
**Information technology — Object
Management Group — Common Object
Request Broker Architecture (CORBA) —
Part 1:
Interfaces**

*Technologies de l'information — OMG (Object Management Group) —
CORBA (Common Object Request Broker Architecture) —*

Partie 1: Interfaces



COPYRIGHT PROTECTED DOCUMENT

© ISO/IEC 2012

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

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

Published in Switzerland

Table of Contents

Foreword	xvii
Introduction	xix
1 Scope	1
2 Conformance and Compliance	1
3 Normative References	1
4 Additional Information	2
4.1 Outline of Contents	2
4.2 Keywords for Requirement Statements	3
5 The Object Model	5
5.1 General	5
5.2 Overview	5
5.3 Object Semantics	6
5.3.1 Objects	6
5.3.2 Requests	6
5.3.3 Object Creation and Destruction	7
5.3.4 Types	7
5.3.5 Interfaces	8
5.3.6 Value Types	9
5.3.7 Abstract Interfaces	9
5.3.8 Operations	9
5.3.9 Attributes	11
5.4 Object Implementation	11
5.4.1 The Execution Model: Performing Services	11
5.4.2 The Construction Model	12
6 CORBA Overview	13
6.1 General	13

6.2 Structure of an Object Request Broker	13
6.2.1 Object Request Broker	17
6.2.2 Clients	18
6.2.3 Object Implementations	18
6.2.4 Object References	18
6.2.5 OMG Interface Definition Language	19
6.2.6 Mapping of IDL to Programming Languages	19
6.2.7 Client Stubs	19
6.2.8 Dynamic Invocation Interface	19
6.2.9 Implementation Skeleton	20
6.2.10 Dynamic Skeleton Interface	20
6.2.11 Object Adapters	20
6.2.12 ORB Interface	20
6.2.13 Interface Repository	21
6.2.14 Implementation Repository	21
6.3 Example ORBs	21
6.3.1 Client- and Implementation-resident ORB	21
6.3.2 Server-based ORB	21
6.3.3 System-based ORB	22
6.3.4 Library-based ORB	22
6.4 Structure of a Client	22
6.5 Structure of an Object Implementation	23
6.6 Structure of an Object Adapter	25
6.7 CORBA Required Object Adapter	26
6.7.1 Portable Object Adapter	26
6.8 The Integration of Foreign Object Systems	26
7 IDL Syntax and Semantics	29
7.1 Overview	29
7.2 Lexical Conventions	30
7.2.1 Tokens	33
7.2.2 Comments	33
7.2.3 Identifiers	33
7.2.4 Keywords	35
7.2.5 Literals	36
7.3 Preprocessing	38
7.4 IDL Grammar	38

7.5 IDL Specification	45
7.6 Import Declaration	45
7.7 Module Declaration	46
7.8 Interface Declaration	47
7.8.1 Interface Header	47
7.8.2 Interface Inheritance Specification	47
7.8.3 Interface Body	48
7.8.4 Forward Declaration	48
7.8.5 Interface Inheritance	49
7.8.6 Abstract Interface	51
7.8.7 Local Interface	51
7.9 Value Declaration	52
7.9.1 Regular Value Type	52
7.9.2 Boxed Value Type	54
7.9.3 Abstract Value Type	55
7.9.4 Value Forward Declaration	55
7.9.5 Valuetype Inheritance	55
7.10 Constant Declaration	57
7.10.1 Syntax	57
7.10.2 Semantics	58
7.11 Type Declaration	61
7.11.1 Basic Types	62
7.11.2 Constructed Types	64
7.11.3 Template Types	68
7.11.4 Complex Declarator	69
7.11.5 Native Types	69
7.11.6 Deprecated Anonymous Types	70
7.12 Exception Declaration	73
7.13 Operation Declaration	73
7.13.1 Operation Attribute	74
7.13.2 Parameter Declarations	74
7.13.3 Raises Expressions	74
7.13.4 Context Expressions	75
7.14 Attribute Declaration	76
7.15 Repository Identity Related Declarations	77
7.15.1 Repository Identity Declaration	77
7.15.2 Repository Identifier Prefix Declaration	78
7.15.3 Repository Id Conflict	79

7.16 Event Declaration	79
7.16.1 Regular Event Type	79
7.16.2 Abstract Event Type	80
7.16.3 Event Forward Declaration	80
7.16.4 Eventtype Inheritance	80
7.17 Component Declaration	80
7.17.1 Component	80
7.17.2 Component Header	81
7.17.3 Component Body	82
7.17.4 Event Sources—publishers and emitters	84
7.17.5 Event Sinks	84
7.17.6 Basic and Extended Components	85
7.18 Home Declaration	85
7.18.1 Home	85
7.18.2 Home Header	86
7.18.3 Home Body	87
7.19 CORBA Module	88
7.20 Names and Scoping	89
7.20.1 Qualified Names	89
7.20.2 Scoping Rules and Name Resolution	90
7.20.3 Special Scoping Rules for Type Names	93
8 ORB Interface	95
8.1 Overview	95
8.2 The ORB Operations	95
8.2.1 ORB Identity	101
8.2.2 Converting Object References to Strings	101
8.2.3 Getting Service Information	102
8.2.4 Creating a New Context	102
8.2.5 Thread-Related Operations	102
8.3 Object Reference Operations	105
8.3.1 Determining the Object Interface	107
8.3.2 Duplicating and Releasing Copies of Object References	107
8.3.3 Nil Object References	107
8.3.4 Equivalence Checking Operation	108
8.3.5 Probing for Object Non-Existence	108
8.3.6 Object Reference Identity	108
8.3.7 Type Coercion Considerations	110
8.3.8 Getting Policy Associated with the Object	110

8.3.9 Overriding Associated Policies on an Object Reference	111
8.3.10 Validating Connection	112
8.3.11 Getting the Domain Managers Associated with the Object	112
8.3.12 Getting Component Associated with the Object	113
8.3.13 Getting the ORB	113
8.3.14 LocalObject Operations	113
8.4 ValueBase Operations	114
8.5 ORB and OA Initialization and Initial References	115
8.5.1 ORB Initialization	115
8.5.2 Obtaining Initial Object References	117
8.5.3 Configuring Initial Service References	120
8.6 Context Object	122
8.6.1 Introduction	122
8.6.2 Context Object Operations	122
8.7 Current Object	125
8.8 Policy Object	126
8.8.1 Definition of Policy Object	126
8.8.2 Creation of Policy Objects	127
8.8.3 Usages of Policy Objects	129
8.8.4 Policy Associated with the Execution Environment	129
8.8.5 Specification of New Policy Objects	130
8.8.6 Standard Policies	131
8.9 Management of Policies	131
8.9.1 Client Side Policy Management	131
8.9.2 Server Side Policy Management	132
8.9.3 Policy Management Interfaces	132
8.10 Management of Policy Domains	134
8.10.1 Basic Concepts	134
8.10.2 Domain Management Operations	136
8.11 TypeCodes	138
8.11.1 The TypeCode Interface	138
8.11.2 TypeCode Constants	142
8.11.3 Creating TypeCodes	143
8.12 Exceptions	148
8.12.1 Definition of Terms	148
8.12.2 System Exceptions	148
8.12.3 Standard System Exception Definitions	150
8.12.4 Standard Minor Exception Codes	156

9	Value Type Semantics	157
9.1	Overview.....	157
9.2	Architecture.....	157
9.2.1	Abstract Values	158
9.2.2	Operations	158
9.2.3	Value Type vs. Interfaces	159
9.2.4	Parameter Passing	159
9.2.5	Substitutability Issues	160
9.2.6	Widening/Narrowing	161
9.2.7	Value Base Type	161
9.2.8	Life Cycle issues	161
9.2.9	Security Considerations	162
9.3	Standard Value Box Definitions	162
9.4	Language Mappings	163
9.4.1	General Requirements	163
9.4.2	Language Specific Marshaling	163
9.4.3	Language Specific Value Factory Requirements	163
9.4.4	Value Method Implementation	164
9.5	Custom Marshaling.....	164
9.5.1	Implementation of Custom Marshaling	164
9.5.2	Marshaling Streams	165
9.6	Access to the Sending Context Run Time	171
10	Abstract Interface Semantics	173
10.1	Overview.....	173
10.2	Semantics of Abstract Interfaces	173
10.3	Usage Guidelines	174
10.4	Example.....	174
10.5	Security Considerations.....	175
10.5.1	Passing Values to Trusted Domains	175
11	Dynamic Invocation Interface	177
11.1	Overview.....	177
11.1.1	Common Data Structures	177
11.1.2	Memory Usage	179
11.1.3	Return Status and Exceptions	179

11.2 Request Operations.....	179
11.2.1 create_request	180
11.2.2 add_arg	182
11.2.3 invoke	182
11.2.4 delete	183
11.2.5 send	183
11.2.6 poll_response	183
11.2.7 get_response	183
11.2.8 sendp	184
11.2.9 prepare	184
11.2.10 sendc	184
11.3 ORB Operations	185
11.3.1 send_multiple_requests	185
11.3.2 get_next_response and poll_next_response	185
11.4 Polling	186
11.4.1 Abstract Valuetype Pollable	187
11.4.2 Abstract Valuetype DIIPollable	188
11.4.3 interface PollableSet	188
11.5 List Operations	189
11.5.1 create_list	190
11.5.2 add_item	190
11.5.3 free	191
11.5.4 free_memory	191
11.5.5 get_count	191
11.5.6 create_operation_list	191
12 Dynamic Skeleton Interface	193
12.1 Introduction.....	193
12.2 Overview.....	193
12.3 ServerRequestPseudo-Object.....	194
12.3.1 ExplicitRequest State: ServerRequestPseudo-Object	194
12.4 DSI: Language Mapping.....	195
12.4.1 ServerRequest's Handling of Operation Parameters	195
12.4.2 Registering Dynamic Implementation Routines	195
13 Dynamic Management of Any Values	197
13.1 General.....	197
13.2 Overview.....	197

13.3 DynAny API	198
13.3.1 Creating a DynAny Object	204
13.3.2 The DynAny Interface	206
13.3.3 The DynFixed Interface	210
13.3.4 The DynEnum Interface	210
13.3.5 The DynStruct Interface	211
13.3.6 The DynUnion Interface	212
13.3.7 The DynSequence Interface	214
13.3.8 The DynArray Interface	215
13.3.9 The DynValueCommon Interface	216
13.3.10 The DynValue Interface	216
13.3.11 The DynValueBox Interface	217
13.4 Usage in C++ Language	218
13.4.1 Dynamic Creation of CORBA::Any values	218
13.4.2 Dynamic Interpretation of CORBA::Any values	219
14 The Interface Repository	221
14.1 Overview	221
14.2 Scope of an Interface Repository	221
14.3 Implementation Dependencies	223
14.3.1 Managing Interface Repositories	223
14.4 Basics	224
14.4.1 Names and Identifiers	224
14.4.2 Types and TypeCodes	225
14.4.3 Interface Repository Objects	225
14.4.4 Structure and Navigation of the Interface Repository	226
14.5 Interface Repository Interfaces	228
14.5.1 Supporting Type Definitions	229
14.5.2 IRObjct	230
14.5.3 Contained	231
14.5.4 Container	233
14.5.5 IDLType	238
14.5.6 Repository	238
14.5.7 ModuleDef	240
14.5.8 ConstantDef	240
14.5.9 TypedefDef	241
14.5.10 StructDef	241
14.5.11 UnionDef	242
14.5.12 EnumDef	243
14.5.13 AliasDef	243

14.5.14 PrimitiveDef	244
14.5.15 StringDef	244
14.5.16 WstringDef	244
14.5.17 FixedDef	245
14.5.18 SequenceDef	245
14.5.19 ArrayDef	245
14.5.20 ExceptionDef	246
14.5.21 AttributeDef	247
14.5.22 ExtAttributeDef	247
14.5.23 OperationDef	248
14.5.24 InterfaceDef	250
14.5.25 ExtInterfaceDef	252
14.5.26 AbstractInterfaceDef	253
14.5.27 ExtAbstractInterfaceDef	254
14.5.28 LocalInterfaceDef	255
14.5.29 ExtLocalInterfaceDef	256
14.5.30 ValueMemberDef	256
14.5.31 ValueDef	257
14.5.32 ExtValueDef	260
14.5.33 ValueBoxDef	262
14.5.34 NativeDef	262
14.6 Component Interface Repository Interfaces	262
14.6.1 ComponentIR::Container	262
14.6.2 ComponentIR::Repository	264
14.6.3 ComponentIR::ProvidesDef	265
14.6.4 ComponentIR::UsesDef	265
14.6.5 ComponentIR::EventDef	266
14.6.6 ComponentIR::EventPortDef	266
14.6.7 ComponentIR::EmitsDef	267
14.6.8 ComponentIR::PublishesDef	268
14.6.9 ComponentIR::ConsumesDef	268
14.6.10 ComponentIR::ComponentDef	268
14.6.11 ComponentIR::FactoryDef	271
14.6.12 ComponentIR::FinderDef	272
14.6.13 ComponentIR::HomeDef	272
14.7 RepositoryIds	274
14.7.1 IDL Format	275
14.7.2 RMI Hashed Format	275
14.7.3 DCE UUID Format	277
14.7.4 LOCAL Format	277
14.7.5 Pragma Directives for RepositoryId	277
14.7.6 For More Information	282

14.7.7 RepositoryIDs for OMG-Specified Types	282
14.7.8 Uniqueness Constraints on Repository IDs	283
14.8 IDL for Interface Repository	284
15 The Portable Object Adapter	303
15.1 Overview	303
15.2 Abstract Model Description	303
15.2.1 Model Components	303
15.2.2 Model Architecture	305
15.2.3 POA Creation	306
15.2.4 Reference Creation	307
15.2.5 Object Activation States	308
15.2.6 Request Processing	308
15.2.7 Implicit Activation	309
15.2.8 Multi-threading	310
15.2.9 Dynamic Skeleton Interface	311
15.2.10 Location Transparency	312
15.3 Interfaces	312
15.3.1 The Servant IDL Type	313
15.3.2 POAManager Interface	314
15.3.3 POAManagerFactory Interface	318
15.3.4 AdapterActivator Interface	319
15.3.5 ServantManager Interface	320
15.3.6 ServantActivator Interface	321
15.3.7 ServantLocator Interface	323
15.3.8 POA Policy Objects	325
15.3.9 POA Interface	328
15.3.10 Current Operations	337
15.4 IDL for PortableServer Module	338
15.5 UML Description of PortableServer	344
15.6 Usage Scenarios	346
15.6.1 Getting the Root POA	346
15.6.2 Creating a POA	347
15.6.3 Explicit Activation with POA-assigned Object Ids	347
15.6.4 Explicit Activation with User-assigned Object Ids	348
15.6.5 Creating References before Activation	349
15.6.6 Servant Manager Definition and Creation	349
15.6.7 Object Activation on Demand	351
15.6.8 Persistent Objects with POA-assigned Ids	352

15.6.9 Multiple Object Ids Mapping to a Single Servant	352
15.6.10 One Servant for All Objects	352
15.6.11 Single Servant, Many Objects and Types, Using DSI	355
16 Portable Interceptors	359
16.1 Introduction	359
16.1.1 Object Creation	359
16.1.2 Client Sends Request	360
16.1.3 Server Receives Request	361
16.1.4 Server Sends Reply	361
16.1.5 Client Receives Reply	362
16.2 General Behavior of Local Objects	362
16.3 Interceptor Interface	362
16.4 Request Interceptors	363
16.4.1 Design Principles	363
16.4.2 General Flow Rules	364
16.4.3 The Flow Stack Visual Model	364
16.4.4 The Request Interceptor Points	365
16.4.5 Client-Side Interceptor	365
16.4.6 Client-Side Interception Points	365
16.4.7 Client-Side Interception Point Flow	367
16.4.8 Server-Side Interceptor	370
16.4.9 Server-Side Interception Points	370
16.4.10 Server-Side Interception Point Flow	372
16.4.11 Request Information	375
16.4.12 RequestInfo Interface	375
16.4.13 ClientRequestInfo Interface	379
16.4.14 ServerRequestInfo Interface	382
16.4.15 ForwardRequest Exception	386
16.5 Portable Interceptor Current	386
16.5.1 Overview	386
16.5.2 Obtaining the Portable Interceptor Current	386
16.5.3 Portable Interceptor Current Interface	387
16.5.4 Use of Portable Interceptor Current	388
16.6 IOR Interceptor	392
16.6.1 Overview	392
16.6.2 An Abstract Model for Object Adapters	392
16.6.3 Object Reference Template	394
16.6.4 IORInterceptor Interface	396
16.6.5 IORInfo Interface	397

16.7	Interceptor Policy Objects	400
16.7.1	ProcessingMode Policy	400
16.8	PolicyFactory	401
16.8.1	PolicyFactory Interface	401
16.9	Registering Interceptors	401
16.9.1	ORBInitializer Interface	401
16.9.2	ORBInitInfo Interface	402
16.9.3	register_orb_initializer Operation	406
16.9.4	Notes about Registering Interceptors	408
16.10	Dynamic Initial References	409
16.10.1	register_initial_reference	409
16.11	Module Dynamic	410
16.11.1	NVList PIDL Represented by ParameterList IDL	410
16.11.2	ContextList PIDL Represented by ContextList IDL	410
16.11.3	ExceptionList PIDL Represented by ExceptionList IDL	410
16.11.4	Context PIDL Represented by RequestContext IDL	410
16.12	Consolidated IDL	410
16.12.1	Dynamic	410
16.12.2	Portions of IOP Relevant to Portable Interceptor	411
16.12.3	PortableInterceptor	412
17	CORBA Messaging	417
17.1	General	417
17.2	Quality of Service	417
17.3	Messaging Quality of Service	417
17.3.1	Rebind Support	419
17.3.2	Synchronization Scope	420
17.3.3	Request and Reply Priority	421
17.3.4	Request and Reply Timeout	422
17.3.5	Routing	424
17.3.6	Queue Ordering	425
17.4	Propagation of Messaging QoS	426
17.4.1	Structures	426
17.4.2	Messaging QoS Profile Component	426
17.4.3	Messaging QoS Service Context	426
17.5	Messaging Programming Model	427
17.6	Running Example	428

17.7 Async Operation Mapping	428
17.7.1 Callback Model Signatures (sendc)	429
17.7.2 Polling Model Signatures (sendp)	431
17.8 Exception Delivery in the Callback Model	433
17.8.1 Messaging::ExceptionHolder valuetype	433
17.9 Type-Specific ReplyHandler Mapping	433
17.9.1 ReplyHandler Operations for NO_EXCEPTION Replies	434
17.9.2 ReplyHandler Operations for Exceptional Replies	435
17.9.3 Example	435
17.10 Generic Poller Value.....	436
17.10.1 operation_target	437
17.10.2 operation_name	437
17.10.3 associated_handler	437
17.10.4 is_from_poller	437
17.11 Type-Specific Poller Mapping	437
17.11.1 Basic Type-Specific Poller	438
17.11.2 Persistent Type-Specific Poller	440
17.11.3 Example	440
17.12 Example Programmer Usage	441
17.12.1 Example Programmer Usage (Examples Mapped to C++)	441
17.12.2 Client-Side C++ Example for the Asynchronous Method Signatures ..	441
17.12.3 Client-Side C++ Example of the Callback Model	442
17.12.4 Client-Side C++ Example of the Polling Model	449
17.12.5 Server Side	454
17.13 Message Routing Interoperability	455
17.14 Routing Object References	456
17.15 Message Routing.....	457
17.15.1 Structures	459
17.15.2 Interfaces	460
17.15.3 Routing Protocol	462
17.16 Router Administration	467
17.16.1 Constants	470
17.16.2 Exceptions	470
17.16.3 Valuetypes	471
17.16.4 Interfaces	471
17.17 CORBA Messaging IDL	472
17.17.1 Messaging Module	472
17.17.2 MessageRouting Module	475

Chapter 17 Annexes

A.1 QoS Abstract Model Design	480
A.2 Model Components	480
A.2.1 Component Relationships	481
A.2.2 Component Design	481
A.3 AMI/TII Abstract Model Design.....	482
A.3.1 Asynchronous Method Invocation Components	482
A.3.2 Time-Independent Invocation Components	483
A.3.3 Component Relationships	483
A.3.4 Callback Model Detailed Design	486
A.3.5 Poller/PersistentRequest Detailed Design	487
A.4 Message Routing Abstract Model Design	488
A.4.1 Model Components	489
A.4.2 Component Relationships	489
A.4.3 Router Administration Design	489
B.1 Conformance Issues	491
B.2 Compatibility Issues	491
B.2.1 Transaction Service	491
B.2.2 Changes to Current OTS Behavior	491
B.2.3 Security Service	492
 Annex A - IDL Tags and Exceptions	 493
Annex B - Legal Information.....	507

Foreword

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

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

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

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

ISO/IEC 19500-1 was prepared by Technical Committee ISO/IEC JTC1, Information technology, in collaboration with the Object Management Group (OMG), following the submission and processing as a Publicly Available Specification (PAS) of the OMG Common Object Request Broker Architecture (CORBA) specification Part 1 Version 3.1 CORBA Interfaces.

ISO/IEC 19500-1 is related to:

- ITU-T Recommendation X.902 (1995) | ISO/IEC 10746-2:1996, Information Technology - Open Distributed Processing - Reference Model: Foundations
- ITU-T Recommendation X.903 (1995) | ISO/IEC 10746-3:1996, Information Technology - Open Distributed Processing - Reference Model: Architecture
- ITU-T Recommendation X.920 (1997) | ISO/IEC 14750:1997, Information Technology - Open Distributed Processing - Interface Definition Language
- ISO/IEC 19500-2, Information Technology - Open Distributed Processing - CORBA Specification Part 2: CORBA Interoperability
- ISO/IEC 19500-3, Information Technology - Open Distributed Processing - CORBA Specification Part 3: CORBA Components

ISO/IEC 19500 consists of the following parts, under the general title *Information technology - Open distributed processing - CORBA specification*:

- Part 1: CORBA Interfaces
- Part 2: CORBA Interoperability
- Part 3: CORBA Components

ISO/IEC 19500-1:2012(E)

It is the common core of the CORBA specification. Optional parts of CORBA, such as mappings to particular programming languages, Real-time CORBA extensions, and the minimum CORBA profile for embedded systems are documented in the other specifications that together comprise the complete CORBA specification. Please visit the CORBA download page at http://www.omg.org/technology/documents/corba_spec_catalog.htm to find the complete CORBA specification set.

Apart from this Foreword, the text of this International Standard is identical with that for the OMG specification for CORBA, v3.1.1, Part 1.

Introduction

The rapid growth of distributed processing has led to a need for a coordinating framework for this standardization and ITU-T Recommendations X.901-904 | ISO/IEC 10746, the Reference Model of Open Distributed Processing (RM-ODP) provides such a framework. It defines an architecture within which support of distribution, interoperability and portability can be integrated.

RM-ODP Part 2 (ISO/IEC 10746-2) defines the foundational concepts and modeling framework for describing distributed systems. The scopes and objectives of the RM-ODP Part 2 and the UML, while related, are not the same and, in a number of cases, the RM-ODP Part 2 and the UML specification use the same term for concepts which are related but not identical (e.g., interface). Nevertheless, a specification using the Part 2 modeling concepts can be expressed using UML with appropriate extensions (using stereotypes, tags, and constraints).

RM-ODP Part 3 (ISO/IEC 10746-3) specifies a generic architecture of open distributed systems, expressed using the foundational concepts and framework defined in Part 2. Given the relation between UML as a modeling language and Part 3 of the RM-ODP standard, it is easy to show that UML is suitable as a notation for the individual viewpoint specifications defined by the RM-ODP.

This International Standard for CORBA Interfaces is a standard for the technology specification of an ODP system. It defines a technology to provide the infrastructure required to support functional distribution of an ODP system, specifying functions required to manage physical distribution, communications, processing and storage, and the roles of different technology objects in supporting those functions.

Context of CORBA

The key to understanding the structure of the CORBA architecture is the Reference Model, which consists of the following components:

- **Object Request Broker**, which enables objects to transparently make and receive requests and responses in a distributed environment. It is the foundation for building applications from distributed objects and for interoperability between applications in hetero- and homogeneous environments. The architecture and specifications of the Object Request Broker are described in this manual.
- **Object Services**, a collection of services (interfaces and objects) that support basic functions for using and implementing objects. Services are necessary to construct any distributed application and are always independent of application domains. For example, the Life Cycle Service defines conventions for creating, deleting, copying, and moving objects; it does not dictate how the objects are implemented in an application. Specifications for Object Services are contained in *CORBA services: Common Object Services Specification*.
- **Common Facilities**, a collection of services that many applications may share, but which are not as fundamental as the Object Services. For instance, a system management or electronic mail facility could be classified as a common facility. Information about Common Facilities will be contained in *CORBA facilities: Common Facilities Architecture*.
- **Application Objects**, which are products of a single vendor or in-house development group that controls their interfaces. Application Objects correspond to the traditional notion of applications, so they are not standardized by OMG. Instead, Application Objects constitute the uppermost layer of the Reference Model.

The Object Request Broker, then, is the core of the Reference Model. It is like a telephone exchange, providing the basic mechanism for making and receiving calls. Combined with the Object Services, it ensures meaningful communication between CORBA-compliant applications.

The architecture and specifications described in this standard are aimed at software designers and developers who want to produce applications that comply with OMG specifications for the Object Request Broker (ORB), or this standard (ISO/IEC 19500). The benefit of compliance is, in general, to be able to produce interoperable applications that are based on distributed, interoperating objects. The ORB provides the mechanisms by which objects transparently make requests and receive responses. Hence, the ORB provides interoperability between applications on different machines in heterogeneous distributed environments and seamlessly interconnects multiple object systems.

This Part of this International Standard includes normative and non-normative annexes.

Information technology - Object Management Group Common Object Request Broker Architecture (CORBA), Interfaces

1 Scope

This part of ISO/IEC 19500 specifies the CORBA Object Model and uses concepts from that model to define the operation of the Object Request Broker (ORB). The ORB is the basic mechanism by which objects transparently make requests to, and receive responses from, each other on the same machine or across a network. A client need not be aware of the mechanisms used to communicate with or activate an object, how the object is implemented, or where the object is located.

2 Conformance and Compliance

The minimum required for a CORBA-compliant system is adherence to the specifications in this part of ISO/IEC 19500 and one mapping. Each additional language mapping is a separate, optional compliance point. Optional means users aren't required to implement these points if they are unnecessary at their site, but if implemented, they must adhere to the CORBA specifications to be called CORBA-compliant. For instance, if a vendor supports C++, their ORB must comply with the OMG IDL to C++ binding specified in the C++ Language Mapping Specification.

The CORBA Language Mappings have been separated from this standard and each language mapping is its own separate OMG specification.

3 Normative References

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

- ITU-T Recommendation X.902 (1995) | ISO/IEC 10746-2:1996, Information Technology - Open Distributed Processing - Reference Model: Foundations
- ITU-T Recommendation X.903 (1995) | ISO/IEC 10746-3:1996, Information Technology - Open Distributed Processing - Reference Model: Architecture
- ITU-T Recommendation X.920 (1997) | ISO/IEC 14750:1997, Information Technology - Open Distributed Processing - Interface Definition Language
- ISO/IEC 14882:2003, Information Technology - Programming languages - C++
- ISO/IEC 9899:1999, Information Technology - Programming languages - C
- [OMA] Object Management Group, "Object Management Architecture Guide, revision 3.0," available from <http://www.omg.org/oma/>

- [ASMOTS] Object Management Group, “Additional Structuring Mechanisms for the OTS,” available from <http://www.omg.org/spec/OTS/>
- [TRANS] Object Management Group, “Transaction Service,” available from <http://www.omg.org/spec/TRANS/>
- [FIREWALL] Object Management Group, “CORBA Firewall Traversal Specification,” available from <http://www.omg.org/members/cgi-bin/doc?ptc/04-04-05.pdf>
- [SCCP] Object Management Group, “CORBA / TC Interworking and SCCP-Inter ORB Protocol (SCCP),” available from <http://www.omg.org/spec/SCCP>
- [FTCORBA] Object Management Group, “Fault Tolerant Corba,” clause 23 of CORBA 3.0.3, available from <http://www.omg.org/cgi-bin/doc?formal/2004-03-01>
- [RTCORBA] Object Management Group, “Real-Time CORBA, version 1.2,” available from <http://www.omg.org/spec/RT/>
- [WATM] Object Management Group, “Wireless Access and Telecom Mobility in CORBA, Version 1.2,” available from <http://www.omg.org/spec/WATM/>
- [DCOMI] Object Management Group, “Interoperability with non-CORBA Systems” clause 20 of CORBA 3.0.3, available from <http://www.omg.org/cgi-bin/doc?formal/2004-03-01>
- [TSAS] Object Management Group “Telecommunications Service Access and Subscription Specification,” available from <http://www.omg.org/spec/TSAS/>
- [SECDOM] Object Management Group “Security Domain Membership Management Service,” available from <http://www.omg.org/members/cgi-bin/doc?orbos/01-06-01.pdf>
- [RFC2119] IETF RFC 2119, “Key words for use in RFCs to Indicate Requirement Levels,” S. Bradner, March 1997. Available from <http://ietf.org/rfc/rfc2119>