel (string modelPackage,out string ErrString)

功 能：编辑模型表信息。

输入参数：modelPackage

输出参数：ErrString 错误信息。此参数输出此方法执行时的异常信息，若执行时未发生异常则此值为空字符串。

返回值：为 ErrorCode 的某个值见附录 A。

14.3.9 删除应用模型信息

ErrorCode DropAppModelByName (string modelName,out string ErrString)

功 能：根据模型表表名称删除模型表信息。

输入参数：modelName 元素名称

输出参数：ErrString 错误信息。此参数输出此方法执行时的异常信息，若执行时未发生异常则此值为空字符串。

返回值：为 ErrorCode 的某个值见附录 A。

14.4 IMapleTRDBAccess (DB 数据访问)

14.4.1 获取模型实例数据

ErrorCodeGetRecord (string modelName,string keyValue,out string dataPackage,out string ErrString) ;

功 能：获取模型实例数据。

输入参数：keyValue 表示 模型表 主键值

ErrMsg 错误信息。此参数输出此方法执行时的异常信息，若执行时未发生异常则此值为空字符串。

返回值：为 ErrorCode 的某个值见附录 A。

14.4.2 获取模型实例数据集合

```
ErrorCode GetRecordSet (string conditionXML , out int totalCount , out string dataPackage, out string ErrMsg);
```

功 能：获取模型实例数据集合。

输入参数：conditionXML 表示 Xml 格式条件，格式参照。

输出参数：totalCount 表示 总记录数

返回值：为 ErrorCode 的某个值见附录 A。

14.4.3 获取关联模型的实例数据

```
ErrorCode GetRelationnalRDSet ( string conditionXML , out int totalCount , out string dataPackage, out string ErrMsg);
```

功 能：获取指定模型的关联模型数据。

输入参数：conditionXML 表示 Xml 格式条件，格式参照。

输出参数：totalCount 表示 总记录数。

ErrMsg 错误信息。此参数输出此方法执行时的异常信息，若执行时未发生异常则此值为空字符串。

返回值：为 ErrorCode 的某个值，请参照附录 A。

14.4.4 新建模型实例数据

```
ErrorCode AddRecord (string dataPackage , out string keyValue , out string ErrMsg);
```

功 能：新建模型实例数据。

输出参数：keyValue 表示返回的模型实例数据主键值。

返回值：为 ErrorCode 的某个值见附录 A。

14.4.5 编辑模型数据

```
ErrorCode EditRecord (string dataPackage , out string ErrMsg);
```

功 能：编辑模型实例数据。

输入参数: dataPackage。

输出参数: ErrString 错误信息。此参数输出此方法执行时的异常信息,若执行时未发生异常则此值为空字符串。

返回值: 为 ErrorCode 的某个值见附录 A。

14.4.6 删除模型实例数据

```
ErrorCode DelRecord (string modelName, string [] arrKeyValue , out string ErrString);
```

功 能: 删除模型实例数据和其包含的实例数据及文件。

输入参数: modelName 元系名称。

arrKeyValue 表示 模型的主键值数组。

输出参数: ErrString 错误信息。此参数输出此方法执行时的异常信息,若执行时未发生异常则此值为空字符串。

返回值: 为ErrorCode的某个值。

附 录 A
(规范性附录)
执行状态 (ErrorCode)

执行状态 ErrorCode如下。

enum ErrorCode

{

 // 通用状态类

 // UNKNOWN = -1,

 Success = 0,

 Failure = 1,

 //2 开头的是数据库类

 DbError = 200,

 //3 开头的是模型类

 InvalidType = 300,

 InvalidPackage = 301,

 InvalidDataId = 302,

 FieldNotExist = 303,

 InvalidFieldType = 304,

 InvalidContainerId = 305,

 DataNotExist = 306,

 //4 开头的是文件类

 SourceFileKeyIsNull = 401,

 FileNotExists = 402,

 FileNameIsNull = 403,

 DiskNotExist = 404,

FileUploadFailure = 405,
FileStreamIsNull = 406,
FilePathIsEmpty = 407,
DiskNoFreeSpace = 408,
DelFileError = 409,
DownFileError = 410,
FileKeyIsEmptyOrNull = 411,
StartPositionGreaterThanFileSize = 412,
CopyFileToLocationOtherError = 413,
FileKeyIsUsed = 414,
FileNameIsInvalid = 415,
FolderNameIsInvalid = 416,
FileConvertError = 417
}
