

Contents

Foreword	vii
Preface	ix
Acknowledgements	xiii
Chapter 1 Introduction	1
1.1 System architecture	4
1.2 The CORBA standard	6
1.2.1 Interface definition language (IDL)	7
1.2.2 Relationship to object oriented analysis and design	9
1.3 CORBA architecture	10
1.4 The Object Management Group (OMG)	14
1.5 Desirable properties of the object system	15
1.6 Base requirements and nature of an ORB	17
Part I Basic CORBA Programming	
Chapter 2 Getting started with CORBA	23
2.1 A simple application	23
2.2 Programming steps	24
2.3 The IDL specification	24
2.4 Compiling the IDL interface	25
2.4.1 The C++ produced	26
2.5 Implementing the interface	27
2.6 Providing a server	31
2.7 Registering the server	32
2.8 Writing a client	32
2.9 Error handling	35
2.9.1 Integration with C++ exceptions	35
	xv

2.10	Distributing the client and server	37
2.11	Summary of programming steps	38
2.12	Possible extensions to the example	39
Chapter 3	Introduction to CORBA IDL	45
3.1	IDL interfaces	45
3.2	One-way operations	48
3.3	Modules	49
3.4	Exceptions	50
3.5	Inheritance	51
3.6	The basic types of IDL	53
3.7	Constructed types	54
3.7.1	Structures	55
3.7.2	Enumerated types	56
3.7.3	Discriminated unions	56
3.8	Arrays	57
3.9	Template types	58
3.9.1	Sequences	58
3.9.2	Strings	60
3.10	Constants	61
3.11	Typedef declaration	62
3.12	Type definitions in interfaces	62
3.13	Forward references; self-referential types	63
3.14	The preprocessor	64
3.15	NamedValue	64
3.16	Context clause	65
3.17	The <code>orb.idl</code> include file	66
3.18	Identifier rules and conventions	66
3.19	Example IDL	67
Chapter 4	IDL to C++ mapping of simple types	69
4.1	Basic types and enums	70
4.1.1	Parameter passing table for basic types	72
4.1.2	Memory management for basic types	72
4.2	Interfaces, scopes, modules, and constants	74
4.3	Strings	77
4.3.1	Parameter passing table for strings	81
4.3.2	Memory management for strings	81
4.4	Object references	89
4.4.1	Parameter passing table for object references	93

4.4.2	Memory management for object references	95
4.4.3	Widening and narrowing object references	100
4.4.4	Extra member and static functions	102
4.5	Allocation and de-allocation of strings and object references	104
Chapter 5	IDL to C++ mapping of compound types	105
5.1	Structs	105
5.1.1	Parameter passing table for structs	109
5.2	Sequences	110
5.2.1	Bounded sequences	110
5.2.2	Unbounded sequences	112
5.2.3	Parameter passing table for sequences	115
5.3	Arrays	117
5.3.1	Parameter passing table for arrays	118
5.4	Unions	120
5.4.1	Parameter passing table for unions	123
5.5	Allocation and de-allocation	124
5.6	C++ keywords	125
Chapter 6	The cinema example	127
6.1	Files generated by the IDL compiler	129
6.2	A client program	132
6.3	The server: implementing interfaces	136
6.3.1	The BOAImpl approach	136
6.3.2	The TIE approach	137
6.4	The server: coding the implementation classes	138
6.4.1	The BOAImpl approach	138
6.4.2	Outline of the cinema implementation (BOAImpl approach)	141
6.4.3	The TIE approach	145
6.4.4	An example of using the TIE approach	146
6.5	The server: main function and object creation	148
6.5.1	The BOAImpl approach	148
6.5.2	The TIE approach	149
6.5.3	Initialization of the ORB	149
6.5.4	<code>impl_is_ready()</code>	150
6.5.5	Construction and markers	151
6.6	Registration, activation, and the daemon	152
6.7	Debugging	153
6.8	Execution trace	153

6.9	Comparison of the TIE and BOAImpl approaches	156	Part 2 Other Object Systems and Languages	
6.9.1	Wrapping existing code	156	Chapter 11 CORBA and OLE integration	207
6.9.2	Interfaces and implementations	157	11.1 CORBA-OLE interworking: OLE to CORBA calls	209
6.9.3	Summary of comparison of the BOAImpl and TIE approaches	158	11.2 CORBA-OLE interworking: CORBA to OLE calls	210
Chapter 7	Naming and binding	159	11.3 Example usage	211
7.1	Assigning markers to objects	159	11.4 Brief comparison of CORBA and OLE	213
7.2	Binding to objects	161	Chapter 12 Java and CORBA	215
7.3	The CORBA Naming Service	162	12.1 Java	215
7.3.1	Terminology and the CosNaming module	164	Part 3 Dynamic Aspects of CORBA	
7.3.2	Format of names within the Naming Service	164	Chapter 13 TypeCode	223
7.3.3	The NamingContext interface	165	13.1 The IDL type TypeCode	224
7.3.4	Listing a naming context	169	13.2 Implementation of TypeCode	227
7.3.5	Exceptions raised by operations in NamingContext	170	13.3 Use of TypeCode	228
7.3.6	The BindingIterator interface	171	13.3.1 Use of CORBA::TypeCode in type CORBA::Any	229
7.4	Examples of using the Naming Service	172	13.3.2 Use of CORBA::TypeCode when interrogating the Interface Repository	230
7.4.1	Creating a naming context graph and binding names	173	Chapter 14 Type any	231
7.4.2	Resolving names	174	14.1 Using operator <=< () to insert into an any	232
7.5	Other features	176	14.2 Using operator >>= () to interpret an any	234
Chapter 8	Registration and activation of servers	177	14.3 Other ways to construct and interpret an any	236
8.1	Registering servers	177	14.3.1 Inserting and extracting boolean, octet, and char	236
8.2	Activation modes	179	14.3.2 Insertion and extraction of bounded (and unbounded) strings	237
Chapter 9	Exception handling	187	14.3.3 Insertion and extraction of object references	238
9.1	The IDL definitions	187	14.3.4 Inserting and extracting array types	238
9.1.1	The generated code	188	14.3.5 Inserting values at construction time	239
9.2	The client: handling exceptions	188	14.3.6 Low-level access to a CORBA::Any	240
9.2.1	Handling specific system exceptions	191	14.4 Summary of insertion and extraction mechanisms	241
9.3	The server: throwing an exception	192	Chapter 15 Dynamic Invocation Interface	243
9.3.1	Other aspects of system exceptions	193	15.1 Example	244
Chapter 10	Inheritance	195	15.2 Using the DII	244
10.1	Example using inheritance	195	15.2.1 Obtaining an object reference	246
10.1.1	Usage from a client	197	15.3 Using the DII: CORBA compliant approach	247
10.1.2	The implementation classes	199	15.3.1 Creating a request	247
10.2	Multiple inheritance of IDL interfaces	202	15.3.2 Setting up a request using _create_request ()	247

15.3.3	Making a request	249
15.3.4	Setting up a request to read or write an IDL attribute	249
15.3.5	Interrogating a request	250
15.4	An alternative way of using the Dll: the Orbix stream-like interface	250
15.4.1	Operation results	251
15.4.2	Inserting and extracting user-defined types	252
15.5	One-way calls	252
15.6	Deferred synchronous requests	253
Chapter 16	Interface repository	255
16.1	Structure of the Interface Repository data	256
16.1.1	Retrieving information from the Interface Repository	259
16.1.2	Modifying information in the Interface Repository	260
16.2	The interface to the Interface Repository	260
16.2.1	Simple types	261
16.2.2	Class hierarchy and abstract base interfaces	262
16.3	Abstract interfaces in the IFR	262
16.3.1	Interface IRObjct (abstract)	263
16.3.2	Interface IDLType (abstract)	264
16.3.3	Interface TypedefDef (abstract)	264
16.3.4	Interface Contained (abstract)	264
16.3.5	Interface Container (abstract)	267
16.4	Non-abstract interfaces in the IFR	271
16.4.1	Repository	271
16.4.2	Interface ModuleDef	273
16.4.3	Interface ConstantDef	274
16.4.4	Interface StructDef	274
16.4.5	Interface UnionDef	275
16.4.6	Interface EnumDef	277
16.4.7	Interface AliasDef	277
16.4.8	Interface PrimitiveDef	278
16.4.9	Interface StringDef	279
16.4.10	Interface SequenceDef	279
16.4.11	Interface ArrayDef	280
16.4.12	Interface ExceptionDef	280
16.4.13	Interface AttributeDef	281
16.4.14	Interface OperationDef	282
16.4.15	Interface InterfaceDef	283
16.5	Example of using the Interface Repository	285

Chapter 17	Introduction to Dynamic Skeleton Interface	289
17.1	Architecture of the DSI support	291
Part 4	Orbix-specific Aspects	
Chapter 18	Filters	297
18.1	Introduction to per-process filters	297
18.2	Introduction to per-object filters	301
18.3	Per-process filters	302
18.3.1	An example per-process filter	303
18.3.2	Installation of the per-process filter	304
18.3.3	Raising an exception in a filter	304
18.3.4	Piggybacking extra data to the request buffer	305
Chapter 19	Smart proxies	307
19.1	Proxy factories	309
19.2	Smart proxy example	312
19.3	Handling changing seat prices	315
19.3.1	Treating the cache as a hint	316
19.3.2	Invalidating the cache	320
19.3.3	Making the call-backs using OrbixTalk	327
Chapter 20	Loaders: support for persistent objects	329
20.1	Specifying a loader for an object	331
20.2	Connection between loaders and object naming	332
20.3	Loading objects: load()	334
20.4	Saving objects: save()	334
20.5	Aspects of writing a loader	335
20.6	Example loader	336
20.7	Summary	342
Chapter 21	Multi-threaded servers	345
21.1	Benefits of multi-threaded clients and servers	345
21.2	How to start threads in servers and clients	349
21.2.1	Creating a thread to handle a request	350
21.3	Models of thread support	352
Chapter 22	Technology integration	355
22.1	Talk and Listen: OrbixTalk	356
22.1.1	Example of using OrbixTalk	359

22.1.2	OrbixTalk MessageStore	361
22.2	Orbix and DBMSs	362
Part 5 CORBAServices		
Chapter 23	CORBAServices 1	377
23.1	Event Service	377
23.2	Security Service	386
23.3	Transaction Service (OTS)	394
23.3.1	IDL interfaces	399
23.4	Trading Service	405
Chapter 24	CORBAServices 2	413
24.1	Life Cycle Service	413
24.2	Externalization Service	416
24.3	Licensing Service	420
24.4	Time Service	422
Chapter 25	CORBAServices 3	429
25.1	Property Service	429
25.2	Relationship Service	435
25.3	Concurrency Control Service	443
25.4	Persistent Object Service	446
25.5	Query Service	451
Part 6 Point of sale example and Conclusions		
Chapter 26	Point of sale example	459
26.1	Introduction to the example	459
26.2	IDL type definitions	460
26.3	The IDL for the depot component	461
26.4	The IDL of the store component	463
26.5	OMT diagram	465
26.6	Implementation of the point of sale terminal	465
26.7	Implementation of the store	466
26.8	Implementation of the depot	467
26.9	Extending the example	468
26.9.1	Reducing the number of calls to the store	468
26.9.2	Smart proxies: caching	469
26.9.3	Overlap between call reductions and caching	469
26.9.4	Smart proxies: extended interface	470
26.9.5	Filters: activity of each point of sale terminal	471

Chapter 27	Conclusions	473
Appendix A	IDL interfaces for selected CORBAServices	479
A.1	Event Service	479
A.2	Transaction Service (OTS)	482
A.3	Trading Service	484
A.4	Life Cycle Service	486
A.5	Externalization Service	487
A.6	Time Service	490
A.7	Property Service	493
A.8	Relationship Service	497
A.9	Concurrency Control Service	501
A.10	Persistent Object Service	503
A.11	Query Service	506
References		510
Index		511