



Index

A

- Abstract pipes, 72
- ACID (atomic, consistent, isolated, and durable), 484
- ACT (Asynchronous Completion Token), 418, 472
- ActiveEnterprise, *see* TIBCO ActiveEnterprise
- Activity diagrams, 21–22, 319
- Adapters, 16, 19–20, 31–32, 86, 129–131, 140
 - connecting to existing systems, 344
 - database, 344
 - message bus, 139
 - messaging systems, 102
 - Web services, 132
- Address Change message, 31
- Addresses, 30–32
- Aggregate interface, 280
- Aggregating
 - loan broker system strategies, 368
 - responses to single message, 298–300
- Aggregation algorithm, 270
- Aggregator class, 276–279
- Aggregator pattern, 24–27, 173, 226–227, 268–282, 352
 - aggregation algorithm, 270
 - collect data for later evaluation algorithm, 273
 - completeness condition, 270
 - composed message processor, 296
 - condense data algorithm, 273
 - correlation, 270
 - correlation identifiers, 270–271
 - event-driven consumers, 278
 - External event strategy, 273
 - first best strategy, 273
 - implementation, 270–272
 - initialized, 274
- JMS (Java Messaging Service), 276–282
 - listing active aggregates, 270
 - listing closed out aggregates, 271
 - loan broker, 275
 - loan broker system, 363, 368
 - loan broker system (ActiveEnterprise), 446–447, 458
 - loan broker system (MSMQ), 402, 422, 424
 - as missing message detector, 275–276
 - out-of-order messages, 284
 - parameters and strategy, 274
 - Publish-Subscribe Channel pattern, 22
 - scatter-gatherers, 300
 - selecting best answer algorithm, 273
 - sequentially numbered child messages, 262
 - splitters and, 274
 - stateful, 269
 - strategies, 272–274
 - timeout strategy, 272
 - timeout with override strategy, 273
 - wait for all strategy, 272
- Apache Axis, 371, 376–378
- Application integration
 - application coupling, 39–40
 - criteria, 39–41
 - data formats, 40
 - data sharing, 40
 - data timeliness, 40
 - encapsulation, 51–52



- Application integration, *continued*
 - file transfer, 41, 43–46
 - intrusiveness, 40
 - messaging, 41, 53–56
 - options, 41–42
 - reliability, 41
 - remote communication, 41
 - remote procedure invocation, 41, 50–52
 - shared database, 41, 47–49
 - sharing functionality, 40
 - technology selection, 40
 - Application layer, 88
 - Applications
 - automatically consuming messages, 498
 - brokering between, 82–83
 - channels, 61
 - client for each messaging system, 134
 - as client of messaging server, 95–96
 - collaborating behaviorally, 55
 - communicating with messaging, 60–66
 - communicating with simple protocol, 127
 - connecting to messaging system, 56
 - consistency, 48
 - consuming messages, 494
 - coupling, 39–40
 - data integrity, 51
 - data types, 88
 - deadlocks, 49
 - decoupling, 88–89
 - deleting files, 45
 - design and encapsulation, 50
 - different conceptual models, 54
 - errors, 117
 - exchanging data, 127
 - explicitly communicating with other applications, 323
 - file-naming conventions, 45
 - files, 44
 - handling different aspects of enterprise, 43
 - integration problems, 117
 - invalid request, 117
 - invoking procedures in, 145–146
 - logical entities, 88
 - messages, 67
 - much more decoupled, 55
 - multiple interfaces to data, 51
 - operating independently, 137–138
 - physical representation of data, 86
 - proprietary data models and data formats, 85
 - semantic dissonance, 47, 54
 - sharing databases, 47–49
 - sharing information, 43, 50–52
 - specific core function, 2
 - spreading business functions across, 2
 - as standalone solution, 127
 - tightly coupled, 39–40
 - transferring data between, 147–150
 - transmitting events between, 151–153
 - two-way conversation, 100
 - Application-specific messages, 20
 - AppSpecific property, 288, 290, 405, 424
 - Architectural patterns, 225, 228
 - Aspects, 219, 221
 - Asynchronous callback, 155–156
 - Asynchronous Completion Token pattern, 167, 472
 - Asynchronous message channels, 27
 - Asynchronous messaging, 71
 - AsyncRequestReplyService class, 408–409
 - Attach() method, 207–209, 212
 - Auction class, 276, 279–280
 - Auction versus distribution, 366–368
 - AuctionAggregate class, 276, 278–280
 - Auction-style scatter-gathers, 298
 - Axis server, 376
- B**
- BAM (Business Activity Monitoring), 537
 - BankConnection class, 422–423
 - BankConnectionManager class, 438
 - BankGateway class, 426
 - BankName parameter, 410
 - BankQuoteGateway class, 379
 - BeginReceive method, 234, 292
 - Bid class, 276
 - Bidirectional channels, 100
 - Big-endian format, 12–13
 - Billing addresses, 30
 - BitConverter class, 12
 - BizTalk Mapper editor, 93
 - BizTalk Orchestration Manager, 320–321

- Blocking gateways, 470–471
- Body, 67
- Body property, 68
- BodyStream property, 68, 591
- BodyType property, 68
- BPEL4WS (Business Process Execution Language for Web Services), 318, 634
- Bridges, 134–136
- Broadcasting
 - document messages, 148
 - messages, 106–110
 - messages to multiple recipients, 298–300
- Buffers, 286–288
- Business applications, 1, 129
- Business logic adapters, 129
- Business object identifier, 166
- Business tasks, 166
- Business-to-business integration, 9
- Byte streams, 12, 66
- BytesMessage subtype, 68

- C**
- C#
 - content-based routers, 233–234
 - delegates, 84
 - dynamic recipient lists, 256–258
 - dynamic routers, 246–248
 - filters, 76–77
 - routers, 83–84
 - smart proxies, 561–568
 - splitting XML order document, 262–267
- CanHandleLoanRequest method, 422
- Canonical Data Model pattern, 67, 90, 130, 355–360
- Canonical data models, 20, 31, 113
 - ActiveEnterprise, 360
 - data format dependencies, 359
 - designing, 358–359
 - double translation, 358
 - indirection between data formats, 356
 - message bus, 140
 - multiple applications, 357
 - transformation options, 357–358
 - WSDL, 359–360
- Canonical messages, 20
- Chain of Responsibility pattern, 231, 308
- Chaining
 - envelope wrappers, 332
 - gateways, 414, 472–473
 - request-reply message pairs, 166–167
 - transformations, 89–90
- Change notification, 151
- Channel Adapter pattern, 63, 86, 97, 102, 127–132, 134–135, 139
- Channel Purger pattern, 572–575
- Channels, 14–15, 20, 26, 57, 60–66, 359
 - acting like multiple channels, 63
 - adapters, 127–132
 - asynchronous, 27
 - bidirectional, 100
 - concurrent threading, 213
 - cost of, 63
 - crash proof, 101–102
 - data types, 101, 112, 220–221
 - datatyped, 111–114
 - dead letter, 119–121
 - dead messages, 101
 - decisions about, 101–102
 - defining for recipients, 250–251
 - deployment time, 62
 - design in Publish-Subscribe example, 219–222
 - designing, 62–63
 - determining set of, 100
 - dynamic, 100
 - for each aspect, 221
 - eliminating dependences, 327–328
 - FIFO (First-In, First-Out), 74
 - fixed set of, 99–100
 - hierarchical, 100
 - input, 107
 - invalid messages, 63, 101, 115–118
 - item number as address, 231
 - JMS, 64
 - message priorities, 113
 - message sequences, 172
 - message types, 78
 - messages, 15
 - mixing data types, 63
 - MSMQ, 65



- Channels, *continued*
 - multiple consumer coordination, 508–509
 - multiple data types sharing, 113
 - multiple receivers, 103–104
 - names, 63
 - non-messaging clients, 102
 - notifying subscriber about event once, 106
 - number needed, 220–222
 - one-to-one or one-to-many relationships, 101–102
 - one-way, 154
 - output, 107
 - persistent, 63, 102
 - Pipes and Filters architecture, 72
 - planning, 61–62
 - point-to-point, 103–105
 - practical limit to, 63
 - preventing more than one receiver monitoring, 103
 - publish-subscribe, 106–110
 - quality-of-service, 113
 - queuing requests, 14
 - routing, 79
 - separating data types, 63–64
 - static, 99
 - subscribing to multiple, 108
 - subscribing to relevant, 237
 - themes, 99–100
 - TIB/RendezVous Transport, 448
 - transmitting units of data, 66
 - two-way, 154
 - unidirectional, 100
- Channel-specific endpoints, 96–97
- Channel-to-RDBMS adapters, 131
- Child messages, 262
- Claim Check pattern, 27, 90, 173, 346–351
- Class diagrams, 88
- Client-Dispatcher-Server pattern, 246
- Clients, 62
 - concurrently processing messages, 502
 - non-messaging, 102
 - transaction control, 484–485
- CLR (Common Language Runtime), 110
- Coarse-grained interfaces, 32
- COBOL, 44
- Collect data for later evaluation algorithm, 273
- Collections and data types, 112
- Command Message pattern, 23, 67, 104, 112, 117, 139, 143–147, 153, 156
 - invoking behavior, 148
- JMS, 146
 - loan broker system (ActiveEnterprise), 452
 - routing, 140
 - SOAP (Simple Object Access Protocol), 146
- Commands, common structure of, 139
- Commercial EAI tools
 - channel adapters, 131
 - content-based routers, 234–236
 - Message Broker pattern, 82–83
 - message brokers, 326
 - message stores, 557
 - Message Translator pattern, 445
 - process Manager pattern, 445
- Common command structure, 139
- Common communication infrastructure, 139
- Communications
 - assumptions, 13–14
 - availability of components, 14
 - big-endian format, 12–13
 - data formats, 14
 - little-endian format, 12–13
 - local method invocation, 10–11
 - location of remote machine, 13
 - loose coupling, 10
 - platform technology, 13
 - reducing assumptions about, 10
 - strict data format, 13
 - TCP/IP, 12
 - tight coupling, 10
- Communications backbone, 102
- Competing Consumers pattern, 74, 97, 104, 172, 502–507
 - JMS, 505–507
 - processors, 289
- Components
 - decoupling, 72, 88–89
 - dependencies between, 71

- filtering out undesirable messages, 238
 - receiving only relevant messages, 237–238
 - two-way communication, 154
 - Composed Message Processor pattern, 25, 28, 227–228, 294–296
 - Composed routers, 225, 227–228
 - Composed service, 309–310
 - Composite messages, processing, 295–296
 - Computations and content enrichers, 339
 - ComputeBankReply method, 412
 - Computer systems
 - communications bus, 139
 - reliability, 124
 - ComputeSubject method, 235–236
 - Concurrency, 368–369
 - Concurrent threading, 213
 - Condense data algorithm, 273
 - Conflict resolution, 248
 - Consumers, 62, 515–516
 - Content, 111
 - Content Enricher pattern, 24–25, 90, 336–341
 - loan broker system, 363
 - loan broker system (ActiveEnterprise), 447
 - loan broker system (Java), 372
 - Content Filter pattern, 75, 90, 342–345
 - Content-Based Router pattern, 22–24, 81–82, 114, 225–226, 230–236
 - C#, 233–234
 - commercial EAI tools, 234–236
 - implementing functionality with filters, 240–242
 - knowledge about every recipient, 308
 - modifying for multiple destinations, 237–238
 - MSMQ, 233–234
 - reducing dependencies, 232–233
 - routing messages to correct validation chain, 303–305
 - routing messages to dynamic list of recipients, 249–250
 - routing rules associated with recipient, 308
 - special case of, 238
 - TIB/MessageBroker, 234–236
 - Context-Based Router, 82
 - ContextBasedRouter class, 594
 - Contivo, 93
 - Control Box pattern, 82
 - Control Bus pattern, 35, 407, 540–544
 - Control channels and dynamic routers, 244
 - ControlReceiver class, 594–595
 - Conway's Law, 3
 - CORBA, 4, 10
 - Correlation
 - aggregators, 270
 - process managers, 315–316
 - Correlation Identifier pattern, 115, 143, 161, 163–169, 172, 197, 206
 - loan broker system (ActiveEnterprise), 457, 459–469
 - loan broker system (MSMQ), 405, 420–421, 439
 - replier, 195, 205
 - reply, 156
 - Correlation identifiers, 164–169, 270–271, 285, 315–316
 - CorrelationId property, 195, 205
 - CreditAgencyGateway class, 379
 - CreditBureauGateway class, 476
 - CreditBureauGatewayImp class, 442
 - CreditBureauRequest struct, 414
 - CreditBureauReply struct, 418
 - Criteria and application integration, 39–41
 - CSPs (Communicating Sequential Processes), 75–76
 - Custom applications, sending and receiving messages, 127–128
- D**
- Data
 - byte stream, 66
 - changes to, 50
 - frequent exchanges of small amounts of, 52
 - inconsistencies, 47
 - knowing where to send, 55–56
 - as message sequence, 171–179
 - moving between domain objects and infrastructure, 477–480

- Data, *continued*
 - multiple interfaces to, 51
 - sharing, 53
 - storage schema and details easily changed, 54
 - storing in tree structure, 260–261
 - transferring between applications, 147–150
 - transformations, 327–329
 - transmitting large amounts, 170–171
 - units, 66
 - wrapping and unwrapping in envelope, 331–335
- Data formats, 56
 - application integration, 40
 - changing, 85–86
 - changing application internal, 357
 - content, 111
 - dependencies, 359
 - designing for changes, 180–181
 - detecting, 353–354
 - distinguishing different, 180–181
 - evolution and extensibility, 40
 - foreign key, 181
 - format document, 181–182
 - format indicator, 181–182
 - integration, 16
 - internal, 16
 - minimizing dependencies, 355–356
 - not enforcing, 47
 - proprietary, 85
 - rate of change, 352
 - standardized, 85
 - transformations, 14
 - translation, 16, 86
 - translators, 353
 - version number, 181
- Data models, proprietary, 85
- Data packets, 55, 57
- Data replication, 7, 31
- Data Representation layer, 87–88, 90
- Data sharing and latency, 40
- Data structure content, 111
- Data Structures layer, 87–88
- Data transfer mechanism, 44, 54
- Data types, 88
 - channels, 101, 112, 220–221
 - collections, 112
 - Datatype Channel pattern, 222
 - multiple sharing channel, 113
- Data Types layer, 87
- Database adapters, 129–130
- Databases
 - adapters with content filters, 344–345
 - adding trigger to relevant tables, 129
 - changes to, 50
 - extracting information directly from, 129–130
 - performance bottleneck, 49
 - sharing, 47–49
 - suitable design for shared, 48
- Datatype Channel pattern, 20, 63, 78, 101, 111–114, 139, 196, 353
 - data types, 220–222
 - Message Channel pattern, 115
 - request channel, 205
 - stock trading, 114
- DCOM, 10
- Dead Letter Channel pattern, 101, 117–121, 144
 - expired messages, 177
 - messaging systems, 120
- Dead letter queue, 120
- Dead message queue, 120
- Dead messages, 101, 117–118, 120
- Deadlocks, 49
- Debugging Guaranteed Delivery pattern, 123–124
- Decoupling, 88–89
- DelayProcessor class, 288–290, 292
- Detach() method, 207, 208
- Detour pattern, 545–546
- Detours, 545–546
- Direct translation, 358
- Dispatchers, 509–512
 - Java, 513–514
 - .NET, 512–513
- Distributed environment and Observer pattern, 208–209
- Distributed query message sequences, 173

- Distributed systems
 - asynchronous messaging, 54
 - change notification, 151
- Distribution versus auction, 366–368
- DNS (Dynamic Naming Service), 13
- Document binding, 375
- Document Message pattern, 20, 67, 104, 143–144, 147–150, 153, 156
- Document/event messages, 153
- Double translation, 358
- Duplicate messages and receivers, 528–529
- Durable Subscriber pattern, 108, 124, 522–527
 - JMS, 525–527
 - observers, 213
 - stock trading, 125, 525
- Dynamic channels, 100
- Dynamic Recipient List pattern, 34, 252–253
 - C#, 256–258
 - MSMQ, 256–258
- Dynamic Router pattern, 226, 233, 242–248
- Dynamic routing slips, 309
- DynamicRecipientList class, 256, 258
- E**
- EAI (Enterprise Application Integration)
 - applications operating independently, 137–138
 - one-minute, 11
 - process manager component, 317–318
 - suites, 2–3
- ebXML, 85
- E-mail
 - data as discrete mail messages, 67
- Encapsulation, 50–52
 - reply-to field, 161
- encodingStyle attribute, 374
- Endpoints, 19, 58, 62, 84
 - channel-specific, 96–97
 - customizing messaging API, 96
 - encapsulating messaging system from rest of application, 96
 - message consumer patterns, 464–466
 - message endpoint themes, 466–467
 - send and receive patterns, 463–464
 - sending and receiving messages, 95–97
 - transactional, 84
- EndReceive method, 204, 292
- Enterprises
 - challenges to integration, 2–4
 - loose coupling, 9–11
 - need for integration, 1
 - services, 8
- Entity-relationship diagrams, 88
- Envelope Wrapper pattern, 69, 90, 330–335
 - adding information to raw data, 332
 - chaining, 332
 - headers, 331–332
 - postal system, 334–335
 - process of wrapping and unwrapping messages, 331
 - SOAP messages, 332–333
 - TCP/IP, 333–334
- Envoy Connect, 136
- EnvoyMQ, 131
- ERP (Enterprise Resource Planning)
 - vendors, 1
- Errors, 117
- Event Message pattern, 67, 123, 143, 148, 151–153, 156
 - Observer pattern, 153
 - Publish-Subscribe Channel, 108
- Event-Driven Consumer pattern, 77, 84, 97, 498–501
 - aggregators, 278
 - gateways, 212
 - JMS MessageListener interface, 500–501
 - loan broker system (MSMQ), 417–418
 - .NET ReceiveCompletedEventHandler delegate, 501
 - pull model, 217
 - replier, 195, 204
- Event-driven gateways, 471–472
- Events
 - content, 152
 - Guaranteed Delivery pattern, 152
 - Message Expiration pattern, 152
 - notify/acknowledge, 156



Events, *continued*
notifying subscriber once about, 106
timing, 152
transmitting between applications,
151–153
Exceptions, 156, 473
Expired messages, 176–179
External event strategy, 273
External packages and schemas, 49

F

FailOverHandler class, 599–600
FIFO (First-In, First-Out) channels, 74
File formats, standard, 44
File transfer, 33, 41, 43–46
File Transfer pattern, 50, 147
decoupling, 53–54
multiple data packets, 53–54
not enforcing data format, 47
reacting to changes, 50
sharing data, 47, 53
Files, 44–45
Filtering
built-in messaging system functions,
239–240
messaging, 71
reactive, 233
splitters, 344
Filters, 58, 71–72, 226, 238
aggregators, 269–270
combining, 227
composability, 312
connection with pipes, 72
decoupling, 79
directly connecting, 78
eliminating messages not meeting
criteria, 226
generic, 75
implementing router functionality,
240–242
loan broker system, 367
multiple channels, 72–73
parallelizing, 74
versus recipient lists, 254–255
sequence of processing steps as
independent, 301–302
single input port and output port, 72

stateful, 239
stateless, 239
Fine-grained interfaces, 32
First best strategy, 273
Fixed routers, 81
Foreign key, 181
Format document, 181–182
Format Indicator pattern, 112, 114,
180–182
Formatter property, 234

G

Gateways, 469
abstracting technical details, 403
asynchronous loan broker gateway
(MSMQ), 475–476
blocking, 470–471
chaining, 414, 472–473
event-driven, 471–472
Event-Driven Consumer pattern, 212
exceptions, 473
generating, 473–474
between observer and messaging
system, 212
pull model, 215–217
sending replies, 217–218
between subject and messaging system,
211
testing, 475
generateGUID method, 457
Generic filters, 75
getaddr method, 93
GetCreditHistoryLength method,
441–442
GetCreditScore method, 414, 420,
441–442, 477
GetLastTradePrice method, 146, 150
GetRequestBodyType method, 407, 409,
412
GetState() method, 207–209, 214, 218
getStateRequestor method, 218, 219
GetTypedMessageBody method, 407
Guaranteed delivery
built-in datastore, 123
debugging, 123–124
events, 152
large amount of disk space, 123

- redundant disk storage, 124
- stock trading, 124–125
- testing, 123–124
- WebSphere MQ, 126
- Guaranteed Delivery pattern, 102, 113, 122–126, 176
- GUIDs (globally unique identifiers), 285
- GUIs and message bus, 140

H

- Half-Sync/Half-Async pattern, 472, 534
- Header, 67
- Hierarchical channels, 100
- Host Integration Server, 135–136
- HTTP Web services, 51
- Hub-and-spoke architecture, 228, 313–314, 324–326

I

- ICreditBureauGateway interface, 442
- Idempotent Receiver pattern, 97, 528–531
- Idempotent receivers, 252, 529–531
- IMessageReceiver interface, 403, 442
- IMessageSender interface, 403, 442
- Incoming messages output channel criteria, 81
- Information Portal scenario, 32
- Information portals, 6
- Initialized aggregators, 274
- Integration
 - application, 39–56
 - big-endian format, 12–13
 - broad definition, 5
 - business-to-business, 9
 - challenges, 2–4
 - channels, 14–5
 - data formats, 16
 - data replication, 7
 - distributed business processes, 8–9
 - existing XML Web services standards, 4
 - far-reaching implications on business, 3
 - information portals, 6
 - limited amount of control over participating applications, 3
 - little-endian format, 12–13
 - location of remote machine, 13
 - loose coupling, 9–11

- loosely coupled solution, 15–16
- message-oriented middleware, 15
- messages, 15
- middleware, 15
- need for, 1–2
- patterns, 4–5
- redundant functionality, 7
- remote data exchange into semantics as
 - local method call, 10
- removing dependencies, 14–15
- routing, 16
- semantic differences between systems, 4
- shared business functions, 7–8
- significant shift in corporate politics, 3
- skill sets required by, 4
- SOAs (service-oriented architectures), 8
- standard data format, 14
- standards, 3–4
- systems management, 16
- tightly coupled dependencies, 11–14
- user interfaces, 129
- Integrators and files, 44–45
- Interfaces, 32
 - loan broker system (Java), 371–372
- Internal data formats, 16
- Invalid application request, 117
- Invalid Message Channel pattern, 101, 115–118, 196–197, 205–206
 - loan broker system (MSMQ), 405
 - messages out of sequence, 172
 - queues, 233
- Invalid messages, 23, 63, 101, 115–118, 120
 - application integration problems, 117
 - ignoring, 116
 - JMS specification, 118
 - monitoring, 117
 - receiver context and expectations, 117
 - receivers, 120
 - Request-Reply example, 196–197
 - stock trading, 118
- InvalidMessenger class, 196, 205
- Inventory Check message, 26
- Inventory systems, 22–23
- IsConditionFulfilled method, 84
- Iterating splitters, 260–261
- Iterator, 261

J

J2EE

- EJBs (Enterprise JavaBeans), 535
- messaging systems, 64

- j2eeadmin tool, 64

Java

- dispatchers, 513–514
- document messages, 149
- event messages, 152
- loan broker system, 371–400

- Java RMI, 10

- JAX-RPC specification, 375

JMS (Java Messaging Service)

- aggregators, 276–282
- channel purgers, 574–575
- channels, 64
- command message, 146
- competing consumers, 505–507
- correlation identifiers, 167
- Correlation-ID property, 167
- document messages, 148
- Durable subscribers, 525–527
- event messages, 152
- expired messages, 178
- invalid messages, 118
- mappers, 483
- message selector, 521
- message sequences, 174
- MessageListener interface, 500–501
- messages, 68
- multiple message systems, 133
- persistent messages, 125–126
- point-to-point channels, 104–105
- producer and consumer, 97
- Publish-Subscribe example, 207–208
- Publish-Subscribe Channel pattern, 109, 124, 186
- receive method, 496
- Reply-To property, 161
- requestor objects, 157–158
- Request-Reply example, 118, 187–197
- Request/Reply pattern, 157–158
- return addresses, 161
- Time-To-Live parameter, 178
- transacted session, 489

- JndiUtil JNDI identifiers, 191
- JWS (Java Web Service) file, 378

K

- Kahn Process Networks, 74

- Kaye, Doug, 9

L

- Large document transfer message sequences, 173

- Legacy application routing slips implementation, 306

- Legacy platform and adapters, 131

- LenderGateway class, 379

- Listens, 62

- little-endian format, 12–13

Loan broker system

- ActiveEnterprise, 445–462
- addressing, 366–368
- aggregating strategies, 368
- Aggregator pattern, 363, 368
- aggregators, 275
- asynchronous timing, 364–366
- bank component, 578
- Content Enricher pattern, 363
- control buses, 544
- credit bureau component, 578
- credit bureau failover, 579, 592–595
- designing message flow, 362–364
- distribution versus auction, 366–368
- enhancing management console, 595–602
- instrumenting, 578–579
- Java, 371–400
- loan broker component, 578
- loan broker quality of service, 578–587
- management console, 578, 579
- managing concurrency, 368–369
- Message Channel pattern, 367–368
- Message Filter pattern, 367
- Message Translators pattern, 364
- MSMQ, 401–444
- normalizer pattern, 364
- obtaining loan quote, 361–362
- patterns, 363
- Point-to-Point pattern, 368
- process managers, 320
- Publish-Subscribe Channel pattern, 363, 366–368
- Recipient List pattern, 366–367

- recipient lists, 256
- Scatter-Gather pattern, 363, 366
- scatter-gatherers, 299
- Selective Consumer pattern, 367
- sequencing, 364–366
- synchronous implementation with Web services, 371–400
- synchronous timing, 364–366
- system management, 577–602
- test client component, 578
- verifying credit bureau operation, 579, 587–592
- wire taps, 549
- XML Web services, 371–400
- Loan broker system (ActiveEnterprise)
 - Aggregator pattern, 446–447, 458
 - architecture, 445–447
 - Command Message pattern, 452
 - Content Enricher pattern, 447
 - Correlation Identifier pattern, 457, 459–460
 - design considerations, 455
 - execution, 460–461
 - implementing synchronous services, 452–454
 - interfaces, 451–452
 - managing concurrent auctions, 459–460
 - Message Translator pattern, 457–458
 - process model implementation, 456–459
 - Publish-Subscribe pattern, 446
 - Request-Reply pattern, 446, 452
 - Return Address pattern, 452
- Loan broker system (Java), 379–381
 - accepting client requests, 378–384
 - Apache Axis, 376–378
 - Bank1.java file, 393–394
 - Bank1WS.jws file, 395
 - Bank.java file, 391–392
 - BankQuoteGateway.java file, 390–391, 396
 - client application, 396–397
 - Content Enricher pattern, 372
 - CreditAgencyGateway.java file, 385–386
 - CreditAgencyWS.java file, 386–388
 - implementing banking operations, 394–3945
 - interfaces, 371–372
 - JWS (Java Web Service) file, 378
 - LenderGateway.java file, 389–390
 - Message Translators pattern, 372
 - Normalizer pattern, 372
 - obtaining quotes, 388–3889
 - performance limitations, 399–400
 - Recipient List pattern, 372
 - running solution, 397–399
 - Service Activator pattern, 372, 379
 - service discovery, 379
 - solution architecture, 371–372
 - Web services design considerations, 372–376
- Loan broker system (MSMQ), 401
 - accepting requests, 428–431
 - Aggregator pattern, 402, 422, 424
 - bank design, 410–412
 - bank gateway, 421–428
 - Bank.cs file, 411–412
 - base classes, 405–409
 - Control Bus pattern, 407
 - Correlation Identifier pattern, 405, 420–421, 439
 - credit bureau design, 412–413
 - credit bureau gateway, 414–421
 - CreditBureau.cs file, 413
 - CreditBureauGateway.cs file, 418–420
 - designing, 413–431
 - Event-Driven Consumer pattern, 417–418
 - external interfaces, 401–402
 - IMessage Sender.cs file, 403–404
 - improving performance, 435–540
 - Invalid Message Channel pattern, 405
 - limitations, 443–444
 - LoanBroker.cs file, 430–431
 - Message Translator pattern, 402
 - message types for bank, 410
 - Messaging Gateway pattern, 402–405
 - MQService.cs file, 406–409
 - Process Manager pattern, 402, 434
 - Recipient List pattern, 402, 422, 424–425
 - refactoring, 431–434

- Loan broker system (MSMQ), *continued*
 - Return Address pattern, 405
 - Scatter-Gather pattern, 402, 422
 - Service Activator pattern, 412
 - testing, 440–443
 - LoanBroker class, 428–431
 - LoanBrokerPM class, 433–434
 - LoanBrokerProcess class, 432–433
 - LoanBrokerProxy class, 582–583
 - LoanBrokerProxyReplyConsumer class, 584–585
 - LoanBrokerProxyRequestConsumer class, 584
 - LoanBrokerWS class, 379
 - Local invocation, 145
 - Local method invocation, 10–11
 - Local procedure calls, 52
 - Logical entities, 88
 - Loose coupling, 9–11
- M**
- ManagementConsole class, 597–598
 - MapMessage subtype, 68
 - Mapper pattern, 480
 - Mapper task, 457–458
 - Mappers, 480–483
 - match attribute, 93
 - MaxLoanTerm parameter, 410
 - Mediator pattern, 509
 - Mediators, 481
 - Message Broker pattern, 228, 322–326
 - brokering between applications, 82–83
 - central maintenance, 324
 - commercial EAI tools, 82–83, 326
 - hierarchy, 325
 - stateless, 324–325
 - translating message data between applications, 325–326
 - Message bus, 102, 139–141
 - Message Bus pattern, 64, 137–141
 - Message Channel pattern, 19, 55, 57, 62, 73, 78, 106
 - Apache Axis, 377
 - availability, 100
 - Datatype Channel pattern, 115
 - decisions about, 101–102
 - decoupling applications, 89
 - fixed set of, 99–100
 - load-balancing capabilities, 82
 - loan broker system, 367–368
 - monitoring tool, 108
 - as pipe, 66
 - security policies, 108
 - unidirectional or bidirectional, 100
 - Message class, 68
 - Message Consumer patterns, 464–466
 - Message Dispatcher pattern, 97, 113, 508–514
 - Message dispatchers, 172
 - Message Endpoint pattern, 56, 58, 61, 173
 - Apache Axis, 376
 - data format translation, 86
 - Selective Consumer pattern, 226
 - Message endpoints, 16, 62, 95–97, 134–135
 - Message Expiration pattern, 67, 108, 119, 123, 144, 176–179
 - Message Filter pattern, 75, 80, 237–242
 - Publish-Subscribe Channel pattern, 226
 - Message History pattern, 81, 551–554
 - Message ID, 166
 - Message identifiers, 285
 - Message pattern, 57–58, 78
 - Message Router pattern, 34, 58, 75, 89, 139, 225, 228
 - capabilities, 139
 - Content-Based Router pattern, 232
 - Message Filter pattern, 238
 - Message Sequence pattern, 67, 115, 144, 156, 170–175
 - Message sequences, 171–179
 - channels, 172
 - Competing Consumers pattern, 172
 - distributed query, 173
 - end indicator field, 171
 - identification fields, 171
 - identifiers, 167
 - JMS, 174
 - large document transfer, 173
 - Message Dispatcher pattern, 172
 - multi-item query, 173
 - .NET, 174
 - position identifier field, 171

- Request-Reply pattern, 172
 - sending and receiving, 172
 - sequence identifier field, 171
 - size field, 171
- Message Store pattern, 555–557
- Message stores, 26–27, 34, 556–557
- Message Translator pattern, 58
 - Channel Adapters, 130
 - commercial EAI products, 445
 - data in incoming message, 336
 - loan broker system, 364
 - loan broker system (ActiveEnterprise), 457–458
 - loan broker system (Java), 372
 - loan broker system (MSMQ), 402
 - metadata, 130
- MessageConsumer class, 191, 278, 562–563
- MessageConsumer type, 97
- MessageGateway, 414
- message-id property, 195, 205
- MessageListener interface, 195, 212, 217, 500–501
- Message-oriented middleware, 15
- Message-processing errors, 117
- MessageProducer class, 125, 191, 195
- MessageProducer type, 97
- MessageQueue class, 97, 105, 126, 167, 201, 204
- MessageQueue instance, 65
- MessageReceiverGateway class, 404
- Messages, 14–15, 66, 159
 - aggregating, 24
 - applications, 67
 - application-specific, 20
 - augmenting with missing information, 338–341
 - authentication information, 70
 - body, 67
 - breaking data into smaller parts, 67
 - broadcasting, 106–110
 - canonical, 20
 - channels, 78
 - checking in data for later use, 27
 - collecting and storing, 269–270
 - combining related to process as whole, 268–269
 - common format, 86
 - conforming to data types, 101
 - containing commands, 146
 - contents are semantically incorrect, 117
 - correlation ID, 166
 - data formats, 56
 - data packets, 57
 - dead, 101, 117–118
 - decoupling destination of, 322–323
 - decrypting, 70–71
 - delivering, 57–58
 - demultiplexing, 113
 - destination of, 80
 - different types of, 67
 - directing, 56
 - document/event, 153
 - duplicate, 70
 - elements requiring different processing, 259–260, 294–295
 - encrypted, 70
 - endpoints, 58
 - expired, 176–179
 - format data, 67
 - formatting in proprietary formats, 31
 - guaranteed delivery, 122–126
 - header, 67
 - high frequency of, 55
 - huge amounts of data, 144
 - improper datatype or format, 115
 - “incoming message massaging module,” 70
 - intent, 143
 - invalid, 101, 115–118, 120
 - JMS, 68
 - large amounts of data, 170–171
 - message ID, 166
 - messaging system, 67
 - missing properties, 115
 - monitoring, 34–36
 - multiple recipients with multiple replies, 297
 - .NET, 68
 - order ID, 24–25
 - out-of-sequence, 227, 283–284
 - peek functions, 108
 - persistent, 122–126
 - private, 358

- Messages, *continued*
- processing in type-specific ways, 113–114
 - processing steps, 71
 - public, 358
 - recombining, 226–227
 - recursive nature, 69
 - reducing data volume, 346
 - removing unimportant data from, 343–345
 - removing valuable elements from, 342–343
 - reordering, 284–293
 - response, 143–144
 - retry timeout parameter, 123
 - return address, 161
 - routing, 58, 80, 85
 - routing slips, 305–306
 - routing to correct recipient based on content, 232–236
 - semantically equivalent in different format, 352–353
 - sending and receiving, 95–97
 - sent time, 178
 - sequence numbers, 285
 - simplifying structure, 343
 - slow, 144
 - SOAP, 68–69
 - splitting, 24, 226, 260–267
 - state reply, 153
 - state request, 153
 - storing data between, 28
 - storing data in central database, 27
 - storing data in tree structure, 260–261
 - testing, 34–36
 - timestamp, 177–178
 - transformation, 54, 58, 327–329
 - transformation levels, 87–88
 - two-way, 154
 - types, 68, 78
 - unable to deliver, 118–121
 - update, 153
 - Wire Tap, 27
- MessageSenderGateway class, 404
- Messaging, 41, 53–56
- asynchronous, 54, 71
 - basic concepts, 57–58
 - filtering, 71
 - invoking procedure in another application, 145–146
 - one-way communication, 154
 - remote procedure invocation, 156
 - remote query, 156
 - transferring data between applications, 147–150
 - transmitting discrete units of data, 66
- Messaging API, 96
- Messaging Bridge pattern, 102, 131, 133–136
- Messaging Gateway pattern, 19–20, 72, 97, 117, 211, 468–476
- loan broker system (MSMQ), 402–405
- Messaging Mapper pattern, 97, 477–483
- Messaging mappers, 357–358
- Messaging pattern, 45–46, 49, 52, 57–58, 163
- Messaging server, applications as clients of, 95–96
- Messaging services, 8
- dynamic discovery, 245
 - invoking with messaging and non-messaging technologies, 532
 - request-reply, 28–29
 - reuse, 29
 - shared business functions as, 28
- Messaging systems
- adapters, 102
 - applications communicating with, 60–66
 - built-in datastore, 123
 - channel adapters, 63
 - communicating without required data items, 336–338
 - communications backbone, 102
 - connecting application to, 56
 - connecting multiple, 133–136
 - connections, 60–61
 - Dead Letter Channel, 120
 - decoupling, 54
 - delivering messages, 57–58
 - encapsulating access to, 468–469
 - filtering built-in functions, 239–240
 - filters, 58
 - hierarchical channel-naming scheme, 63

- implementation of single function
 - spread across, 230–232
 - inconsistency, 55
 - interoperability, 133–134
 - invalid messages, 330
 - J2EE, 64
 - logical addresses, 61
 - managing channels, 95
 - messages, 67
 - pipes, 58
 - Pipes and Filters architecture, 70–77
 - planning channels, 61–62
 - receivers inspecting message properties, 79
 - reducing data volume of messages, 346
 - sending and receiving messages, 95–97
 - specific messaging requirements, 330–331
 - store-and-forward process, 122
 - uneconomical or impossible to adjust components, 79
 - valid messages, 330
 - WebSphere MQ for Java, 64–65
 - Metadata
 - management and transformations, 328–329
 - Message Translators pattern, 130
 - Metadata adapter, 130–131
 - MetricsSmartProxy class, 566
 - Meunier, Regine, 74
 - Middleware, 15
 - MIDL (Microsoft Interface Definition Language), 531
 - Missing messages
 - aggregators as detector of, 275–276
 - stand-in messages for, 287–288
 - MockQueue, 404
 - Model-View-Controller architecture, 151
 - Monitor class, 589–592
 - MonitorStatusHandler class, 598
 - MQSend class, 288
 - MQSequenceReceive class, 289
 - MQService class, 405–409, 412
 - MSMQ (Microsoft Messaging Queuing Service)
 - asynchronous loan broker gateway, 475–476
 - bridges, 135–136
 - content-based routers, 233–234
 - distribution lists, 110
 - dynamic recipient lists, 256–258
 - dynamic routers, 246–248
 - filters, 76–77
 - loan broker system, 401–444
 - maximum message size, 173
 - message channels, 65
 - multiple-element format names, 110
 - one-to-many messaging model, 109
 - persistent channels, 124
 - queues, 65
 - real-time messaging multicast, 109
 - resequencers, 288–293
 - routers, 83–84
 - smart proxies, 561–568
 - splitting order document, 264–267
 - Transactional Clients pattern, 124
 - transactional filter, 490–493
 - Multi-item queries and message sequences, 173
 - Multiple asynchronous responses, 174
 - Multiplexing, 113
- N**
- .NET
 - CLR (Common Language Runtime), 110
 - correlation identifiers, 167–168
 - Correlation-Id property, 167–168
 - delegates, 418
 - dispatchers, 512–513
 - document messages, 148
 - event messages, 152
 - expired messages, 179
 - message sequences, 174
 - MessageQueue class, 97
 - messages, 68
 - persistent messages, 126
 - point-to-point channels, 105
 - Receive method, 496–497
 - ReceiveCompletedEventHandler
 - delegate, 501
 - Request-Reply example, 118, 198–206
 - resequencers, 288–293
 - Response-Queue property, 162
 - return addresses, 162

- .NET, *continued*
 - selective consumers, 521
 - serialization and deserialization, 416
 - Time-To-Be-Received property, 179
 - Time-To-Reach-Queue property, 179
 - transactional queue, 490
 - .NET Framework, 404–405
 - .NET Framework SDK, 415
 - .NET Remoting, 10
 - Networks, inefficiencies and recipient lists, 253–254
 - Neville, Sean, 375
 - New Order message, 22, 27, 30
 - Normalizer pattern, 90, 352–354
 - loan broker system, 364
 - loan broker system (Java), 372
 - Normalizers, 353–354
 - Notify() method, 207, 208, 211, 213
 - notifyNoState() method, 217
 - Null Object, 238
- O**
- OAGIS, 85
 - ObjectMessage class, 196
 - ObjectMessage subtype, 68
 - Objects, notifying dependents of change, 207–208
 - Observer pattern, 106, 110, 151
 - distributed environment, 208–209
 - Event Message pattern, 153
 - implementing, 209–212
 - JMS Publish-Subscribe example, 207–208
 - .NET Framework, 404
 - pull model, 153
 - push model, 153
 - ObserverGateway class, 212, 218
 - Observers, 207–208
 - concurrent threading, 213
 - Durable Subscriber pattern, 213
 - implementing, 209–213
 - losing notification, 209
 - multiple aspects, 219
 - receiving messages, 213
 - reply channels, 214–215
 - subscribing and unsubscribing from channels, 213
 - OnBestQuote method, 431
 - OnCreditReply method, 431
 - OnCreditReplyEvent delegate, 476
 - OnCreditReplyEvent event, 420
 - One-minute EAI (Enterprise Application Integration) suites, 11
 - One-way channels, 154
 - OnMessage event, 404
 - onMessage method, 84, 195, 197, 212, 217, 234, 264, 278, 407–408
 - OnMsgEvent delegate, 404
 - OnReceiveCompleted method, 77, 204
 - Operating environment, 339
 - ORB (object request broker) environment, 208
 - Order ID, 24–25
 - Order Item Aggregator, 25
 - Order Item messages, 24–25
 - Order message, 24–25, 263
 - Ordered or unordered child messages, 262
 - Orders, checking status, 26–29
 - Out-of-order messages, 283–284
 - Out-of-sequence messages, 227
- P**
- Parallelizing filters, 74
 - Pattern matching, 93
 - Patterns
 - combining with scatter-gatherers, 299–300
 - comparing Process Manager pattern with, 319–320
 - loan broker system, 363
 - pattern form, xliii–xlvi
 - Peek functions, 108
 - PeekByCorrelationId() method, 168
 - Persistence, 123
 - Persistent channels, 63, 102, 126
 - Persistent messages, 122–126
 - JMS, 125–126
 - .NET, 126
 - Persistent recipient lists, 252
 - Persistent store, 29, 347–348
 - PGM (Pragmatic General Multicast), 109
 - Pipeline processing, 73–74

- Pipes, 58
 - abstract, 72
 - composability, 312
 - connection with filters, 72
 - managing state, 316
 - Message Channel pattern, 66
 - simple in-memory queue to implement, 72
- Pipes and Filters architecture, 58
 - directly connecting filters, 78
 - history of, 74–75
 - large number of required channels, 72
- Pipes and Filters pattern, 227
 - chaining transformations, 89
 - composability of individual components, 79
 - composability of processing units, 312
 - distributed, 317
 - pipeline processing, 73–74
 - processing messages, 73–74
 - processing steps, 230
 - sequence of processing steps as independent filters, 301–302
 - testability, 73
- Point-to-Point Channel pattern, 63, 73–74, 101, 124, 147, 368
- Point-to-Point channels, 20, 23, 26–27, 103–105
 - broadcasting messages, 153
 - command messages, 146
 - document messages, 148
 - eavesdropping, 107–108
 - inspecting messages, 547–550
 - JMS, 104–105
 - .NET, 105
 - request channel, 155
 - stock trading, 104
- Polling Consumer pattern, 97, 155, 494–497
- Port, 72
- Postal service
 - data as discrete mail messages, 67
 - envelope wrappers, 334–335
- Predictive routing, 80
- Private messages, 358
- Procedures, invoking in another application, 145–146
- Process definitions, 315
 - process managers creation of, 317–318
 - TIB/IntegrationManager Process Manager Tool, 449
- Process instances, 28
 - process managers, 314–315
 - TIB/IntegrationManager Process Manager Tool, 449
- Process Manager pattern, 312–321
 - commercial EAI products, 445
 - comparing with other patterns, 319–320
 - loan broker system (MSMQ), 402, 434
- Process managers, 27–31, 309, 313
 - BizTalk Orchestration Manager, 320–321
 - central, 317
 - claim checks, 350–351
 - correlation, 315–316
 - hub-and-spoke pattern, 313–314
 - keeping state in messages, 316–317
 - loan broker, 320
 - process definition, 315, 317–318
 - process instances, 314–315
 - state maintenance, 314
 - storing intermediate information, 314
 - trigger message, 313
 - versatility, 314
- Process method, 77
- Process template, 28
- Processes
 - marshaling and unmarshaling data, 66
 - passing piece of data, 66
 - synchronizing with IO (input-output), 75
- Processing
 - composite messages, 295–296
 - orders, 20–23
- Processing pipeline, 73–74
- ProcessMessage method, 76–77, 290, 292, 408–412, 431
- Processor class, 76, 290, 292
- Processors competing with consumers, 289
- Producers, 62
- Protocols, tunneling, 330

- Provider, 62
 - Public messages, 358
 - Publisher, 62
 - Publish-Subscribe Channel pattern, 62–63, 80, 101, 104, 139, 147, 153, 207, 209
 - loan broker system (ActiveEnterprise), 446
 - Publish-subscribe channels, 23, 26, 31, 33–34, 106–110, 249–250
 - announcing address changes, 220
 - basic routing, 323
 - as debugging tool, 107
 - document messages, 148
 - eavesdropping, 107–108
 - Event Message pattern, 108
 - filters versus recipient lists, 254–255
 - hierarchical structure, 239
 - implementing router functionality with filters, 240–242
 - JMS, 109, 124, 186
 - loan broker system, 363, 366–368
 - Message Filters pattern, 226
 - multiple output channels, 107
 - one input channel, 107
 - out-of-product announcements, 220
 - receiving change notification code, 211–212
 - request channel, 155
 - Scatter-Gather pattern, 228
 - special wildcard characters, 108
 - stock trading, 108–109
 - storing messages, 108
 - subscription to, 237
 - Publish-Subscribe example
 - channel design, 219–222
 - code to announce change, 210–211
 - Command Message pattern, 185
 - comparisons, 212–213
 - Datatype Channel pattern, 185
 - distributed notification between applications, 212–213
 - Document Message pattern, 185
 - Durable Subscriber pattern, 185
 - Event Message pattern, 185
 - Event-Driven Consumer pattern, 185
 - implementing observers, 209–212
 - Java using JMS, 186
 - Messaging Gateway pattern, 185
 - Observer pattern, 185
 - Publish-Subscribe Channel pattern, 185
 - pull model, 213–219
 - push model, 213–219
 - Request-Reply pattern, 185
 - Return Address pattern, 185
 - serialization, 213
 - Pull model, 153, 207–208
 - Event-Driven Consumer pattern, 217
 - gateways, 215–217
 - Publish-Subscribe example, 213–219
 - PullObserverGateway class, 218
 - PullSubjectGateway class, 217
 - Push model, 153, 207–208
 - Publish-Subscribe example, 213–219
- Q**
- Quality-of-Service Channel pattern, 113
 - Queries, 173
 - Queue instance, 64
 - Queue interface, 104
 - QueueRequestor class, 192
 - Queues
 - Invalid Message Channel pattern, 233
 - peek functions, 108
- R**
- RatePremium parameter, 410
 - Reactive filtering, 80, 233
 - ReceiveByCorrelationID() method, 168
 - ReceiveCompleted event, 404
 - ReceiveCompletedEventHandler class, 204
 - ReceiveCompletedEventHandler delegate, 501
 - Receive() method, 192, 202
 - Receivers, 62
 - communicating message type to, 112
 - content data structure and data format, 111
 - dead messages, 120
 - duplicate messages, 528–529
 - Event-Driven Consumer pattern, 97
 - idempotent receivers, 529–531
 - inspecting message properties, 79

- invalid messages, 120
- multiple on channel, 103–104
- Polling Consumer pattern, 97
- response from, 154–158
- type of messages received, 111
- ReceiveSync() method, 192, 202
- Receiving sequences, 172
- Recipient List pattern, 110, 226, 249–258
 - loan broker system (Java), 372
 - loan broker system (MSMQ), 402, 422, 424–425
- Recipient lists, 242, 250–251
 - dynamic, 252–253
 - idempotent receivers, 252
 - list of recipients, 251
 - loan broker, 256
 - network inefficiencies, 253–254
 - persistent, 252
 - versus publish-subscribe channels and filters, 254–255
 - restartable, 252
 - robustness, 252
 - routing, 439
 - scatter-gathers, 298
 - sending copy of message to all recipients, 251
 - sending preferences to, 253
 - single transaction, 252
- Recipients
 - broadcasting messages to multiple, 298–300
 - defining channel for, 250–251
 - list of, 251
 - multiple with multiple replies, 297
 - routing messages to dynamic list, 249–250
 - sending copy of message to all, 251
- Recombining messages, 226–227
- Redundant functionality, 7
- Relational databases, SQL-based, 48
- Relationships and entities, 88
- Reliability of Web services, 375–376
- Remote invocation, 145
- Remote Procedure Call pattern, 209
- Remote procedure calls, 52
- Remote Procedure Invocation pattern, 46, 49, 62, 145, 147, 151
- Remote procedure invocations, 41
 - failure of, 53–54
 - messaging, 156
 - sharing functionality, 53
 - synchronous, 163
 - two-way communication, 147
- Remote query and messaging, 156
 - Web services, 375
- Reordering messages, 284–293
- Replier class, 183, 187, 198
- Repliers, 155
 - agreeing on details, 165
 - correlation identifier, 164
 - Correlation Identifier pattern, 195, 205
 - Event-Driven Consumer pattern, 195, 204
 - Return Address pattern, 195, 204
- Replies
 - callback processor to process, 160
 - correlation identifier, 164–169, 165
 - Correlation Identifier pattern, 156
 - document messages, 148
 - exceptions, 156
 - gateway sending, 217–218
 - from multiple recipients, 297
 - one-to-one correspondence with request, 159
 - pointer or reference to request, 164
 - processing, 195
 - reassembling multiple into one, 228
 - result value, 156
 - return address, 159–162
 - token, 166
 - void, 156
 - where to send, 159–162
 - which requests they are for, 163–169
- Reply channels and observers, 214–215
- Request channel, 155, 205
- Requestor class, 183, 187, 198
- Requestor.receiveSync() method, 197
- Requestors, 62, 155
 - agreeing on details, 165
 - callback processor to process replies, 160
 - correlation identifier, 164
 - map of request IDs and business object IDs, 166

- Requestors, *continued*
 - receiving reply messages, 192, 202
 - sending request messages, 192, 202
- Request-replies
 - asynchronous callback, 155–156
 - chaining message pairs, 166–167
 - channels to transmit messages, 214
 - loan broker system (ActiveEnterprise), 446, 452
 - message sequences, 172
 - replier, 155
 - requestor, 155
 - synchronous block, 155
- Request-Reply example
 - Command Message pattern, 188
 - Correlation Identifier pattern, 184, 189, 200
 - Datatype Channel pattern, 184
 - Document Message pattern, 184, 188
 - Event Driven Consumer pattern, 184
 - Invalid Message Channel pattern, 184
 - Invalid Message example, 196–197, 205–206
 - JMS, 187–197
 - JMS API in Java J2EE, 184
 - jms/InvalidMessages queue, 187
 - jms/ReplyQueue queue, 187, 188
 - jms/RequestQueue queue, 187
 - Message Channel pattern, 184
 - MSMQ API in Microsoft .NET using C#, 184
 - .NET, 198–206
 - Point-to-Point Channel pattern, 184
 - Polling Consumer pattern, 184
 - .private\$\InvalidQueue queue, 198
 - .private\$\ReplyQueue queue, 198
 - .private\$\RequestQueue queue, 198
 - Replier class, 183, 187, 198
 - Requestor class, 183, 187, 198
 - Request-Reply code, 189–196, 200–205
 - Request-Reply pattern, 184
 - Return Address pattern, 184, 188–189, 199, 204
- Request-Reply pattern, 67, 104, 108, 143–144, 154–158
 - JMS, 157–158
 - reply channel, 100
- RequestReplyService class, 408–409, 412, 424
- Request-Response Message Exchange pattern
 - return addresses, 162
 - SOAP 1.2, 157, 162, 168–169
 - Web services, 162, 168–169
- Requests
 - correlation identifier, 165
 - messaging query, 156
 - notify/acknowledge messages, 156
 - pointer or reference to, 164
 - remote procedure invocation messages, 156
 - Return Address pattern, 100, 156
 - return addresses, 167, 195
 - sent and received timestamps, 199
 - unique ID, 166
 - which replies are for, 163–169
- Resequencer class, 289
- Resequencer pattern, 74, 164, 227, 283–293
- Resequencers, 227, 284
 - avoiding buffer overrun, 286–288
 - buffers, 286
 - internal operations, 285–286
 - MSMQ, 288–293
 - .NET, 288–293
 - out-of-sequence messages, 285–286
 - sequence numbers, 285
 - stand-in messages for missing messages, 287–288
 - throttling message producer with active acknowledgment, 287
- ResponseQueue property, 202
- Responses, 143–144
 - aggregating to single message, 298–300
 - delivered out of order, 268
 - from receivers, 154–158
- Retry timeout parameter, 123
- Return Address pattern, 115, 143, 159–162
 - loan broker system (ActiveEnterprise), 452
 - loan broker system (MSMQ), 405
 - replier, 195, 204
 - request message, 100
 - requests, 156

- Return addresses, 29, 35, 36, 159–162
 - JMS, 161
 - .NET, 162
 - Request-Response Message Exchange pattern, 162
 - requests, 167
- RIP (Routing Information Protocol), 245
- RMI (Remote Method Invocation), 10
- RosettaNet, 85
- Router slips, 308–309
- Routers, 25, 56, 58, 73, 78–84, 140, 359
 - abuse of, 81
 - architectural patterns, 225, 228
 - avoiding dependency, 243
 - built-in intelligence, 82
 - C#, 83–84
 - combining variants, 228
 - composed, 225, 227–228
 - content-based, 81–82, 225–226, 230–236
 - context-based, 82
 - Control Bus pattern, 82
 - decoupling filters, 80
 - degrading performance, 81
 - destination based on environment conditions, 82
 - destination of message, 80–82
 - dynamic, 244–248
 - eliminating dependencies, 327–328
 - filters, 238
 - fixed, 81
 - fixed rules for destination of in-coming message, 226
 - hard-coded logic, 82
 - implementing functionality with filters, 240–242
 - knowledge of all destination channels, 80
 - loosely coupled systems, 81
 - maintaining efficiency, 243
 - maintenance bottleneck, 80
 - MSMQ, 83–84
 - multiple in parallel, 81
 - parallel processing, 82
 - performance bottleneck, 81
 - selecting correct for purpose, 228–229
 - self-configuring, 244–248
 - simple, 225–227
 - stateful, 82, 227
 - stateless, 82, 233
 - variants, 81–82
- Routing, 16, 58
 - basic form, 79
 - channels, 79
 - command messages, 140
 - to correct recipient based on content, 232–236
 - flexibility, 302
 - maintaining state of sequence, 313–321
 - message flow efficiency, 302
 - moving logic to middleware layer, 323
 - recipient lists, 439
 - resource usage efficiency, 302
 - simple maintenance, 302
 - unknown non-sequential processing steps, 312–313
- Routing messages, 80, 85
 - based on criteria, 226
 - to correct translator, 353–354
 - to dynamic list of recipients, 249–250
 - with multiple elements, 259–260
 - for system management, 545–546
 - through series of unknown steps, 301–305
- Routing Slip pattern, 301–311
- Routing slips
 - acting as chain of responsibility, 308
 - binary validation steps, 307
 - as composed service, 309–310
 - decision postponed until end, 307
 - dynamic, 309
 - legacy application implementation, 306
 - limitations, 306
 - processing steps, 312
 - stateless transformation steps, 307
 - WS-Routing (Web Services Routing Protocol), 310–311
- RPC (Remote Procedure Call), 10, 51, 103
 - asynchronous messaging, 122
 - binding, 375
 - marshaling, 66
- RPC-style SOAP messaging, 149
- RPC-style Web services, 10
- Run method, 412

- S**
- SASE (Self-Addresses Stamped Envelope) pattern, 219
 - Scatter-Gather pattern, 228, 297–300
 - loan broker system, 363, 366
 - loan broker system (ActiveEnterprise), 446
 - loan broker system (MSMQ), 402, 422
 - Publish-Subscribe Channel pattern, 228
 - Scatter-gatherers, 298–300
 - Schemas, 49
 - Security and Web services, 375–376
 - Selecting best answer algorithm, 273
 - Selective Consumer pattern, 63, 119, 168, 222, 226, 239–240, 515–521
 - JMS message selector, 521
 - loan broker system, 367
 - .NET, 521
 - separating types, 520
 - Selectors, 239–240
 - Semantic dissonance, 47, 54–55
 - Semantic enrichment, 414
 - Send and receive patterns, 463–464
 - SendConsecutiveMessages method, 290
 - Senders, 62
 - communicating message type to receiver, 112
 - decoupling message destination from, 322–323
 - Send() method, 192, 202
 - SendReply method, 408, 433
 - Sent time, 178
 - Sequence identifier, 172
 - Sequence numbers, 285
 - Sequencer, 261
 - Sequencing, 364–366
 - Serializable command object, 146
 - Serialization in Publish-Subscribe example, 213
 - Service Activator pattern, 97, 117, 139, 532–535
 - Service activators, 140, 533–534
 - Axis server, 376
 - loan broker system (Java), 379
 - loan broker system (MSMQ), 412
 - Service stubs, 403
 - Service-oriented architecture (SOA), 8, 140
 - Shared business functions, 7–8
 - Shared Database pattern, 46–50, 147
 - Shared databases, 29, 41, 53
 - avoiding semantic dissonance, 55
 - unencapsulated data structure, 50
 - Sharing data, 53
 - Sharing information, 43
 - Shipping addresses, 30
 - Silly Window Syndrome, 287
 - Simple routers, 225–227, 308–309
 - SimpleRouter class, 84
 - Slow messages, 144
 - Smart proxies, 29, 35, 36, 559–560
 - C#, 561–568
 - MSMQ, 561–568
 - Smart Proxy pattern, 558–568
 - SmartProxyBase class, 563
 - SmartProxyReplyConsumer class, 565
 - SmartProxyReplyConsumerMetrics class, 566
 - SmartProxyRequestConsumer class, 564
 - SOAP (Simple Object Access Protocol)
 - binding styles, 375
 - command messages, 146
 - document messages, 148, 149–150
 - encoding style, 374
 - messages, 68–69
 - recursive nature of messages, 69
 - transport protocol, 373
 - Web services, 372–373
 - SOAP 1.2 and Request-Response Message Exchange pattern, 157, 162, 168–169
 - SOAP messages
 - envelope wrappers, 332–333
 - Request-Reply pairs, 157
 - SOAP request messages, 168, 174
 - SOAP response messages
 - correlation to original request, 168–169
 - sequencing and correlation to original request, 174–175
 - SonicMQ Bridges, 136
 - Splitter pattern, 173, 226, 259–267
 - Splitters, 24, 25
 - aggregators and, 274
 - C# XML order document, 262–267
 - filtering, 344

- iterating, 260–261
 - MSMQ XML order document, 264–267
 - ordered or unordered child messages, 262
 - static, 261
 - Splitting messages, 226, 260–267
 - SQL-based relational databases, 48
 - Stale information, 45
 - Standard file formats, 44
 - Standardized data formats, 85
 - State
 - aspects, 219
 - keeping in messages, 316–317
 - process manager maintenance, 314
 - State request messages, 153
 - Static channels, 99
 - Static splitters, 261, 343–344
 - Stock trading
 - bridges, 135
 - channel adapter, 131
 - Datatype Channel pattern, 114
 - dead letter channels, 121
 - Durable Subscriber pattern, 125
 - guaranteed delivery, 124–125
 - invalid messages, 118
 - message bus, 141
 - Publish-Subscribe Channel pattern, 108–109
 - Store-and-forward process, 122
 - StreamMessage subtype, 68
 - Structural transformations, 90–93
 - SubjectGateway class, 211, 217
 - Subscribers, 62
 - avoiding missing messages, 522–523
 - durable or nondurable, 108
 - multiple channels, 108
 - notifying once about event, 106
 - special wildcard characters, 108
 - Synchronous block, 155
 - Synchronous implementation of loan broker system, 371–400
 - Syntax layer, 88
 - System management, 537
 - analyzing and debugging message flow, 551–554
 - avoiding infinite loops, 554
 - internal faults, 569
 - leftover messages, 572–575
 - loan broker system, 577–602
 - monitoring and controlling, 538
 - observing and analyzing message traffic, 538
 - reporting against message information, 555–557
 - routing messages for, 545–546
 - testing and debugging, 539
 - tracking messages, 558–568
 - widely distributed system, 540–541
 - Systems
 - data transfer between, 87–88
 - management, 16
 - out of synchronization, 45
- T**
- Taking orders, 18–19, 24–25
 - Talks, 62
 - TCP/IP, 12–13, 88
 - ensuring in-sequence delivery of messages, 287
 - envelope wrappers, 333–334
 - tightly coupled dependencies, 11–12
 - Tee, 547
 - Template methods, 292, 404
 - TemporaryQueue class, 215
 - Test data generator, 36
 - Test data verifier, 36
 - Test Message pattern, 569–571
 - Test messages, 36, 569–571
 - Testing
 - gateways, 475
 - Guaranteed Delivery pattern, 123–124
 - loan broker system (MSMQ), 440–443
 - Text-based files, 44
 - TextMessage subtype, 68
 - TIBCO ActiveEnterprise
 - canonical data models, 360
 - loan broker system, 445–462
 - message history, 553–554
 - TIBCO Repository for Metadata Management Integration, 450–451
 - TIB/IntegrationManager Process Manager Tool, 448–450
 - TIB/MessageBroker, 234–236



- TIB/RendezVous Transport, 448
 - Tight coupling, 10, 32
 - Tightly coupled applications, 39–40
 - Tightly coupled dependencies
 - integration, 11–14
 - TCP/IP, 11–12
 - Timeout strategy, 272
 - Timeout with override strategy, 273
 - Topic interface, 109
 - TopicPublisher class, 109, 209
 - TopicSubscriber class, 109
 - Transactional Client pattern, 77, 84, 97, 131, 172, 484–493
 - JMS transacted session, 489
 - message groups, 487–488
 - message/database coordination, 488
 - message/workflow coordination, 488
 - MSMQ, 124
 - .NET transactional queue, 490
 - send-receive message pairs, 487
 - transactional filter with MSMQ, 490–493
 - Transactions, 172, 484–485
 - Transform method, 265
 - Transformations, 54, 58
 - chaining, 89–90
 - changing application internal data format, 357
 - changing at individual level, 90
 - content enrichers, 338–341
 - Data Representation layer, 87, 88
 - Data Structures layer, 87, 88
 - Data Types layer, 87, 88
 - decoupling levels, 88–89
 - dragging and dropping, 94
 - eliminating dependencies, 327–328
 - external translators, 357–358
 - implementing messaging mapper, 357–358
 - levels of, 87–88
 - metadata management, 328–329
 - at multiple layers, 89
 - options, 357–358
 - outside of messaging, 329
 - structural, 90–93
 - Transport layer, 87–88
 - visual tools, 93–94
 - XML documents, 90–93
 - Translators, 20, 23, 31, 56, 85–94
 - chaining multiple units, 89–90
 - data formats, 353
 - double translation, 358
 - external, 357–358
 - versus mappers, 482
 - resolving data format differences, 355–356
 - routing messages to correct, 353–354
 - Transport protocols, 87
 - Tree structure, 260–261
 - Trigger message, 313
 - Tunneling, 330, 334
 - Two-way channels, 154
 - Two-way messages, 154
- U**
- UDDI (Universal Description, Discovery and Integration), 379
 - UML (Unified Modeling Language)
 - activity diagrams, 21–22
 - Unidirectional adapters, 130
 - Unidirectional channels, 100
 - Universal storage mechanism, 44
 - Update messages, 153
 - updateConsumer method, 218
 - Update() method, 151, 207–209, 213–214
 - updateNoState() method, 218–219
 - Updating files, 45
 - User interface adapters, 129
 - User interfaces, 129
- V**
- Validated Order message, 26
 - Verify Customer Standing message, 27
 - Visual transformation tools, 93–94
 - Void replies, 156
- W**
- Wait for all strategy, 272
 - Web services, 3
 - adapters, 132
 - Apache AXIS toolkit, 371
 - architecture usage scenarios, 174–175
 - asynchronous versus synchronous messaging, 373–374
 - discovery, 379

- encoding style, 374
 - existing standards, 4
 - HTTP, 51
 - loan broker system (Java) design
 - considerations, 372–376
 - reliability, 375–376
 - Remote Procedure Invocation pattern, 375
 - Request-Response Message Exchange pattern, 162, 168–169
 - security, 375–376
 - SOAP (Simple Object Access Protocol), 372–373
 - synchronous implementation of loan broker system, 371–400
 - transport protocol, 373
 - Web Services Gateway, 132
 - WebSphere Application Server, 132
 - WebSphere MQ for Java
 - Guaranteed Delivery, 126
 - messaging systems, 64–65
 - persistent channels, 126
 - queues, 65
 - WGRUS (Widgets & Gadgets 'R Us), 17
 - announcements, 17, 33–34
 - changing addresses, 17, 30–32
 - channels to interact with customers, 18
 - checking order status, 17
 - checking status, 26–29
 - internal systems, 18
 - inventory systems, 22–23
 - processing orders, 17, 20–25
 - requirements, 17
 - SOAs (service-oriented architectures), 8
 - taking orders, 17, 18–20
 - testing and monitoring, 17, 34–36
 - updating catalog, 17, 32–33
 - Wire Tap pattern, 547–550
 - World Wide Web Consortium Web site, 373, 374
 - Wrapping and unwrapping data in envelope, 331–335
 - WSDO (Web Services Deployment Descriptor), 378
 - WSDL (Web Services Definition Language), 374
 - canonical data models, 359–360
 - Command Message pattern, 146
 - document messages, 149–150
 - WSFL (Web Services Flow Language), 318
 - WS-Routing (Web Services Routing Protocol), 310–311
- X**
- XLANG, 318, 634
 - XML, 3, 149, 182
 - XML documents, 68, 90–93
 - XML files, 44
 - XML schema, 374–375
 - XML Schema Definition Tool, 415
 - XML Web services, 371–400
 - XML Splitter class, 263–264
 - XmlMessageFormatter class, 201
 - XSL, 3, 90–93
 - XSLT (XSL Transformation) language, 90
 - XslTransform class, 265