



BSI Standards Publication

Industrial automation systems and integration — Product data representation and exchange

Part 27: Implementation methods: Java™ programming language binding
to the standard data access interface with Internet/Intranet extensions

National foreword

This Published Document is the UK implementation of ISO/TS 10303-27:2000.

The UK participation in its preparation was entrusted to Technical Committee AMT/4, Industrial data and manufacturing interfaces.

A list of organizations represented on this committee can be obtained on request to its committee manager.

Contractual and legal considerations

This publication has been prepared in good faith, however no representation, warranty, assurance or undertaking (express or implied) is or will be made, and no responsibility or liability is or will be accepted by BSI in relation to the adequacy, accuracy, completeness or reasonableness of this publication. All and any such responsibility and liability is expressly disclaimed to the full extent permitted by the law.

This publication is provided as is, and is to be used at the recipient's own risk.

The recipient is advised to consider seeking professional guidance with respect to its use of this publication.

This publication is not intended to constitute a contract. Users are responsible for its correct application.

This publication is not to be regarded as a British Standard.

© The British Standards Institution 2023
Published by BSI Standards Limited 2023

ISBN 978 0 539 22246 3

ICS 25.040.40

Compliance with a Published Document cannot confer immunity from legal obligations.

This Published Document was published under the authority of the Standards Policy and Strategy Committee on 31 March 2023.

Amendments/corrigenda issued since publication

Date	Text affected
------	---------------

TECHNICAL SPECIFICATION

ISO/TS 10303-27

First edition
2000-11-15

Industrial automation systems and integration — Product data representation and exchange —

Part 27:

Java TM programming language binding to the standard data access interface with Internet/ Intranet extensions

*Systèmes d'automatisation industrielle et intégration —
Représentation et échange de données de produits —*

*Partie 27: Méthodes de mise en application: Liaison de langage de
programmation Java TM à l'interface d'accès aux données normales
avec extensions Internet/Intranet*



Reference number
ISO/TS 10303-27:2000(E)

AVA Documents
Document Sharing Hub
© ISO 2000



COPYRIGHT PROTECTED DOCUMENT

© ISO 2000, Published in Switzerland

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized otherwise in any form or by any means, electronic or mechanical, including photocopying, or posting on the internet or an intranet, without prior written permission. Permission can be requested from either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office
Ch. de Blandonnet 8 • CP 401
CH-1214 Vernier, Geneva, Switzerland
Tel. +41 22 749 01 11
Fax +41 22 749 09 47
copyright@iso.org
www.iso.org

Contents

Page

Foreword	viii
Introduction	xiii
1 Scope	1
2 Normative references	1
3 Terms, definitions, and abbreviations	2
3.1 Terms defined in ISO 10303-1	2
3.2 Terms defined in ISO 10303-11	2
3.3 Terms defined in ISO 10303-22	3
3.4 Terms defined in The Java Language Specification	3
3.5 Other definitions	4
3.6 Abbreviations	5
4 General characteristics	5
4.1 Entity Attribute and Aggregate Member access	6
4.2 Dynamic SdaiRepositories	7
4.3 Packages and EXPRESS schema	8
4.4 Aggregates	8
4.5 Conformance requirements	8
4.6 Agglomerate domain	8
4.7 Date and time	9
5 Mapping of the SDAI_parameter_data_schema	9
5.1 String_value	9
5.2 Binary_value	9
5.3 Integer_value	9
5.4 Real_value	9
5.5 Number_value	9
5.6 Boolean_value	9
5.7 Logical_value	9
5.8 Query_source	10
5.9 Iterator	10
5.10 Entity_instance	10
5.11 Attribute_value	10
5.12 Select_value	10
5.13 Select_aggregate_instance	10
5.14 Enumeration_value	10
5.15 Aggregate_instance	10
5.16 Non_persistent_list_instance	11
6 SDAI Session classes	11
6.1 SdaiSession	11
6.1.1 Open session	11
6.1.2 Close session	12
6.1.3 Start transaction read-write access	12
6.1.4 Start transaction read-only access	12
6.1.5 Get complex entity definition	12
6.1.6 Link repository	12
6.1.7 Create repository	12
6.1.8 Import clear text encoding	13
6.2 Implementation	13
6.3 SdaiRepository	14
6.3.1 Create SDAI-model	14
6.3.2 Create schema instance	14
6.3.3 Open repository	15
6.3.4 Close repository	15

6.3.5	Get session identifier.....	15
6.3.6	SDAI query.....	15
6.3.7	Export clear text encoding.....	15
6.3.8	Delete repository.....	15
6.3.9	Unlink repository.....	16
6.4	ASdaiRepository.....	16
6.5	SdaiTransaction.....	16
6.5.1	Commit.....	16
6.5.2	Abort.....	16
6.5.3	End transaction access and commit.....	16
6.5.4	End transaction access and abort.....	17
6.6	SchemaInstance.....	17
6.6.1	getValidationDateLong.....	17
6.6.2	Delete schema instance.....	17
6.6.3	Rename schema instance.....	17
6.6.4	Add SDAI-model.....	17
6.6.5	Remove SDAI-model.....	18
6.6.6	SDAI query.....	18
6.6.7	Validate global rule.....	18
6.6.8	Validate uniqueness rule.....	18
6.6.9	Validate instance reference domain.....	18
6.6.10	Validate schema instance.....	18
6.6.11	Is validation current.....	18
6.7	ASchemaInstance.....	18
6.7.1	getAssociatedModels.....	18
6.8	SdaiModel.....	19
6.8.1	getMode.....	20
6.8.2	Delete SDAI-model.....	20
6.8.3	Rename SDAI-model.....	20
6.8.4	Start read-only access.....	20
6.8.5	Promote SDAI-model to read-write.....	20
6.8.6	Reduce SDAI-model to read-only.....	20
6.8.7	End read-only access.....	20
6.8.8	Start read-write access.....	21
6.8.9	End read-write access.....	21
6.8.10	Get entity definition.....	21
6.8.11	Create entity instance.....	21
6.8.12	Copy application instance.....	21
6.8.13	copyInstances.....	21
6.8.14	Undo changes.....	21
6.8.15	Save changes.....	21
6.8.16	SDAI query.....	21
6.8.17	Get instances.....	21
6.8.18	Get exact instances.....	22
6.8.19	substituteInstance.....	22
6.8.20	Get defined schema.....	23
6.9	ASdaiModel.....	23
6.10	EntityExtent.....	23
6.11	AEntityExtent.....	23
6.12	LOGICAL.....	24
7	Late binding.....	24
7.1	Late binding base types.....	24
7.2	Select data types for late binding.....	24
7.3	EEntity.....	25
7.3.1	Get attribute.....	26
7.3.2	Get inverse attribute.....	26
7.3.3	Test attribute.....	26
7.3.4	Find entity instance SDAI-model.....	26

7.3.5	Get instance type	27
7.3.6	Is instance of	27
7.3.7	Is kind of	27
7.3.8	Find entity instance users	27
7.3.9	Find entity instance used in	27
7.3.10	Get attribute value bound	27
7.3.11	Find instance roles	27
7.3.12	Find instance data types	27
7.3.13	Delete application instance	27
7.3.14	Put attribute	28
7.3.15	Unset attribute value	28
7.3.16	Create aggregate instance	28
7.3.17	Get persistent label	28
7.3.18	Get description	28
7.3.19	Validate where rule	28
7.3.20	Validate required explicit attributes assigned	28
7.3.21	Validate inverse attributes	28
7.3.22	Validate explicit attributes references	28
7.3.23	Validate aggregates size	28
7.3.24	Validate aggregates uniqueness	28
7.3.25	Validate array not optional	29
7.3.26	Validate string width	29
7.3.27	Validate binary width	29
7.3.28	Validate real precision	29
7.3.29	compareValueBoolean	29
7.3.30	compareValueLogical	29
7.3.31	toString	29
7.4	Aggregate	30
7.4.1	Get aggregation type	31
7.4.2	Get member count	31
7.4.3	Is member	31
7.4.4	Create iterator	31
7.4.5	Attach iterator	31
7.4.6	Get lower bound	31
7.4.7	Get upper bound	32
7.4.8	SDAI query	32
7.4.9	Get by index	32
7.4.10	Get value bound by index	32
7.4.11	Put by index	32
7.4.12	Create aggregate instance by index	32
7.4.13	Test by index for late binding	32
7.4.14	Get lower index	32
7.4.15	Get upper index	32
7.4.16	Unset value by index	33
7.4.17	Reindex array	33
7.4.18	Reset array index	33
7.4.19	Add by index	33
7.4.20	Add aggregate instance by index	33
7.4.21	Remove by index	33
7.4.22	Add unordered	33
7.4.23	Create aggregate instance unordered	33
7.4.24	Remove unordered	33
7.4.25	Test current member	34
7.4.26	Get current member	34
7.4.27	Create aggregate instance as current member	34
7.4.28	Put current member	34
7.4.29	Add before current member	34
7.4.30	Add after current member	34

7.4.31	Create aggregate instance before current member	34
7.4.32	Create aggregate instance after current member	34
7.4.33	Clear	35
7.4.34	toString	35
7.5	SdaiIterator	35
7.5.1	Beginning	35
7.5.2	Next	35
7.5.3	Get value bound by iterator	35
7.5.4	Remove current member	35
7.5.5	End	36
7.5.6	Previous	36
7.5.7	Unset value current member	36
7.6	Aggregates of basic types	36
7.6.1	AEntity	36
7.6.2	A_string	36
7.6.3	Aa_string	36
7.6.4	A_double	37
7.6.5	Aa_double	37
7.6.6	A_integer	37
7.6.7	Aa_integer	38
7.6.8	A_enumeration	38
7.6.9	Aa_enumeration	38
7.6.10	A_boolean	38
7.6.11	Aa_boolean	39
7.6.12	A_binary	39
7.6.13	Aa_binary	39
8	Early binding	39
8.1	Naming conventions	40
8.2	Schema package	40
8.3	Schema class	40
8.4	Defined type interface	40
8.5	ENUMERATION class	41
8.6	Select data types	41
8.7	Java types for entity attributes and aggregate elements	43
8.8	Access methods for entity attributes	44
8.9	Entity Interface	44
8.9.1	Test attribute	44
8.9.2	Get attribute	44
8.9.3	Unset attribute value	45
8.9.4	Put attribute	45
8.9.5	Create aggregate instance	45
8.10	Entity class	45
8.10.1	Field definition	45
8.10.2	Get attribute definition	46
8.10.3	Find entity instance used in	46
8.11	Aggregate class	46
8.11.1	Is member	46
8.11.2	Test by index	47
8.11.3	Get by index	47
8.11.4	Put by index	47
8.11.5	Create aggregate instance by index	47
8.11.6	Add by index	47
8.11.7	Add aggregate instance by index	47
8.11.8	Test current member	48
8.11.9	Get current member	48
8.11.10	Put current member	48
8.11.11	Create aggregate instance as current member	48
8.11.12	Add before current member	48

8.11.13	Create aggregate instance before current member	49
8.11.14	Add after current member	49
8.11.15	Create aggregate instance after current member.....	49
8.11.16	Add unordered.....	49
8.11.17	Create aggregate instance unordered.....	49
8.11.18	Remove unordered.....	49
9	Error handling.....	50
9.1	SdaiException.....	50
10	SDAI dictionary	53
10.1	Entity definition	53
10.1.1	Is subtype of.....	54
10.1.2	Is domain equivalent with.....	54
Annex A	(normative) Information object registration	55
Annex B	(normative) Mapping between Part-21 and SDAI	56
Annex C	(informative) Events.....	57
Annex D	(informative) Examples.....	59
Annex E	(informative) Select path generator	72

Foreword

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

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 3.

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

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

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

An ISO/PAS or ISO/TS is reviewed every three years with a view to deciding whether it can be transformed into an International Standard.

Attention is drawn to the possibility that some of the elements of this part of ISO 10303 may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights.

ISO/TS 10303-27 was prepared by Technical Committee ISO/TC 184, *Industrial automation systems and integration*, Subcommittee SC 4, *Industrial data*.

ISO 10303 consists of the following parts under the general title *Industrial automation systems and integration — Product data representation and exchange*:

- Part 1, Overview and fundamental principles;
- Part 11, Description methods: The EXPRESS language reference manual;
- Part 12, Description methods: The EXPRESS-I language reference manual;
- Part 14, Description methods: The EXPRESS-X language reference manual;
- Part 21, Implementation methods: Clear text encoding of the exchange structure;
- Part 22, Implementation method: Standard data access interface specification;
- Part 23, Implementation method: C++ language binding to the standard data access interface;
- Part 24, Implementation method: C language binding to the standard data access interface;
- Part 26, Implementation method: Interface definition language binding to the standard data access interface;
- Part 27, Implementation method: Java™ programming language binding to the standard data access interface with Internet/Intranet extensions;

- Part 28, Implementation method: XML representation for EXPRESS-driven data;
- Part 29, Implementation method: Lightweight Java programming language binding to the standard data access interface with Internet/Intranet extensions;
- Part 31, Conformance testing methodology and framework: General concepts;
- Part 32, Conformance testing methodology and framework: Requirements on testing laboratories and clients;
- Part 34, Conformance testing methodology and framework: Abstract test methods;
- Part 35, Conformance testing methodology and framework: Abstract test methods for standard data access interface implementations;
- Part 41, Integrated generic resources: Fundamentals of product description and support;
- Part 42, Integrated generic resources: Geometric and topological representation;
- Part 43, Integrated generic resources: Representation structures;
- Part 44, Integrated generic resources: Product structure configuration;
- Part 45, Integrated generic resource: Materials;
- Part 46, Integrated generic resources: Visual presentation;
- Part 47, Integrated generic resource: Shape variation tolerances;
- Part 49, Integrated generic resource: Process structure and properties;
- Part 50, Integrated generic resource: Mathematical construct;
- Part 101, Integrated application resource: Draughting;
- Part 104, Integrated application resource: Finite element analysis;
- Part 105, Integrated application resource: Kinematics;
- Part 106, Integrated application resource: Building construction core model;
- Part 107, Integrated application resource: Engineering analysis core application reference model (EA C-ARM);
- Part 108, Integrated application resource: Parameterization and constraints for explicit geometric product models;
- Part 201, Application protocol: Explicit draughting;
- Part 202, Application protocol: Associative draughting;
- Part 203, Application protocol: Configuration controlled 3D designs of mechanical parts and assemblies;
- Part 204, Application protocol: Mechanical design using boundary representation;
- Part 205, Application protocol: Mechanical design using surface representation;
- Part 207, Application protocol: Sheet metal die planning and design;
- Part 208, Application protocol: Life cycle management - Change process;
- Part 209, Application protocol: Composite and metallic structural analysis and related design;
- Part 210, Application protocol: Electronic assembly, interconnect, and packaging design;

- Part 212, Application protocol: Electrotechnical design and installation;
- Part 213, Application protocol: Numerical control process plans for machined parts;
- Part 214, Application protocol: Core data for automotive mechanical design processes;
- Part 215, Application protocol: Ship arrangement;
- Part 216, Application protocol: Ship moulded forms;
- Part 217, Application protocol: Ship piping;
- Part 218, Application protocol: Ship structures;
- Part 221, Application protocol: Functional data and their schematic representation for process plant;
- Part 223, Application protocol: Exchange of design and manufacturing product information for casting parts;
- Part 224, Application protocol: Mechanical product definition for process plans using machining features;
- Part 225, Application protocol: Building elements using explicit shape representation;
- Part 226, Application protocol: Ship mechanical systems;
- Part 227, Application protocol: Plant spatial configuration;
- Part 230, Application protocol: Building structural frame: Steelwork;
- Part 231, Application protocol: Process engineering data: Process design and process specification of major equipment;
- Part 232, Application protocol: Technical data packaging core information and exchange;
- Part 233, Application Protocol: Systems engineering data representation;
- Part 234, Application protocol: Ship Operational logs, records, and messages;
- Part 235, Application Protocol: Materials information for the design and verification of products;
- Part 236, Application Protocol: Furniture product data and project data;
- Part 301, Abstract test suite: Explicit draughting;
- Part 302, Abstract test suite: Associative draughting;
- Part 303, Abstract test suite: Configuration controlled design;
- Part 304, Abstract test suite: Mechanical design using boundary representation;
- Part 305, Abstract test suite: Mechanical design using surface representation;
- Part 307, Abstract test suite: Sheet metal die planning and design;
- Part 308, Abstract test suite: Life cycle management --- Change process;
- Part 309, Abstract test suite: Composite and metallic structural analysis and related design;
- Part 310, Abstract test suite: Electronic assembly, interconnect, and packaging design;
- Part 312, Abstract test suite: Electrotechnical design and installation;
- Part 313, Abstract test suite: Numerical control process plans for machined parts;

- Part 314, Abstract test suite: Core data for automotive mechanical design processes;
- Part 315, Abstract test suite: Ship arrangement;
- Part 316, Abstract test suite: Ship moulded forms;
- Part 317, Abstract test suite: Ship piping;
- Part 318, Abstract test suite: Ship structures;
- Part 321, Abstract test suite: Functional data and schematic representation for process plant;
- Part 322, Abstract test suite: Exchange of product data for composite structures;
- Part 323, Abstract test suite: Exchange of design and manufacturing product information for cast parts;
- Part 324, Abstract test suite: Mechanical product definition for process plans using machining features;
- Part 325, Abstract test suite: Building elements using explicit shape representation;
- Part 326, Abstract test suite: Ship mechanical systems;
- Part 327, Abstract test suite: Plant spatial configuration;
- Part 329, Abstract test suite: Exchange of design and manufacturing product information for forged parts;
- Part 330, Abstract test suite: Building structural frame: Steelwork;
- Part 331, Abstract test suite: Process engineering data: Process design and process specification of major equipment;
- Part 332, Abstract test suite: Technical data packaging core information and exchange;
- Part 333, Abstract test suite: Systems engineering data representation;
- Part 334, Abstract test suite: Ship Operational logs, records, and messages;
- Part 335, Abstract test suite: Materials information for the design and verification of products;
- Part 336, Abstract test suite: Furniture product data and project data;
- Part 501, Application interpreted construct: Edge-based wireframe;
- Part 502, Application interpreted construct: Shell-based wireframe;
- Part 503, Application interpreted construct: Geometrically bounded 2D wireframe;
- Part 504, Application interpreted construct: Draughting annotation;
- Part 505, Application interpreted construct: Drawing structure and administration;
- Part 506, Application interpreted construct: Draughting elements;
- Part 507, Application interpreted construct: Geometrically bounded surface;
- Part 508, Application interpreted construct: Non-manifold surface;
- Part 509, Application interpreted construct: Manifold surface;
- Part 510, Application interpreted construct: Geometrically bounded wireframe;
- Part 511, Application interpreted construct: Topologically bounded surface;

- Part 512, Application interpreted construct: Faceted boundary representation;
- Part 513, Application interpreted construct: Elementary boundary representation;
- Part 514, Application interpreted construct: Advanced boundary representation;
- Part 515, Application interpreted construct: Constructive solid geometry;
- Part 517, Application interpreted construct: Mechanical design geometric presentation;
- Part 518, Application interpreted construct: Mechanical design shaded presentation;
- Part 519, Application interpreted construct: Geometric tolerances;
- Part 520, Application interpreted construct: Associative draughting elements;
- Part 1001, Application module: Appearance assignment;
- Part 1002, Application Module: Colour;
- Part 1003, Application module: Curve appearance;
- Part 1004, Application module: Elemental shape;
- Part 1005, Application module: Elemental topological shape;
- Part 1006, Application module: Foundation representation;
- Part 1007, Application module: General surface appearance;
- Part 1008, Application Module: Layer assignment;
- Part 1009, Application Module: Shape appearance and layers;

The structure of this International Standard is described in ISO 10303-1. The numbering of the parts of the International Standard reflects its structure:

- Parts 11 to 14 specify the description methods,
- Parts 21 to 29 specify the implementation methods,
- Parts 31 to 35 specify the conformance testing methodology and framework,
- Parts 41 to 50 specify the integrated generic resources,
- Parts 101 to 108 specify the integrated application resources,
- Parts 201 to 236 specify the application protocols,
- Parts 301 to 336 specify the abstract test suites,
- Parts 501 to 520 specify the application interpreted constructs,
- Parts 1001 to 1009 specify the application module

Should further parts of ISO 10303 be published, they will follow the same numbering pattern.

[Annexes A](#) and [B](#) form a normative part of this part of ISO 10303. [Annexes C](#) to [E](#) are for information only.

Introduction

ISO 10303 is an International Standard for the computer-interpretable representation and exchange of product data. The objective is to provide a neutral mechanism capable of describing product data throughout the life cycle of a product, independent from any particular system. The nature of this description makes it suitable not only for neutral file exchange, but also as a basis for implementing and sharing product databases and archiving.

This International Standard is organised as a series of parts, each published separately. The parts of ISO 10303 fall into one of the following series: description methods, integrated resources, application integrated constructs, application protocols, abstract test suites, implementation methods, and conformance testing. The series is described in ISO 10303-1. This part of ISO 10303 is a member of the implementation methods series.

This part of ISO 10303 specifies the complete Java™ programming language binding to the standard data access interface (SDAI). SDAI is the access interface specification to data that has been defined using ISO 10303-11. SDAI is specified in ISO 10303-22.

Major subdivisions in this part of ISO 10303 are:

- Fundamental concepts in [clauses 4](#) and [5](#);
- SDAI session classes to manage the environment in [clause 6](#);
- Late binding: Generic Java interfaces and classes for arbitrary EXPRESS schemas in [clause 7](#);
- Early binding: Specific Java interfaces and classes for particular EXPRESS schemas in [clause 8](#);
- Error handling through the Java exception mechanism in [clause 9](#).

This page deliberately left blank

Industrial automation systems and integration — Product data representation and exchange —

Part 27:

Implementation methods: Java™ programming language binding to the standard data access interface with Internet/Intranet extensions

1 Scope

This part of ISO 10303 specifies a binding of the Java¹⁾ programming language to application data modelled in EXPRESS, ISO 10303-11 and to the standard data access interface, ISO 10303-22. It also specifies an import and export mechanism for data according to the clear text encoding of the exchange structure, ISO 10303-21. A further extension is that SDAI repositories can be created, deleted and linked while the SDAI session is open. Dynamically linking SDAI repositories through a network like Internet or Intranet allows accessing and changing of remote data.

In addition to the scope of ISO 10303-22 the scope of this part of ISO 10303 contains:

- creating, deletion and linking of the data repositories during an SDAI session;
- specific support for linking a remote data repository through a network like Internet or Intranet;
- convenience interfaces, classes, fields and methods suitable to this Java programming language binding;
- implementation mechanisms for the handling of errors as specified in ISO 10303-22;
- import from and export to the clear text encoding of the exchange structure as specified in ISO 10303-21.

2 Normative references

The following normative documents contain provisions which, through reference in this text, constitute provisions of this part of ISO 10303. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this part of ISO 10303 are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated below. For undated references, the latest edition of the normative document referred to applies. Members of ISO and IEC maintain registers of currently valid International Standards.

ISO/IEC 8824-1:1998, *Information technology — Abstract Syntax Notation One (ASN.1): Specification of basic notation*.

ISO 10303-1:1994, *Industrial automation systems and integration — Product data representation and exchange — Part 1: Overview and fundamental principles*.

ISO 10303-11:1994, *Industrial automation systems and integration — Product data representation and exchange — Part 11: Description methods: The EXPRESS language reference manual*.

1) Java is the trade mark of a product supplied by Sun Microsystems, Inc. This information is given for the convenience of users of this part of ISO 10303 and does not constitute an endorsement by ISO of the product name. Equivalent products may be used if they can be shown to lead to the same result.

ISO 10303-21:1994, *Industrial automation systems and integration — Product data representation and exchange — Part 21: Implementation methods: Clear text encoding of the exchange structure.*

ISO 10303-22:1998, *Industrial automation systems and integration — Product data representation and exchange — Part 22: Implementation methods: Standard data access interface.*

Technical Corrigendum 1 to ISO 10303-21:1994, *Industrial automation systems and integration — Product data representation and exchange — Part 21: Implementation methods: Clear text encoding of the exchange structure.*

The Java Language Specification, Second Edition by James Gosling, Bill Joy and Guy Steele, Addison Wesley, Reading Massachusetts, 2000, ISBN: 0201310082.
<http://java.sun.com/docs/books/jls/index.html>

The Java Virtual Machine Specification (2nd edition) by Tim Lindhold and Frank Yellin, Addison Wesley, Reading Massachusetts, 1999, ISBN: 0201432943.
<http://java.sun.com/docs/books/vmspec/index.html>

3 Terms, definitions, and abbreviations

For the purposes of this part of ISO 10303, the following definitions and naming conventions apply.

3.1 Terms defined in ISO 10303-1

For the purposes of this part of ISO 10303, the following terms defined in ISO 10303-1 apply:

- application;
- application protocol;
- data;
- implementation method;
- information;
- information model;
- product information model;
- structure.

3.2 Terms defined in ISO 10303-11

For the purposes of this part of ISO 10303, the following terms defined in ISO 10303-11 apply:

- aggregation data type;
- complex entity data type;
- partial complex entity data type;
- constructed data type;
- data type;
- defined data type;
- entity;
- entity data type;

- entity instance;
- enumeration data type;
- instance;
- named data type;
- population;
- select data type;
- value.

3.3 Terms defined in ISO 10303-22

For the purposes of this part of ISO 10303, the following terms defined in ISO 10303-22 apply:

- application schema;
- constraint;
- current schema;
- external schema;
- foreign schema;
- identifier;
- implementation class;
- iterator;
- native schema;
- repository;
- schema instance;
- SDAI-model;
- session;
- validation.

3.4 Terms defined in The Java Language Specification

For the purposes of this part of ISO 10303, the following terms defined in The Java Language Specification apply:

- class;
- constructor;
- exception;
- field;
- interface;
- member;
- method;

- overloading;
- package;
- parameter;
- primitive;
- reference;
- statement;
- type.

3.5 Other definitions

For the purpose of this part of ISO 10303, the following definitions apply.

3.5.1

aggregate class

a Java class representing an aggregation data type.

3.5.2

aggregate-dependent

The base type of an entity attribute or an aggregation type is **aggregate-dependent** if the aggregate value, returned by the EXPRESS **TypeOf** function for this base type contains the name of a defined type whose underlying type is an aggregation data type or if the base type is directly an aggregation type.

3.5.3

application entity

an entity declared in an application schema.

3.5.4

dictionary entity

an entity declared in the sdai_dictionary_schema.

3.5.5

entity class

a Java class representing a complex or single entity data type of an application schema or the sdai_dictionary_schema.

3.5.6

entity interface

a Java interface representing an entity data type of an application schema or the sdai_dictionary_schema.

3.5.7

persistent label

a string by which an entity instance can be uniquely identified within a SDAI repository.

3.5.8

receiver

a Java object upon which a specified member method is invoked.

3.5.9

SDAI method

a Java method representing an SDAI operation.

3.5.10

select-dependent

The base type of an entity attribute or an aggregation type is **selectdependent** if the aggregate value, returned by the EXPRESS **TypeOf** function for this base type contains the name of a defined type whose underlying type is a SELECT data type.

3.5.11

select path

a sequence of defined data types needed to specify a value of a select data type as defined in the technical corrigendum of ISO 10303-21:11.1.8.

3.5.12

session class

a specialized Java class for a session entity.

3.5.13

session entity

an entity declared in the SDAI_session_schema or the sdai_population_schema.

3.6 Abbreviations

For the purposes of this part of ISO 10303, the following abbreviations apply.

SDAI	Standard Data Access Interface, ISO 10303-22
part-21	Clear text encoding of the exchange structure, ISO 10303-21
URL	Universal Resource Locator

4 General characteristics

ISO 10303-22 specifies operations independently of any programming language. Language bindings of the operations are developed for programming languages to define the functionality required of conforming implementations. Two types of language bindings are identified: late bindings and early bindings. The concept of language bindings is defined in ISO 10303-22 clause 4.8. This part of ISO 10303 specifies an harmonised late and early language binding of Java to the SDAI. For this, Java packages, classes, interfaces, methods and data fields implementing SDAI and EXPRESS elements are defined. This part of ISO 10303 extends the functionality defined in ISO 10303-22 to provide more efficient or convenient operations.

Entity data types are mapped to the Java interface EEntity and other interfaces extending it for early binding access. These interfaces are called **entity interfaces**. Complex entity data types are mapped to the Java class CEntity and other classes extending it for early binding access. These classes are called **entity classes**. Each entity class implements one or several entity interfaces. Aggregation data types are mapped to the interface Aggregate and specialised classes implementing it. The class AEntity and other classes extending it represent aggregations of entity instances only. Further Java interfaces and classes are specified for other EXPRESS elements like schema and defined data types.

The SDAI operations are mapped to public Java methods. The input and output parameters of an SDAI operation are mapped to the parameters, receiver and return value of the corresponding method in Java. These methods are called **SDAI methods**. While executing SDAI methods errors may occur. These errors are reported to the application program by throwing an SdaiException. Every SDAI method throws SdaiException. The possible error indicators for an SDAI method are specified in the definition of the corresponding SDAI operation and, in addition, in this part of ISO 10303.

NOTE — An implementation may have further public or protected data fields and methods, however, to ensure robustness, such additional public or protected data fields and methods should be avoided. In the case where they cannot be avoided, their usage should be specified by the implementation. Except for classes representing aggregates all classes defined here should have no public constructors.

The entities defined in the SDAI_session_schema and the SDAI_population_schema are not mapped to entity classes and entity interfaces. Instead, specific session classes are defined for them, see [clause 6](#). The SDAI_dictionary_schema schema is mapped identically to application schemas with a few exceptions as defined in [clause 10](#). The mapping of the SDAI_parameter_data_schema to Java classes and interfaces is defined in [clause 5](#).

An implementation of this part of ISO 10303 is free to substitute any specified Java interface by an equivalent Java class as long as the Java class behaves exactly as the specified Java interface from a user perspective.

An application program should not extend or implement any of the Java interfaces or Java classes specified in this part of ISO 10303.

4.1 Entity Attribute and Aggregate Member access

Early and late binding SDAI methods are defined for the following entity and aggregate operations:

- Test attribute, specified in ISO 10303-22:10.10.2;
- Get attribute, specified in ISO 10303-22:10.10.1;
- Put attribute, specified in ISO 10303-22:10.11.3;
- Unset attribute value, specified in ISO 10303-22:10.11.4;
- Create aggregate instance, specified in ISO 10303-22:10.11.5;
- Test current member, specified in ISO 10303-22:10.17.2;
- Test by index, specified in ISO 10303-22:10.17.1;
- Is member, specified in ISO 10303-22:10.12.2;
- Get by index, specified in ISO 10303-22:10.15.1;
- Get current member, specified in ISO 10303-22:10.12.7;
- Put by index, specified in ISO 10303-22:10.16.1;
- Put current member, specified in ISO 10303-22:10.13.2;
- Add before current member, specified in ISO 10303-22:10.19.1;
- Add after current member, specified in ISO 10303-22:10.19.2;
- Add by index, specified in ISO 10303-22:10.19.3;
- Add unordered, specified in ISO 10303-22:10.14.1;
- Unset value by index, specified in ISO 10303-22:10.18.1;
- Unset value current, member specified in ISO 10303-22:10.18.2;
- Create aggregate instance by index, specified in ISO 10303-22:10.16.2;
- Create aggregate instance as current member, specified in ISO 10303-22:10.13.1;
- Create aggregate instance before current member, specified in ISO 10303-22:10.19.4;
- Create aggregate instance after current member, specified in ISO 10303-22:10.19.5;
- Add aggregate instance by index, specified in ISO 10303-22:10.19.6;
- Create aggregate instance unordered, specified in ISO 10303-22:10.14.2;
- Remove by index, specified in ISO 10303-22:10.19.7;
- Remove unordered, specified in ISO 10303-22:10.14.3;
- Remove current member, specified in ISO 10303-22:10.13.3.

An SdaiException VA_NSET shall be thrown when Get attribute, Get by index, or Get current member operation would result in an unset value. The Test attribute, Test by index, or Test current member operation shall be used to check for unset values.

An SdaiException VT_NVLD shall be thrown when Put Attribute, Put by index, Put current member, Add before current member, Add after current member, Add by index, and Add unordered is used with an unset value. The operations Unset attribute value, Unset value by index, or Unset value current member shall be used to unset values.

An SdaiException VT_NVLD shall be thrown when Put Attribute, Put by index, Put current member, Add before current member, Add after current member, Add by index, and Add unordered is applied with a value of type aggregate. The operations Create aggregate instance, Create aggregate instance by index, Create aggregate instance as current member, Create aggregate instance before current member, Create aggregate instance after current member, Create aggregate instance unordered, and Add aggregate instance by index shall be used to create new aggregates.

Attributes of session entities are read-only. They can only be accessed through the specialized early binding style methods defined in [clause 6](#).

For late and early binding it is important whether or not the base type of an entity attribute or an aggregation type is **select-dependent**.

4.2 Dynamic SdaiRepositories

While an SdaiSession is open all the available SdaiRepositories for this SdaiSession are accessible through the attribute known_servers. SDAI defines no operations to dynamically create, delete, or link other SdaiRepositories to the SdaiSession. This language binding defines additional SDAI methods to create, delete, link, and unlink SdaiRepositories while a SdaiSession is open. Furthermore, import and export functionality to part-21 exchange structures is defined.

After the open session operation, an initial, implementation dependent set known_servers is available. Further SdaiRepositories can be linked to the SdaiSession with the linkRepository method, see [6.1.6](#). For this, the additional attribute location is introduced for SdaiRepositories.

During linkRepository location is used to specify where to find the SdaiRepository. The interpretation of location is implementation dependent. It may be a path in the local file system or an Internet/Intranet address with login information such as user and password. Implementations may also support specific repository names from which the location of a repository can be derived.

A linked SdaiRepository may contain entity instances referencing other entity instances in other SdaiRepositories. These referenced SdaiRepositories shall also be included in the set of known_servers. However, the location of these SdaiRepositories may be unknown - such repositories are called **degenerated**. Degenerated SdaiRepositories can't be opened. The getLocation method applied to such an SdaiRepository returns a null value. Only the name of a degenerated SdaiRepository is known. The linkRepository method can be used to specify a location for a degenerated SdaiRepository and to find an SdaiRepository in the set of known_servers by name.

In a distributed environment it may happen that there are several SdaiRepositories with the same name. However, the name in the set of known_servers must be unique. To solve this problem, an application may remove the location information from an SdaiRepository with the unlinkRepository method, see [6.3.9](#), and then link it to a new location.

NOTE: The methods linkRepository and unlinkRepository do not copy the contents of the SdaiRepositories. They only influence the connection between the SdaiSession and the physical locations of SdaiRepositories.

New SdaiRepositories can be created with the create method, see [6.1.7](#). The deleteRepository operation physically deletes an SdaiRepository, see [6.3.8](#). A part-21 exchange structure can be imported into a new SdaiRepository with the importClearTextEncoding method, see [6.1.8](#).

The creation, deletion, linking and unlinking of SdaiRepositories, and the importClearTextEncoding are operations outside the SdaiTransaction mechanism. Applications may commit or

SdaiTransaction before invoking one of these operations. It is not possible to undo or abort these operations afterwards.

4.3 Packages and EXPRESS schema

Throughout this document the package prefix "SDAI." is used as a placeholder for an implementation specific package name. Implementations shall not directly use the package prefix "SDAI."

NOTE 1 — *The Java Language Specification* provides guidance how to choose unique package names.

All schema-independent Java interfaces and classes defined in this part of ISO 10303 shall be contained in a Java package named SDAI.lang. The SDAI_dictionary_schema schema is mapped into package SDAI.dictionary.

Further early binding packages are defined in [clause 8.2](#).

4.4 Aggregates

The EXPRESS aggregates LIST, ARRAY, SET, BAG and also the SDAI non_persistent_list are treated as Aggregates, see [7.4](#). However, instances of the aggregates need some knowledge of the purpose for which they are used. For this, the convenience method getAggregationType, see [7.4.1](#), is introduced. It is the responsibility of the implementation to allow only such operations on the aggregates, which fit to the aggregation type. Otherwise, an SdaiException has to be thrown.

4.5 Conformance requirements

An implementation of this part of ISO 10303 shall conform to an implementation class as specified in ISO 10303-22 clause 13. The level of session event recording support, specified in ISO 10303-22:13.1.3 and the level of scope support, specified in ISO 10303-22:13.1.4 shall not be considered when determining the implementation class, specified in ISO 10303-22:13.2.

For this language binding support of the SDAI_dictionary_schema for late binding access is mandatory.

The implementation shall support all SDAI methods whose original specification in ISO 10303-22 contains an operation required by the implementation class and shall support all convenience functions defined in this part of ISO 10303.

This part of ISO 10303 does not define details on how to implement the specified Java classes and methods.

4.6 Agglomerate domain

As defined in ISO 10303-22:8.4.1, a schema_instance is a logical collection of sdai_models, defining the **domain** over which references between entity instances in different SdaiModels are supported. The following SDAI operations use an aggregate of schema_instances as an **agglomerate domain** of schema_instances:

- Find entity instance users, specified in ISO 10303-22:10.10.8;
- Find entity instance usedin, specified in ISO 10303-22:10.10.9;
- Find instance roles, specified in ISO 10303-22:10.10.11.

Furthermore, such an agglomerate domain is needed for the operation Get Attribute, specified in ISO 10303-22:10.10.1, in the case of derived and inverse attributes. For the purpose of this language binding to the SDAI, an agglomerate domain shall be represented by an aggregate of sdai_models, ASdaiModel. This ASdaiModel aggregate is defined as a union of sets **associated_models** taken over all members of the aggregate of schema_instances. The ASchemaInstance method getAssociatedModels, see [clause 6.7.1](#), can be used to retrieve the agglomerate domain of SdaiModels.

4.7 Date and time

In Java, the actual date and time can be retrieved with the `java.lang.System` method `currentTimeMillis` as a Java long value with a precision of a millisecond. This long value has a higher accuracy than the date and time specification defined by the string-type `time_stamp`, see ISO 10303-22:7.3.3. Therefore, all validation and change times in this language binding can be alternatively accessed as values of type long.

5 Mapping of the SDAI_parameter_data_schema

This clause defines how EXPRESS types from the **SDAI_parameter_data_schema**, specified in ISO 10303-22 clause 9, shall be mapped to the Java programming language.

5.1 String_value

The Java class `java.lang.String` shall be associated with the SDAI data type **string_value**. For late binding attribute access the value shall be passed as an instance of type `java.lang.Object`.

5.2 Binary_value

The Java class `SDAI.lang.Binary` shall be associated with the SDAI data type **binary_value**. For late binding attribute access the value shall be passed as an instance of type `java.lang.Object`.

```
package SDAI.lang;
public final class Binary {
    public Binary(String str) throws SdaiException;
    public Binary(byte [] str, int size) throws SdaiException;
    public String toString();
    public int toByteArray(byte [] byte_array);
    public int getSize();
    boolean equals(Binary bin);
}
```

5.3 Integer_value

By default, the Java type `int` shall be associated with the SDAI data type **integer_value**. Late binding implementations may represent the actual value using an instance of `java.lang.Integer` or `java.math.BigInteger`, and the value shall be passed as an instance of type `java.lang.Object`.

5.4 Real_value

By default, the Java type `double` shall be associated with the SDAI data type **real_value**. Late binding implementations may represent the actual value using an instance of `java.lang.Double` or `java.math.BigDecimal`, and the value shall be passed as an instance of type `java.lang.Object`.

5.5 Number_value

By default, the Java type `double` shall be associated with the SDAI data type **number_value**. Late binding implementations may represent the actual value using an instance of `java.lang.Double` or `java.math.BigDecimal`, and the value shall be passed as an instance of type `java.lang.Object`.

5.6 Boolean_value

The Java type `boolean` shall be associated with the SDAI data type **boolean_value**.

5.7 Logical_value

The Java type `int` shall be associated with the SDAI data type **logical_value**.

5.8 Query_source

Query_source shall be associated with a method, query, in Aggregate, SdaiModel, SdaiRepository, and SchemaInstance.

5.9 Iterator

The Java class type SdaiIterator shall be associated with the SDAI data type **iterator**.

5.10 Entity_instance

A Java object of type EEntity shall represent either an **application_instance** or **dictionary_instance**. A Java object of type SdaiSession, SdaiRepository, SchemaInstance, SdaiModel, SdaiTransaction, or EntityExtent shall represent **sessioninstance**.

5.11 Attribute_value

This part of ISO 10303 does not define how to store **attribute_values**, however, it defines methods for late- and early-binding access to attribute values.

5.12 Select_value

This part of ISO 10303 does not define how to store **select_values**, however, it defines how select_values are handled for late and early-binding entity attribute access.

select_values shall be handled differently for late and early binding attribute access. For both bindings the same rules as for Select data types, defined in the technical corrigendum as specified in ISO 10303-21:11.1.8, shall be used.

For late binding attribute access the actual select type information for the attribute access methods shall be defined through a parameter of type Java array of EDefined_ type.

For early binding attribute access the actual select type information shall be defined through zero, one, or more parameters of the early binding Java interface types for the attribute access methods.

5.13 Select_aggregate_instance

This part of ISO 10303 does not define how to store select_aggregate_instances.

5.14 Enumeration_value

An enumeration_value shall be mapped to the Java type int. The numbering of the enumeration values shall correspond to the list indices of attribute elements of SDAI entity **enumeration_type**, specified in ISO 10303-22:6.4.28.

5.15 Aggregate_instance

An aggregate_instance shall be mapped to an instance of the Java interface Aggregate or Java interfaces which extend this interface.

The following subtypes of aggregate_instance shall be mapped to instances of Aggregate:

- set_instance;
- bag_instance;
- schemadefinedlistinstance;
- non_persistent_list_instance;

- array instance;
- application indexed array instance.

Except for `non_persistent_list_instances`, an application shall not use the Java new operator to create instances of any class implementing `Aggregate`.

For late binding attribute access the value shall be passed as an instance of type `java.lang.Object`.

5.16 Non_persistent_list_instance

A `non_persistent_list_instance` shall be either mapped to an instance of the Java class `AEntity` or to an instance of an early binding aggregate, see [clause 8.10](#). Non persistent lists are created with the Java new operator.

NOTE — There is no way to use an instance of a `non_persistent_list` as a value for an entity attribute. Non persistent lists are used only temporarily.

6 SDAI Session classes

This clause describes the session classes that represent the session entities, specified in ISO 1030322:7 and ISO 10303-22:8 and aggregate classes thereof.

6.1 SdaiSession

The `SdaiSession` class shall represent the SDAI entity **`sdai_session`**, specified in ISO 1030322:7.4.1.

In SDAI the inverse attribute **`active_transaction`** is defined as **SET [0:1] of `sdai_transaction`**, see ISO 10303-22:7.4.1. For the purpose of this language binding **`active_transaction`** is redefined as **OPTIONAL `sdai_transaction`**.

```
package SDAI.lang;
import SDAI.dictionary.*;
public class SdaiSession {
    public Implementation getSdaiImplementation() throws SdaiException;
    public ASdaiRepository getKnownServers() throws SdaiException;
    public ASdaiRepository getActiveServers() throws SdaiException;
    public boolean testDataDictionary() throws SdaiException;
    public SchemaInstance getDataDictionary() throws SdaiException;
    public ASdaiModel getActiveModels() throws SdaiException;
    public boolean testActiveTransaction() throws SdaiException;
    public SdaiTransaction getActiveTransaction() throws SdaiException;
    public static SdaiSession openSession() throws SdaiException;
    public void closeSession() throws SdaiException;
    public SdaiTransaction startTransactionReadWriteAccess() throws SdaiException;
    public SdaiTransaction startTransactionReadOnlyAccess() throws SdaiException;
    public EEntity_definition getComplexEntityDefinition(EEntity_definition types[],
        ESchema_definition schema) throws SdaiException;

    public Class getComplexEntityClass(Class[] types, Class schema) throws SdaiException;
    public SdaiRepository linkRepository(String repository_name, Object location)
        throws SdaiException;
    public SdaiRepository createRepository(String repository_name, Object location)
        throws SdaiException;
    public SdaiRepository importClearTextEncoding (String repository_name, Object
source_location,
        Object destination_location) throws SdaiException;
}
```

6.1.1 Open session

The static method `openSession` shall implement the SDAI operation **Open session**, specified in ISO 10303-22:10.3.1.

6.1.2 Close session

The `closeSession` method shall implement the SDAI operation **Close session**, specified in ISO 10303-22:10.4.4.

6.1.3 Start transaction read-write access

The `startTransactionReadWriteAccess` method shall implement the SDAI operation **Start transaction read-write access**, specified in ISO 10303-22:10.4.6.

6.1.4 Start transaction read-only access

The `startTransactionReadOnlyAccess` method shall implement the SDAI operation **Start transaction read-only access**, specified in ISO 10303-22:10.4.7.

6.1.5 Get complex entity definition

The `getComplexEntityDefinition` and `getComplexEntityClass` methods shall implement the SDAI operation **Get complex entity definition**, specified in ISO 10303-22:10.9.1.

The `types` array shall contain the leaf entity data types of the complex entity data type. Two overloaded methods are specified:

- the late binding method accepts and returns `EEntity_definitions`;
- the early binding method accepts Java classes for entity interfaces and returns the entity class representing complex entity datatype. This class shall implement all entity interlaces specified by the `types` parameter.

NOTE — The return value may be used as an input parameter for Create entity instance.

6.1.6 Link repository

The `linkRepository` method is a convenience method to make an already existing but external repository known to the current session. On success, the external repository shall be added to the current session's `known_servers` and returned. All repositories which are referenced by the application instances or `schema_instances`, associated with this repository, shall also be added to the set of `known_servers`. This method is an extension to the functionality of SDAI.

The parameter `location` shall be used to specify an address of the repository. The interpretation of this parameter is implementation-dependent. In the case when the value of this parameter is null, the method searches for the repository with a given name within `known_servers` and, if success, returns it.

Possible error indicators:

SS_NOPN	An SDAI session is not open.
TR_NEXS	A transaction has not been started.
RP_NEXS	The repository does not exist.
RP_DUP	Repository duplicate.
SY_ERR	An underlying system error occurred.

6.1.7 Create repository

The `createRepository` method is a convenience method which shall be used to create a repository.

The parameter `repository_name` shall specify the name of the SDAI repository. The parameter `location` shall be used to specify the address of the newly created repository. The value of this parameter is

implementation-dependent. The value may be null. The created repository is added to the list of known_servers.

Possible error indicators:

SS_NOPN	An SDAI session is not open.
RP_DUP	Repository duplicate.
SY_ERR	An underlying system error occurred.

6.1.8 Import clear text encoding

The importClearTextEncoding method shall be used to create a new repository, then to import a part-21 exchange structure into this repository, and open this repository. On success, the new repository is added to the set of known_servers and also to the set of **active_servers**.

The parameter name specifies the name of the SDAI repository. If this parameter is null the value of the attribute name in the header entity file_name, as defined in ISO 10303-21:9.2.2, shall be used as the name for the SDAI repository. If the name is invalid, then the SdaiException VT_NVLD is thrown.

The parameter source_location is used to specify where to find the exchange structure. The interpretation of this parameter is implementation-dependent.

The parameter destination_location is used to specify where to place the new created repository. The interpretation of this parameter is implementation-dependent. The value of this parameter may be null.

Possible error indicators:

SS_NOPN	An SDAI session is not open.
FN_NAVL	This function is not supported by this implementation.
TR_NEXS	A transaction has not been started.
TR_NRW	The transaction is not read-write.
TR_NAVL	The transaction is currently not available.
TR_EAB	The transaction ended abnormally.
VT_NVLD	The SDAI repository name is invalid.
RP_DUP	Repository duplicate.
SY_ERR	An underlying system error occurred.

6.2 Implementation

The Implementation class shall represent the SDAI entity **implementation**, specified in ISO 10303-22:7.4.2

```
package SDAI.lang;
import SDAI.dictionary.*;
public class Implementation {
    public String getName() throws SdaiException;
    public String getLevel() throws SdaiException;
    public String getSdaiVersion() throws SdaiException;
    public String getBindingVersion() throws SdaiException;
    public int getImplementationClass() throws SdaiException;
    public int getTransactionLevel() throws SdaiException;
    public int getExpressionLevel() throws SdaiException;
    public int getDomainEquivalenceLevel() throws SdaiException;
}
```

NOTE — The **level of event recording** and the **level of scope support** are not available for this language binding.