Middleware

Petr Tůma

Middleware

by Petr Tůma

Table of Contents

Chapter 1. Introduction

Middleware Definition (Klingenstein)

The term "middleware" is defined by one's point of view. Many interesting categorizations exist, all centered around sets of tools and data that help applications use networked resources and services. Some tools, such as authentication and directories, are in all definitions of middleware. Other services, such as coscheduling of networked resources, secure multicast, object brokering, and messaging, are the particular interests of certain communities, such as scientific researchers or business systems vendors. This breadth of meaning is reflected in the following working definition: Middleware is "the intersection of the stuff that network engineers don't want to do with the stuff that applications developers don't want to do."

—Klingenstein K. J.: Middleware: The Second Level of IT Infractructure.

Middleware Definition (Bray)

Middleware is connectivity software that consists of a set of enabling services that allow multiple processes running on one or more machines to interact across a network. Middleware is essential to migrating mainframe applications to client-server applications and to providing for communication across heterogeneous platforms. This technology has evolved during the 1990s to provide for interoperability in support of the move to client-server architectures. The most widely-publicized middleware initiatives are the Open Software Foundation's Distributed Computing Environment (DCE), Object Management Group's Common Object Request Broker Architecture (CORBA), and Microsoft's COM/DCOM (COM, DCOM).

—Bray M.: Middleware.

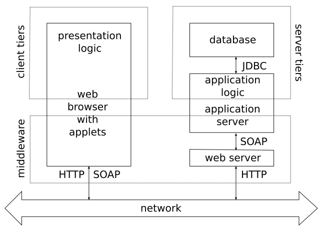
Middleware Definition (Coulson)

The role of middleware is to ease the task of designing, programming and managing distributed applications by providing a simple, consistent and integrated distributed programming environment. Essentially, middleware is a distributed software layer, or platform, which abstracts over the complexity and heterogeneity of the underlying distributed environment with its multitude of network technologies, machine architectures, operating systems and programming languages.

—Coulson G.: Middleware.

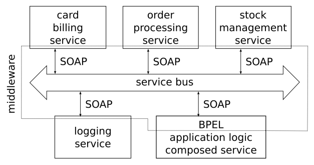
Tiered Architectures

Tiered Architecture Example



Service-Oriented Architectures

Service-Oriented Architecture Example



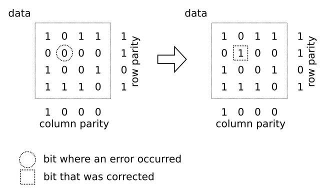
Chapter 2. Communication

Reliability In Unicast

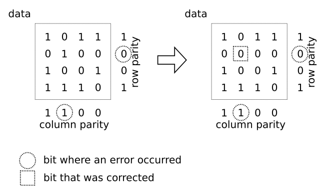
Packet Damage

Detection Using Cross Parity

Cross Parity Correction Example



Cross Parity Miscorrection Example



Timing Guarantees

Example: Real Time Protocol

RTP Packet Structure

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|V=2|P|X|  CC   |M|     PT      |       sequence number         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           timestamp                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           synchronization source (SSRC) identifier            |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
|            contributing source (CSRC) identifiers             |
|                             ....                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

- V - version
- P - padding present
- X - extension header present
- CC - number of CSRC identifiers present
- M - marker for significant packets
- PT - payload type

RTCP Packet Structure

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|V=2|P|    RC   |   PT=SR=200   |             length            | header
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         SSRC of sender                        |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
|              NTP timestamp, most significant word             | sender
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ info
|             NTP timestamp, least significant word             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         RTP timestamp                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     sender’s packet count                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      sender’s octet count                     |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
|                 SSRC_1 (SSRC of first source)                 | report
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ block
| fraction lost |       cumulative number of packets lost       |   1
-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           extended highest sequence number received           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      interarrival jitter                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         last SR (LSR)                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   delay since last SR (DLSR)                  |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
|                 SSRC_2 (SSRC of second source)                | report
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ block
:                               ...                             :   2
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
|                  profile-specific extensions                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

- RC - number of report blocks present
- PT - packet type is sender report
- NTP and RTP timestamps - make it possible to relate RTP virtual time to NTP real time and thus better estimate temporal properties of data stream

Ordering Guarantees

Causal Relation

Causal Relation Definition

Event A causally precedes event B if

- A and B are in the same process and A was executed before B,
- A is sending message M and B is receiving message M,
- there exists C such that A causally precedes C and C causally precedes B.

When A causally precedes B, we write $A \prec B$.

Lamport Clock

Lamport Clock Definition

LC timestamp is an integer local to every process

- LC(P) incremented after every step of P
- LC(P) added to M every time P sends M
- LC(P) adjusted to max(LC(M),LC(P)) every time P receives M

$A \prec B \implies LC(A) < LC(B)$

Vector Clock

Vector Clock Definition

VC timestamp is an integer vector local to every process

- VC(Pi)[i] incremented after every step of Pi
- VC(P) added to M every time P sends M
- $\forall$i VC(P)[i] adjusted to max(VC(M)[i],VC(P)[i]) every time P receives M

$VC(A) < VC(B) \equiv \forall i\ VC(A)[i] \leq VC(B)[i] \land \exists j\ VC(A)[j] < VC(B)[j]$

$VC(A) > VC(B) \equiv \forall i\ VC(A)[i] \geq VC(B)[i] \land \exists j\ VC(A)[j] > VC(B)[j]$

$A \prec B$ <u>E365</u> $VC(A) < VC(B)$

$A \succ B$ <u>E365</u> $VC(A) > VC(B)$

Remote Procedure Call

Client Side Stub Sketch

```
SomeReturnType SomeServerFunction (SomeParameterTypes parameters)
{
  // Create a message that describes the call.
  Message callRequest = new Message ();
  callRequest.put ("SomeServerFunction");
  callRequest.putSomeParameterTypes (parameters);

  // Send the request to the server.
  callRequest.send (ServerAddress);

  // Wait for server reponse.
  Message callResponse = Message.receive (ServerAddress);

  // Extract return value from server response.
  SomeReturnType result = callResponse.getSomeReturnType ();
  return (result);
}
```

- What is the function signature ?
- How is the server function identified ?
- How are the parameters encoded in the message ?
- Where does the server address come from ?
- What if the communication fails ?

Server Side Stub Sketch

```
void ServerMainLoop ()
{
  // Keep responding to messages for ever.
  while (true)
  {
    // Wait for client request.
    Message callRequest = Message.receive ();

    // Figure out what function is called.
    String function = callRequest.getString ();

    // The rest of message processing depends on function.
    if (function.equals ("SomeServerFunction")
    {
      SomeParameterTypes parameters = callRequest.getSomeParameterTypes ();
      SomeReturnType result = SomeServerFunction (parameters);

      // Create a message that describes the result.
      Message callResponse = new Message ();
      callResponse.putSomeReturnType (result);

      // Send the response to the client.
      callResponse.send (ClientAddress);
    }
    else ...
  }
}
```

- Where does the function implementation come from ?
- Where does the client address come from ?

- What if there are multiple clients ?
- What if the function fails ?

Chapter 3. Systems

GM

gm_send_with_callback Function

```
void gm_send_with_callback (
  struct gm_port *p,
  void *message,
  unsigned int size,
  gm_size_t len,
  unsigned int priority,
  unsigned int target_node_id,
  unsigned int target_port_id,
  gm_send_completion_callback_t callback,
  void *context
);
```

- size - size of buffer to use on receiver side
- len - size of message

gm_receive And gm_unknown Functions

```
gm_recv_event_t *gm_receive (
  gm_port_t *p);

void gm_unknown (
  gm_port_t *p,
  gm_recv_event_t *e
);
```

gm_provide_receive_buffer_with_tag Function

```
void gm_provide_receive_buffer_with_tag (
  gm_port_t *p,
  void *ptr,
  unsigned size,
  unsigned priority,
  unsigned int tag
);
```

- tag - tag to add to receive event

IBM MQ

Queues And Messages

MQCONN And MQDISC Functions

```
MQCONN (in MQCHAR48 QMgrName,
        out MQHCONN Hconn,
        out MQLONG CompCode,
        out MQLONG Reason);
MQDISC (inout MQHCONN Hconn,
        out MQLONG CompCode,
        out MQLONG Reason);
```

- QMgrName - string name of queue manager from configuration
- Hconn - handle of opened connection to queue manager
- CompCode - completion code
- Reason - failure reason for completion code other than MQCC_OK

MQOPEN And MQCLOSE Functions

```
MQOPEN (in MQHCONN Hconn,
        inout MQOD ObjDesc,
        in MQLONG Options,
        out MQHOBJ Hobj,
        out MQLONG CompCode,
        out MQLONG Reason);
MQCLOSE (in MQHCONN Hconn,
         inout MQHOBJ Hobj,
         in MQLONG Options,
         out MQLONG CompCode,
         out MQLONG Reason);
```

- Hconn - handle of opened connection to queue manager
- ObjDesc - descriptor of queue with string name of queue
- Hobj - handle of opened queue object
- CompCode - completion code
- Reason - failure reason for completion code other than MQCC_OK

MQPUT And MQGET Functions

```
MQPUT (in MQHCONN Hconn,
       in MQHOBJ Hobj,
       inout MQMD MsgDesc,
       inout MQPMO PutMsgOpts,
       in MQLONG BufferLength,
       in MQBYTExBufferLength Buffer,
       out MQLONG CompCode,
       out MQLONG Reason);
MQGET (in MQHCONN Hconn,
       in MQHOBJ Hobj,
       inout MQMD MsgDesc,
       inout MQGMO GetMsgOpts,
       in MQLONG BufferLength,
       out MQBYTExBufferLength Buffer,
       out MQLONG DataLength,
       out MQLONG CompCode,
       out MQLONG Reason);
```

Message Descriptor

- StrucId - magic number identifying the descriptor
- Version - number identifying the descriptor version
- Report - what report messages are required
- MsgType - message type, either one of predefined system types or application type
- Expiry - how long until the message can be discarded
- Feedback - status inside report message
- Encoding - what numeric encoding is used
- CodedCharSetId - what textual encoding is used
- Format - name of message data format
- Priority - message priority
- Persistence - message persistence options
- MsgId - unique message identifier
- CorrelId - unique message identifier of related message
- BackoutCount - counter of failed attempts at message processing
- ReplyToQ - where to send a reply to
- ReplyToQMgr - where to send a reply to
- UserIdentifier
- AccountingToken
- ApplIdentityData - additional identity data
- PutApplType
- PutApplName
- PutDate
- PutTime
- ApplOriginData - additional origin data
- GroupId - logical message group identifier
- MsgSeqNumber - sequence number of message in logical message group
- Offset - offset of physical message in logical message
- MsgFlags - flags controlling segmentation into logical message group
- OriginalLength - length of logical message for physical message

Message Encoding

Data Conversion Exit Definition Example

```
struct TEST
{
  MQLONG SERIAL_NUMBER;
  MQCHAR ID[5];
  MQSHORT VERSION;
  MQBYTE CODE[4];
  MQLONG DIMENSIONS[3];
  MQCHAR NAME[24];
};
```

Web Services

SOAP

Message Example

```
<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/"
  SOAP:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">

  <!-- Header with additional information -->
  <SOAP:Header>
    <wscoor:CoordinationContext
      xmlns:wscoor="http://schemas.xmlsoap.org/ws/2003/09/wscoor"
      SOAP:mustUnderstand="true">
      <wscoor:Identifier>
        http://example.com/context/1234
      </wscoor:Identifier>
    </wscoor:CoordinationContext>
  </SOAP:Header>

  <!-- Body with message content -->
  <SOAP:Body>
    <m:aMethodRequest xmlns:m="http://example.com/soap.wsdl">
      <aNumber xsi:type="xsd:int">42</aNumber>
    </m:aMethodRequest>
  </SOAP:Body>

</SOAP:Envelope>
```

WSDL

Service Example

```
<?xml version="1.0"?>
<definitions name="StockQuote"
             targetNamespace="http://example.com/stockquote.wsdl"
             xmlns:tns="http://example.com/stockquote.wsdl"
             xmlns:xsd="http://example.com/stockquote.xsd"
             xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
             xmlns="http://schemas.xmlsoap.org/wsdl/">

  <!-- Types used in communication -->
  <types>
    <schema targetNamespace="http://example.com/stockquote.xsd"
            xmlns="http://www.w3.org/2000/10/XMLSchema">
      <element name="TradePriceRequest">
        <complexType>
          <all>
            <element name="tickerSymbol" type="string"/>
          </all>
        </complexType>
      </element>
      <element name="TradePriceReply">
         <complexType>
           <all>
             <element name="price" type="float"/>
           </all>
         </complexType>
      </element>
    </schema>
  </types>

  <!-- Messages exchanged in communication -->
  <message name="GetLastTradePriceInput">
    <part name="body" element="xsd:TradePriceRequest"/>
  </message>
  <message name="GetLastTradePriceOutput">
    <part name="body" element="xsd:TradePriceReply"/>
  </message>

  <!-- Ports available in communication -->
  <portType name="StockQuotePortType">
    <operation name="GetLastTradePrice">
      <input message="tns:GetLastTradePriceInput"/>
      <output message="tns:GetLastTradePriceOutput"/>
    </operation>
  </portType>

  <!-- Bindings used in communication -->
  <binding name="StockQuoteSoapBinding" type="tns:StockQuotePortType">
    <soap:binding
      style="document"
      transport="http://schemas.xmlsoap.org/soap/http"/>
    <operation name="GetLastTradePrice">
      <soap:operation soapAction="http://example.com/GetLastTradePrice"/>
      <input><soap:body use="literal"/></input>
      <output><soap:body use="literal"/></output>
    </operation>
  </binding>

  <!-- Service -->
  <service name="StockQuoteService">
    <documentation>Stock quoter service.</documentation>
    <port name="StockQuotePort" binding="tns:StockQuoteSoapBinding">
      <soap:address location="http://example.com/stockquote"/>
    </port>
  </service>

</definitions>

<!-- Example adjusted from WSDL 1.1 Specification. -->
```

Service Composition

BPEL Example

```
<process name="anExampleProcess">

  <!-- Partners of the example process -->
  <partnerLinks>
    <partnerLink name="client"
      partnerLinkType="aClientPort"
      myRole="aProviderRole"/>
    <partnerLink name="serverOne"
      partnerLinkType="aServerPort"
      myRole="aClientRole"
      partnerRole="aServerRole"/>
    <partnerLink name="serverTwo"
      partnerLinkType="aServerPort"
      myRole="aClientRole"
      partnerRole="aServerRole"/>
  </partnerLinks>

  <!-- Internal variables -->
  <variables>
    <variable name="ClientRequest" messageType="RequestMessage"/>
    <variable name="ServerOneResponse" messageType="ResponseMessage"/>
    <variable name="ServerTwoResponse" messageType="ResponseMessage"/>
    <variable name="ProviderResponse" messageType="ResponseMessage"/>
  </variables>

  <!-- Process definition -->
  <sequence>
    <!-- Get the request from the client -->
    <receive partnerLink="client"
      portType="aClientPort"
      operation="GetOffer"
      variable="ClientRequest"
      createInstance="yes"/>

    <!-- Forward the request to both servers -->
    <flow>
      <invoke partnerLink="serverOne"
        portType="aServerPort"
        operation="GetOffer"
        inputVariable="ClientRequest"
        outputVariable="ServerOneResponse"
      />
      <invoke partnerLink="serverTwo"
        ...
      />
    </flow>

    <!-- Create response from cheapest offer -->
    <switch>
      <case condition="bpws:getVariableData ('ServerOneResponse','price')
                       <
                       bpws:getVariableData ('ServerTwoResponse','price')">
        <assign>
          <copy>
            <from variable="ServerOneResponse"/>
            <to variable="ProviderResponse"/>
          </copy>
        </assign>
      </case>
      <otherwise>
        ...
      </otherwise>
    </switch>

    <!-- Return the response to the client -->
    <reply partnerLink="client"
      portType="aClientPort"
      operation="GetOffer"
      variable="ProviderResponse"/>
  </sequence>
</process>
```

DCE

Remote Procedure Call

Interface Definition Example

```
[uuid (12345678-9ABC-DEF0-1234-56789ABCDEF0), version (1.0)]
interface Foo
{
  [context_handle] void *Bar ([in] long X, [out] long *Y);
  [idempotent] void Rab ([in, ref, string] char *A);
};
```

EJB

Stateful Session Beans

Session Bean Home Interface Example (EJB 2)

```
public interface ASessionBeanHome extends javax.ejb.EJBHome
{
  public ASessionBean createOneWay (int iArgument)
    throws RemoteException, CreateException;
  public ASessionBean createAnotherWay (int iArgument, String sArgument)
    throws RemoteException, CreateException;
}

public interface EJBHome extends Remote
{
  public void remove (Handle handle) throws RemoteException, RemoveException;
  ...
}
```

Session Bean Remote Interface Example (EJB 2)

```
public interface ARemoteInterface extends javax.ejb.EJBObject
{
  public void myMethodOne (int iArgument) throws RemoteException { ... }
  public int myMethodTwo (Object oArgument) throws RemoteException { ... }
}
```

Session Bean Class Example (EJB 2)

```
public class ASessionBean implements javax.ejb.SessionBean
{
  // Method that provides reference to standard session context object
  public void setSessionContext (SessionContext sessionContext) { ... };

  // Method that is called after construction
  public void ejbCreateOneWay (int iArgument)
  throws RemoteException, CreateException
  { ... }
  public void ejbCreateAnotherWay (int iArgument, String sArgument)
  throws RemoteException, BadAccountException, CreateException
  { ... }

  // Method that is called before destruction
  public void ejbRemove () { ... }

  // Methods that are called after activation and before passivation
  public void ejbActivate () { ... }
  public void ejbPassivate () { ... };

  // Some business methods ...
  public void myMethodOne (int iArgument) { ... }
  public int myMethodTwo (Object oArgument) { ... }
}
```

Session Bean Class Example (EJB 3)

```
@Stateful public class ASessionBean implements ABusinessInterface
{
  // Injected reference to standard session context object
  @Resource public SessionContext sessionContext;

  // Method that is called after construction or activation
  @PostConstruct @PostActivate
  public void myInitMethod () { ... }

  // Method that is called before passivation or destruction
  @PreDestroy @PrePassivate
  public void myDoneMethod () { ... }

  // Some business methods ...
  public void myMethodOne (int iArgument) { ... }
  public int myMethodTwo (Object oArgument) { ... }

  // Business method that removes the bean instance
  @Remove public void myRemovalMethod () { ... }

  // Interceptor method that can also be in separate interceptor class
  @AroundInvoke
  public Object myInterceptor (InvocationContext inv)
  throws Exception
  {
    ...
    Object result = inv.proceed ();
    ...
    return (result);
  }
}
```

Stateless Session Beans

Obtaining Session Bean Interface

```
// Business interface dependency injection
// Instance per session
@EJB Cart cart;

// Business interface naming service lookup
// Instance per session
@Resource SessionContext ctx;
Cart cart = (Cart) ctx.lookup ("cart");

// Home interface dependency injection
// Instance created explicitly
@EJB CartHome cartHome;
Cart cart = cartHome.createLargeCart (...);

// Home interface naming service lookup
// Instance created explicitly
@Resource SessionContext ctx;
CartHome cartHome = (CartHome) ctx.lookup ("cartHome");
Cart cart = cartHome.createLargeCart (...);
```

Message Driven Beans

Obtaining Message Bean Interface

```
// Destination dependency injection
@Resource Queue stockInfoQueue;

// Destination naming service lookup
Context initialContext = new InitialContext ();
Queue stockInfoQueue =
  (javax.jms.Queue) initialContext.lookup
    ("java:comp/env/jms/stockInfoQueue");
```

Entities

Entity Bean Home Interface Example (EJB 2)

```
public interface AccountHome extends javax.ejb.EJBHome
{
  public Account create (String firstName, String lastName, double initialBalance)
    throws RemoteException, CreateException;
  public Account create (String accountNumber, double initialBalance)
    throws RemoteException, CreateException, LowInitialBalanceException;
  public Account createLargeAccount (String firstname, String lastname, double initialB
    throws RemoteException, CreateException;
  ...
  public Account findByPrimaryKey (String AccountNumber)
    throws RemoteException, FinderException;
  ...
}

public interface EJBHome extends Remote
```

```
{
  public void remove (Object primaryKey) throws RemoteException, RemoveException;
}
```

Field Based Entity Bean Class Example (EJB 3)

```
@Entity public class AnEntity
{
  // With field based access fields are persistent by default.
  private int someField;
  private String someOtherField;

  // Relationships among entities must be annotated.
  @OneToMany private Collection<AnotherEntity> relatedEntities;

  // Every entity must have a primary key.
  @Id private long aKeyField;

  // Field that is not persistent
  @Transient private String aTransientString;

  // Obligatory constructor with no arguments
  public AnEntity () { ... }

  // Additional business methods ...
  public void myMethodOne (int iArgument) { ... }
  public int myMethodTwo (Object oArgument) { ... }
}
```

Property Based Entity Bean Class Example (EJB 3)

```
@Entity public class AnEntity
{
  // With property based access fields are not persistent themselves.
  private int someTransientField;
  private String someOtherTransientField;

  // Relationships among entities must be annotated.
  private Collection<AnotherEntity> relatedEntities;
  @OneToMany public Collection<AnotherEntity> getRelatedEntities ()
  {
    return (relatedEntities);
  }
  public void setRelatedEntities (Collection<AnotherEntity> entityCollection)
  {
    relatedEntities = entityCollection;
  }

  // Getter and setter methods for primary key.
  private long aKeyField;
  @Id Long getAKeyField () { return (aKeyField); }
  public void setAKeyField (Long aKeyField) { this.aKeyField = aKeyField; }

  // Obligatory constructor with no arguments
  public AnEntity () { ... }

  // Additional business methods ...
  public void myMethodOne (int iArgument) { ... }
  public int myMethodTwo (Object oArgument) { ... }
}
```

Obtaining Entity Bean Interface

```
// Home interface naming service lookup
Context initialContext = new InitialContext ();
AccountHome accountHome =
  (AccountHome) initialContext.lookup
    ("java:comp/env/ejb/accounts");

// Creation
accountHome.createLargeAccount (...);

// Location
accountHome.findByPrimaryKey (...);
```

Entity Manager Interface

```
public interface EntityManager
{
  void persist (Object entity);
  void refresh (Object entity);
  void remove (Object entity);

  void detach (Object entity);
  <T> T merge (T entity);

  void lock (Object entity, LockModeType lockMode);

  // Find by primary key
  <T> T find (Class<T> entityClass, Object primaryKey);

  // Find by primary key and return lazy reference
  <T> T getReference (Class<T> entityClass, Object primaryKey);

  // Clear persistence context and detach all entities
  void clear ();

  // Check whether persistence context contains managed entity
  boolean contains (Object entity);

  // Synchronize persistence context with database
  // Flush mode governs automatic synchronization
  // upon query execution or upon commit
  void flush ();
  FlushModeType getFlushMode ();
  void setFlushMode (FlushModeType flushMode);

  Query createQuery (String ejbqlString);
  Query createNamedQuery (String name);
  Query createNativeQuery (String sqlString);
  ...
}
```

Query Interface

```
public interface Query
{
  // Execute a query that returns a result list
  List getResultList ();
  // Execute a query that returns a single result
  Object getSingleResult();
  // Execute an update query
  int executeUpdate ();
  // Methods used to fetch results step by step
  Query setMaxResults (int maxResult);
  Query setFirstResult (int startPosition);

  // Bind a parameter in a query
  Query setParameter (String name, Object value);
  Query setParameter (String name, Date value, TemporalType temporalType);
  Query setParameter (String name, Calendar value, TemporalType temporalType);
  Query setParameter (int position, Object value);
  Query setParameter (int position, Date value, TemporalType temporalType);
  Query setParameter (int position, Calendar value, TemporalType temporalType);
}
```

Transactions

Transaction Attributes

- not supported
- required
- supports
- requires new
- mandatory
- never

Specifying Transaction Attributes

```
@TransactionAttribute (NOT_SUPPORTED)
@Stateful
public class MyBean implements MyBeanInterface
{
  @TransactionAttribute (REQUIRES_NEW)
  public void methodOne () {...}

  @TransactionAttribute (REQUIRED)
  public void methodTwo () {...}

  public void methodThree () {...}

  public void methodFour () {...}
}
```

JMS

Connections and Sessions and Contexts

Connection Creation Example

```
// Get an initial naming context
```

```
Context initialContext = new InitialContext ();

// Look up the connection factory using
// a well known name in the initial context
ConnectionFactory connectionFactory;
connectionFactory = (ConnectionFactory) initialContext.lookup ("ConnectionFactory");

// Create a connection using the factory
Connection connection;
connection = ConnectionFactory.createConnection ();

// A connection only delivers messages
// once it is explicitly started
connection.start ();
```

Session Creation Example

```
// Create a session for a connection, requesting
// no transaction support and automatic message
// acknowledgement
Session session;
session = connection.createSession (false, Session.AUTO_ACKNOWLEDGE);
```

Context Creation Example

```
// Create a context that includes a connection and a session.
// Use try with resources to close the context when done.
try (JMSContext context = connectionFactory.createContext ();)
{
  // Create another context reusing the same connection.
  try (JMSContext another = context.createContext ();)
  {
    ...
  } catch (JMSRuntimeException ex) { ... }
} catch (JMSRuntimeException ex) { ... }
```

Destinations

Destination Creation Example

```
Queue oQueue = oSession.createQueue ("SomeQueueName");
Topic oTopic = oSession.createTopic ("SomeTopicName");

Queue oTemporaryQueue = oSession.createTemporaryQueue ();
Topic oTemporaryTopic = oSession.createTemporaryTopic ();
```

Messages

Message Header

Fields filled in by sender.

- JMSCorrelationID - correlated message identifier

- JMSReplyTo - suggested reply destination
- JMSType - message type understood by recipient

Fields controlled by sender.

- JMSDestination - message recipient
- JMSExpiration - message lifetime
- JMSPriority - message priority
- JMSDeliveryMode - PERSISTENT or NON_PERSISTENT
- JMSDeliveryTime - earliest message delivery time

Fields filled in by middleware.

- JMSMessageID - unique message identifier
- JMSTimestamp - message timestamp
- JMSRedelivered - repeated delivery indication

Message Body Types

- StreamMessage - stream of primitive types
- MapMessage - set of named values
- TextMessage - java.lang.String
- ObjectMessage - serializable object
- BytesMessage - byte array

Producers and Consumers

Producer And Consumer Creation Example

```
// Uses the classic API.

MessageProducer sender;
MessageConsumer recipient;

sender = session.createProducer (oQueue);
recipient = session.createConsumer (oQueue);
```

Producer And Consumer Creation Example

```
// Uses the simplified API.

// Configure sender with method chaining.
// Sender is not bound to destination here.
JMSProducer sender = context.createProducer ().
                     setDeliveryMode (PERSISTENT).
                     setDeliveryDelay (1000).
                     setTimeToLive (10000);

JMSConsumer recipient = context.createConsumer (oQueue);
```

Synchronous Message Send Example

```
// Uses the classic API.

TextMessage message;

message = session.createTextMessage ();
message.setText ("Hello");

// Always blocks until message is sent.
sender.send (message);
```

Synchronous Message Send Example

```
// Uses the simplified API.

// By default blocks until message is sent.
// Overloaded versions for all body types exist.
sender.send (oQueue, "Hello");
```

Message Receive Example

```
// Uses the classic API.

TextMessage oMessage;

oMessage = (TextMessage) recipient.receive ();
oMessage = (TextMessage) recipient.receive (1000);
```

Message Listener Example

```
// Uses the classic API.

public class SomeListener implements MessageListener
{
  public void onMessage (Message message)
  { ... }
}

SomeListener oListener = new SomeListener ();
recipient.setMessageListener (oListener);
```

Message Receive Example

```
// Uses the simplified API.

// Template versions for all body types exist.
String body = consumer.receiveBody (String.class);
```

Message Filter Example

```
String selector;
MessageConsumer receiver;

selector = new String ("(SomeProperty = 1000)");
receiver = session.createConsumer (oQueue, selector);
```

Durable Subscriber Example

```
session.createDurableSubscriber (oTopic,"DurableSubscriberName");
```

Shared Subscriber Example

```
MessageConsumer consumer;

consumer = session.createSharedConsumer (oQueue, "SharedSubscriberName");
```

MPI

Peer To Peer Communication

MPI_Send Function

```
int MPI_Send (void *buf,
              int count,
              MPI_Datatype datatype,
              int dest,
              int tag,
              MPI_Comm comm);
```

- buf - address of send buffer
- count - number of elements in send buffer
- datatype - datatype of each send buffer element
- dest - rank of destination
- tag - message tag
- comm - communicator

MPI_Recv Function

```
int MPI_Recv (void *buf,
              int count,
              MPI_Datatype datatype,
              int source,
              int tag,
              MPI_Comm comm,
              MPI_Status *status);
```

- buf - address of receive buffer
- count - maximum number of elements in receive buffer
- datatype - datatype of each receive buffer element
- source - rank of source
- tag - message tag
- comm - communicator
- status - status object

Synchronization And Blocking Modes

Synchronization

- Send - may block, buffer available on return, asynchronous
- BSend - may block, uses supplied buffers, buffer available on return, asynchronous
- SSend - may block, synchronous
- RSend - may block, target must be receiving

Blocking

- ISend, IRecv - do not block, return request handle
- Wait - block on request handle
- Test - check request status
- Cancel - cancel blocking wait
- - plus B, S and R versions
- - plus WaitAny, WaitAll, WaitSome
- - plus TestAny, TestAll, TestSome
- Probe, IProbe - block but do not receive

Group Communication

Group Communication Primitives

- Bcast - odesílatel A, příjemci A ...
- Gather - odesílatelé A, B, C, příjemce ABC
- Scatter - odesílatel ABC, příjemci A, B, C
- Allgather - odesílatelé A, B, C, příjemci ABC ...
- Alltoall - odesílatelé ABC, DEF, GHI, příjemci ADG ...
- Reduce - odesílatelé A, B, C, příjemce A+B+C
- Allreduce - odesílatelé A, B, C, příjemci A+B+C ...
- Reduce_scatter - odesílatelé ABC, DEF, GHI, příjemci A+D+G ...
- Scan - odesílatelé A, B, C, příjemci A, A+B, A+B+C
- Barrier - rendez vous

Remote Memory Access

MPI_Win_Create Function

```
int MPI_Win_Create (void *base,
                    int size,
                    int disp_unit,
                    MPI_Info info,
                    MPI_Comm comm,
                    MPI_Win *win);
```

- disp_unit - unit size used in scaling remote offsets
- info - optimization hints

MPI_Win_Put Function

```
int MPI_Win_Put (void *origin_addr,
                 int origin_count,
                 MPI_Datatype origin_datatype,
                 int target_rank,
                 int target_disp,
                 int target_count,
                 MPI_Datatype target_datatype,
                 MPI_Win *win);
```

.NET Remoting

Interface

Remotely Accessible Object Example

```
public class Example : MarshalByRefObject
{
  public void printMessage (string message)
  {
    System.Console.WriteLine (message);
  }
}
```

- Inheritance used to request passing by reference.
- Inheritance used to get some utility methods.
- Serializable arguments.

Implementation

Server Code Example

```
TcpChannel channel = new TcpChannel (12345);
ChannelServices.RegisterChannel (channel);

RemotingConfiguration.RegisterWellKnownServiceType (
  typeof (Example), "Example",
  WellKnownObjectMode.Singleton);
```

- Transport channel instantiated explicitly.
- Remote object type registered under well known name.
- Lifecycle managed using predefined pattern.

Client Code Example

```
TcpChannel channel = new TcpChannel ();
ChannelServices.RegisterChannel (channel);

Example obj = (Example) Activator.GetObject (
  typeof (Example), "tcp://localhost:12345/Example");

obj.printMessage ("Hello World !");
```

- Transport channel instantiated explicitly.
- Remote object located using well known name.
- Lifecycle managed using predefined pattern.

Java RMI

Interface

Remotely Accessible Type Example

```
public interface Example extends Remote
{
    void printMessage (String message) throws RemoteException;
}
```

- Inheritance used to request passing by reference.
- Serializable arguments.
- Remote exception.

Implementation

Remotely Accessible Object Example

```
public class ExampleImpl
  extends UnicastRemoteObject
  implements Example
{
  public ExampleImpl () throws RemoteException { }
  public void printMessage (String message) { System.out.println (message); }
}
```

- Interface used to mark remotely accessible object.
- Inheritance used to export the instance.
- Constructor can return exception.

exportObject Methods

```
static Remote exportObject (Remote obj, int port)
static Remote exportObject (Remote obj, int port,
                            RMIClientSocketFactory csf,
                            RMIServerSocketFactory ssf)
static boolean unexportObject (Remote obj, boolean force)
```

Lifecycle

Unreferenced Interface

```
public interface Unreferenced
{
  void unreferenced ();
}
```

Called some time after no client holds reference to remote object.

Naming

Naming Interface

```
class java.rmi.Naming
{
  static void bind (String name, Remote obj);
  static void rebind (String name, Remote obj);
  static void unbind (String name);
  static Remote lookup (String name);
  static String [] list (String name);
}
```

Naming Example

```
// Server side registration.
ExampleImpl obj = new ExampleImpl ();
Naming.rebind ("//localhost/Example", obj);

// Client side lookup.
Example obj = (Example) Naming.lookup ("//localhost/Example");
obj.printMessage ("Hello World !");
```

Rehearsal

Remote Object Rehearsal

Remote objects are transparently represented by proxies.

- How are remote objects compared for equality ?
- Can more proxies represent one remote object ?
- Can proxies exist on the server ?
- When does any of this matter ?

Naming Rehearsal

Naming registers object references under string names and looks up the references using the names.

- What is the scope of the names, can names collide ?
- Is the **rmiregistry** server a common server ?

Sun RPC

Interface Definition Example

```
const MNTPATHLEN = 1024; /* maximum bytes in a pathname argument */
const MNTNAMLEN = 255;  /* maximum bytes in a name argument */
const FHSIZE = 32;  /* size in bytes of a file handle */

typedef opaque fhandle [FHSIZE];
typedef string name <MNTNAMLEN>;
typedef string dirpath <MNTPATHLEN>;

union fhstatus switch (unsigned fhs_status) {
  case 0:
    fhandle fhs_fhandle;
  default:
    void;
};

typedef struct mountbody *mountlist;
struct mountbody {
  name ml_hostname;
  dirpath ml_directory;
  mountlist ml_next;
};

typedef struct groupnode *groups;
struct groupnode {
  name gr_name;
  groups gr_next;
};

typedef struct exportnode *exports;
struct exportnode {
  dirpath ex_dir;
  groups ex_groups;
  exports ex_next;
};

program MOUNTPROG {
  version MOUNTVERS {
    void  MOUNTPROC_NULL (void) = 0;
    fhstatus  MOUNTPROC_MNT (dirpath) = 1;
    mountlist  MOUNTPROC_DUMP (void) = 2;
    void MOUNTPROC_UMNT (dirpath) = 3;
    void MOUNTPROC_UMNTALL (void) = 4;
    exports MOUNTPROC_EXPORT (void) = 5;
    exports MOUNTPROC_EXPORTALL (void) = 6;
  } = 1;
} = 100005;
```

Portmapper Services Example

```
> rpcinfo -p
   program vers proto   port
    100000    2   tcp    111  portmapper
    100000    2   udp    111  portmapper
    100011    1   udp    892  rquotad
    100011    2   udp    892  rquotad
    100011    1   tcp    895  rquotad
    100011    2   tcp    895  rquotad
    100003    2   udp   2049  nfs
    100003    3   udp   2049  nfs
    100021    1   udp  39968  nlockmgr
    100021    3   udp  39968  nlockmgr
    100021    4   udp  39968  nlockmgr
    100005    1   udp  39969  mountd
    100005    1   tcp  45529  mountd
    100005    2   udp  39969  mountd
    100005    2   tcp  45529  mountd
    100005    3   udp  39969  mountd
    100005    3   tcp  45529  mountd
    100024    1   udp  39970  status
    100024    1   tcp  45530  status
    391002    2   tcp  45533  sgi_fam
```

DCOM

Interface Definition Language

MIDL Interface Definition Example

```
[object, uuid(12345678-9ABC-DEF0-1234-56789ABCDEF0), ]
interface ISomething : IUnknown
{
  typedef unsigned char BUFFER [1234];
  HRESULT MethodOne ([in] short InOne,
                     [out] long *pOutOne,
                     [in, out] BUFFER *pBuffer);
};
```

Interface And Component Attributes

MIDL Interface Attributes

- auto_handle - bind functions with no binding
- endpoint - default protocol and address
- local - local rather than remote interface
- object - COM rather than RPC interface
- uuid - universally unique identifier
- version - major and minor version number

MIDL Component Attributes

- aggregatable - supports aggregation
- appobject - complete application
- control - user interface component
- hidden - hidden component

Type Attributes

MIDL Type Attributes

- allocate - memory is allocated and freed
- context_handle - contains server side context
- decode - deserialization accessible to the programmer
- encode - serialization accessible to the programmer
- ignore - ignore pointer target when marshalling
- represent_as - present wire type as given local type
- transmit_as - transport local type as given wire type

- user_marshal - programmer supplied marshalling for local type
- wire_marshal - programmer supplied marshalling for wire type

Function Attributes

MIDL Function Attributes

- async - asynchronous client stub
- bindable - accessor function with change notification
- broadcast - call delivered to all available servers
- call_as - call complex function as given simplified function
- callback - callback within remote call context
- idempotent - same effect if executed multiple times
- immediatebind - accessor function with persistent changes
- maybe - does not need to be executed reliably
- message - call delivered as asynchronous message
- notify - server failure notification called
- propget - getter accessor function
- propput - setter accessor function
- usesgetlasterror - signals error using SetLastError and GetLastError

Argument Attributes

MIDL Argument Attributes

- in - passed from client to server
- out - passed from server to client
- optional - argument is optional
- readonly - cannot be assigned to
- partial_ignore - pointer to potentially uninitialized data
- defaultvalue - default value for optional argument
- retval - return value

- ptr - full pointer, can be NULL and have aliases
- ref - reference pointer, cannot be NULL and cannot have aliases
- unique - unique pointer, can be NULL but cannot have aliases
- force_allocate - always allocate dynamically

- byte_count - size of referenced data
- first_is - index of first array item
- last_is - index of last array item

- length_is - length of array
- switch_is - discriminant of a union

- pipe - stream between client and server
- comm_status - failure code on communication error
- fault_status - failure code on server error

Components

IUnknown

IUnknown Interface

```
interface IUnknown
{
  HRESULT QueryInterface (REFIID iid, void **ppvObject);
  ULONG AddRef (void);
  ULONG Release (void);
};
```

IDispatch

IDispatch Interface

```
interface IDispatch
{
  HRESULT GetTypeInfoCount (unsigned int FAR *pcTInfo);
  HRESULT GetTypeInfo (
    unsigned int iTInfo,
    LCID lcid,
    ITypeInfo FAR* FAR* ppTInfo);
  HRESULT GetIDsOfNames (
    REFIID riid,
    OLECHAR FAR* FAR* rgszNames,
    unsigned int cNames,
    LCID lcid,
    DISPID FAR* rgDispId);
  HRESULT Invoke (
    DISPID dispIdMember,
    REFIID riid,
    LCID lcid,
    WORD wFlags,
    DISPPARAMS FAR* pDispParams,
    VARIANT FAR* pVarResult,
    EXCEPINFO FAR* pExcepInfo,
    unsigned int FAR* puArgErr);
};
```

Lifecycle

Creating Object Instance

```
class IClassFactory : IUnknown
{
  STDMETHOD CreateInstance (IUnknown* pUnkOuter, IID &riid, void **ppv);
  STDMETHOD LockServer (BOOL fLock);
};

CoCreateInstance (
  REFCLSID rclsid,
  LPUNKNOWN pUnkOuter,
  DWORD dwClsContext,
  REFIID riid,
  LPVOID *ppv);

IClassFactory *pCF;
CoGetClassObject (rclsid, dwClsContext, NULL, IID_IClassFactory, &pCF);
hresult = pCF->CreateInstance (pUnkOuter, riid, ppvObj);
pCF->Release ();
```

JAXB

Mapping Example XML Schema Input

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="purchase" type="OrderType"/>
  <xsd:element name="comment" type="xsd:string"/>

  <xsd:complexType name="OrderType">
    <xsd:sequence>
      <xsd:element name="shipTo" type="Address"/>
      <xsd:element name="billTo" type="Address"/>
      <xsd:element ref="comment" minOccurs="0"/>
      <xsd:element name="items" type="Items"/>
    </xsd:sequence>
    <xsd:attribute name="date"type="xsd:date"/>
  </xsd:complexType>

  <xsd:complexType name="Address">
    <xsd:sequence>
      <xsd:element name="name" type="xsd:string"/>
      <xsd:element name="street" type="xsd:string"/>
      <xsd:element name="city" type="xsd:string"/>
      <xsd:element name="zip" type="xsd:decimal"/>
    </xsd:sequence>
    <xsd:attribute name="country" type="xsd:NMTOKEN" fixed="Some Country"/>
  </xsd:complexType>

  <xsd:complexType name="Items">
    <xsd:sequence>
      <xsd:element name="item" minOccurs="1" maxOccurs="unbounded">
        <xsd:complexType>
          <xsd:sequence>
            ...
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>

  ...

  <!-- Example adjusted from JAXB 2.2 Specification. -->
```

Mapping Example Java Output

```
public class OrderType {
  Address getShipTo () {...}
  void setShipTo (Address) {...}
  Address getBillTo () {...}
  void setBillTo (Address) {...}

  String getComment () {...}
  void setComment (String) {...}

  Items getItems () {...}
  void setItems (Items) {...}

  XMLGregorianCalendar getOrderDate () {...}
  void setOrderDate (XMLGregorianCalendar) {...}
};

public class Address {
  String getName () {...}
  void setName (String) {...}
  String getStreet () {...}
  void setStreet (String) {...}
  String getCity () {...}
  void setCity (String) {...}
  int getZip () {...}
  void setZip (int) {...}

  static final String COUNTRY="Some Country";
};

public class Items {
  public class ItemType {...}

  List<Items.ItemType> getItem () {...}
}

// Example adjusted from JAXB 2.2 Specification.
```

OSGi

Bundles

OSGi Bundle Activator Interface

```
interface BundleActivator {
  void start (BundleContext context) throws Exception;
  void stop (BundleContext context) throws Exception;
}
```

OSGi Bundle Context Interface (Bundle Related)

```
interface BundleContext {
  // Access to framework properties
  String getProperty (String key);

  // Access to objects representing bundles
  Bundle getBundle ();
  Bundle getBundle (long id);
  Bundle [] getBundles ();

  // Support for bundle management
  Bundle installBundle (String location, InputStream input) throws BundleException;
  Bundle installBundle (String location) throws BundleException;

  // Support for bundle lifecycle notifications
  void addBundleListener (BundleListener listener);
  void removeBundleListener (BundleListener listener);

  // Support for framework event notifications
  void addFrameworkListener (FrameworkListener listener);
  void removeFrameworkListener (FrameworkListener listener);

  // Support for persistent storage
  File getDataFile (String filename);

  ...
}
```

Services

OSGi Bundle Context Interface (Service Related)

```
interface BundleContext {

  ...

  // Support for service management
  ServiceRegistration registerService (String [] clazzes, Object service, Dictionary pr
  ServiceRegistration registerService (String clazz, Object service, Dictionary propert

  Filter createFilter (String filter) throws InvalidSyntaxException;
  ServiceReference [] getServiceReferences (String clazz, String filter) throws Invalid
  ServiceReference [] getAllServiceReferences (String clazz, String filter) throws Inva

  ServiceReference getServiceReference (String clazz);
  Object getService (ServiceReference reference);
  boolean ungetService (ServiceReference reference);

  // Support for service lifecycle notifications
  void addServiceListener (ServiceListener listener, String filter) throws InvalidSynta
  void addServiceListener (ServiceListener listener);
  void removeServiceListener (ServiceListener listener);
}
```

Chord

Application Interface

Chord Routing API

```
void event_register ((fn)(int));
ID next_hop (ID j, ID k);
```

Chord Hashing API

```
err_t insert (void *key, void *value);
void *lookup (void *key);
```

CORBA

Interface Definition Language

Basic Types

Integer Types

short	16 bit signed integer
long	32 bit signed integer
long long	64 bit signed integer
unsigned short	16 bit unsigned integer
unsigned long	32 bit unsigned integer
unsigned long long	64 bit unsigned integer

Values

```
18, 022, 0x12, 0X12
```

Constants

```
const short aShortConstant = 6 * 7;
```

Floating Point Types

float
: 24 bit signed fraction, 8 bit signed exponent

double
: 53 bit signed fraction, 11 bit signed exponent

long double
: 113 bit signed fraction, 15 bit signed exponent

Values

```
3.14, 12.34e5, 1.2E-4
```

Constants

```
const float aFloatConstant = 3.141593;
```

Character Types

char
: character in single-byte character set

wchar
: character in multiple-byte character set

Values

```
'a', '\n', '\000', '\x12'
```

Constants

```
const char aTab = '\t';
const wchar aWideTab = L'\t';
```

Logical Types

boolean
: logical value

Values

```
TRUE, FALSE
```

Constants

```
const boolean aTrueValue = TRUE;
const boolean aFalseValue = FALSE;
```

Special Types

octet
: 8 bits of raw data

any
: container of another arbitrary type

Constructed Data Types

Structures

Declaration

```
struct aPerson
{
  string firstName;
  string lastName;
  short  age;
};
```

Exceptions

Declaration

```
exception anException
{
  string reason;
  string severity;
};
```

Standard System Exception

```
exception COMM_FAILURE
{
  unsigned long minor;
  completion_status completed;
};
```

Unions

Declaration

```
union aSillyUnion switch (short)
{
  case 1  : long aLongValue;
  case 2  : float aFloatValue;
  default : string aStringValue;
};
```

Enums

Declaration

```
enum aBaseColor { red, green, blue }
```

Arrays

Declaration

```
typedef long aLongArray [10];
```

Sequences

Declaration

```
typedef sequence<long,10> aBoundedVector;
typedef sequence<long> anUnboundedVector;
```

Strings

Declaration

```
typedef string<10> aBoundedString;
typedef string anUnboundedString;
```

Constants

```
const string aHello = "Hello\n";
const wstring aWideHello = L"Hello\n";
```

Fixed Point Types

Declaration

```
typedef fixed<10,2> aPrice;
```

Constants

```
const fixed aPrice = 12.34D;
```

Constructed Object Types

Interface Types

Declaration

```
abstract interface aParentInterface
{
  attribute string aStringAttribute;
  short aMethod (in long aLongArgument, inout float aFloatArgument);
}

interface aChildInterface : aParentInterface
{
  readonly attribute short aShortAttribute;
  oneway void aOnewayMethod (in long anArgument);
  void aTwowayMethod () raises anException;
}
```

Keywords

- local - interface not invoked remotely
- abstract - runtime determines passing semantics

- oneway - best effort delivery
- readonly - attribute without setter

Value Types

Declaration

```
valuetype aChildValue : truncatable aParentValue, supports anInterface
{
  private short aShortMember;
  public aParentValue aValueMember;
  factory aFactory (in string anArgument);
  short aLocalMethod (in long aLongArgument, in float aFloatArgument);
}
```

Keywords

- custom - custom marshalling
- abstract - base type not instantiated
- truncatable - state compatible with parent

- public - value used by clients
- private - value used by implementation
- factory - portable initializer

Language Mapping

Integer And Floating Point Types

Holder Class Example

```
public final class IntHolder
implements org.omg.CORBA.portable.Streamable
{
  public int value;

  public IntHolder () { }
  public IntHolder (int o) { value = o; }

  public TypeCode _type ()
  {
    return ORB.init ().get_primitive_tc (TCKind.tk_long);
  }

  public void _read (org.omg.CORBA.portable.InputStream in)
  {
    value = in.read_long ();
  }

  public void _write (org.omg.CORBA.portable.OutputStream out)
  {
    out.write_long (value);
  }
}
```

Character And String Types

Var Class Example

```
class String_var
{
  private:

    char *data;

  public:

    inline String_var ()        { data = 0; }
    inline String_var (char *p) { data = p; }

    inline String_var (const char *p)
    {
      if (p) data = CORBA::string_dup (p);
      else   data = 0;
    }

    inline ~String_var ()
    {
      CORBA::string_free (data);
    }

    inline String_var &operator = (char *p)
    {
      CORBA::string_free (data);
      data = p;
      return (*this);
    }

    inline operator char * () { return (data); }

    inline char &operator [] (CORBA::ULong index)
    {
      return (data [index]);
    }

    ...
}
```

Var Class Usage

```
void FunctionWithoutLeaks (void)
{
  // All strings must be allocated using specific functions
  String_var vSmartPointer = string_dup ("A string ...");

  // Except assignment from const string which copies
  const char *pConstPointer = "A const string ...";
  vSmartPointer = pConstPointer;

  // Assignment releases rather than overwrites
  vSmartPointer = string_dup ("Another string ...");

  // Going out of scope releases too
  throw (0);
}
```

Any Type

Any Class Example

```
class Any
{
  public:

    // Types passed by value are easy
    void operator <<= (Any &, Short);
    Boolean operator >>= (const Any &, Short &);
    ...

    // Types passed by reference introduce ownership issues
    void operator <<= (Any &, const Any &);
    void operator <<= (Any &, Any *);
    ...

    // Types where overloading fails introduce resolution issues
    struct from_boolean { from_boolean (Boolean b) : val (b) { } Boolean val; };
    struct from_octet { from_octet (Octet o) : val (o) { } Octet val; };
    struct from_char { from_char (Char c) : val (c) { } Char val; };
    ...

    void operator <<= (from_boolean);
    void operator <<= (from_octet);
    void operator <<= (from_char);
    ...

    struct to_boolean { to_boolean (Boolean &b) : ref (b) { } Boolean &ref; };
    ...

    Boolean operator >>= (to_boolean) const;
    ...

  private:

    // Private operators can detect resolution issues
    unsigned char void operator <<= (unsigned char);
    Boolean operator >>= (unsigned char &) const;
}
```

Any Class Insertion

```
Any oContainer;

// Small types can be stored easily
Long iLongValue = 1234;
Float fFloatValue = 12.34;
oContainer <<= iLongValue;
oContainer <<= fFloatValue;

// Constant references have copying semantics
const char *pConstString = "A string ...";
oContainer <<= pConstString;

// Non constant references have adoption semantics
```

```
String_var vString = string_dup ("A string ...");
oContainer <<= Any::from_string (vString, 0, FALSE);
oContainer <<= Any::from_string (vString._retn (), 0, TRUE);

// Some types need to be resolved explicitly
Char cChar = 'X';
Octet bOctet = 0x55;
oContainer <<= Any::from_char (cChar);
oContainer <<= Any::from_octet (bOctet);
```

Any Class Extraction

```
Any oContainer;

// Small types can be retrieved easily
Long iLongValue;
Float fFloatValue;
if (oContainer >>= iLongValue) ...;
if (oContainer >>= fFloatValue) ...;

// References remain owned by container
const char *pConstString;
if (oContainer >>= Any::to_string (pConstString, 0)) ...;

// Some types need to be resolved explicitly
Char cChar;
Octet bOctet;
if (oContainer >>= Any::to_char (cChar)) ...;
if (oContainer >>= Any::to_octet (bOctet)) ...;
```

Structures And Exceptions

Exception Class Example

```
class Exception
{
  public:

    // Method for throwing most derived type
    virtual void _raise () const = 0;
    ...
}
```

Unions

Union Class Example

```
class AUnion
{
  public:

    ...

    void _d (Short);     // Set discriminator
    Short _d() const;    // Get discriminator
    void ShortItem (Short);      // Store ShortItem and set discriminator
    Short ShortItem () const;    // Read ShortItem if stored

    void LongItem (Long);        // Store LongItem and set discriminator
    Long LongItem () const;      // Read LongItem if stored

    ...
}
```

Union Class Usage

```
AUnion oUnion;
Short iShortValue = 1234;
Long iLongValue = 5678;

// Storing sets discriminator
oUnion.ShortItem (iShortValue);
oUnion.LongItem (iLongValue);

// Retrieving must check discriminator
if (oUnion._d () == 1) iShortValue = oUnion.ShortItem ();
if (oUnion._d () == 2) iLongValue = oUnion.LongItem ();
```

Enum Types

Enum Class Example

```
public class AnEnum
{
  public static final int _red = 0;
  public static final AnEnum red = new AnEnum (_red);

  public static final int _green = 1;
  public static final AnEnum green = new AnEnum (_green);

  ...

  public int value () {...};
  public static AnEnum from_int (int value) {...};
}
```

Enum Class Usage

```
AnEnum oEnum;

// Assignments are type safe
oEnum = AnEnum.red;
oEnum = AnEnum.green;

// Switch statements use ordinal values
switch (oEnum.value ())
{
  case AnEnum._red: ...;
  case AnEmum._green: ...;
}
```

Sequences

Sequence Class Example

```
class ASequence
{
  public:

    ASequence ();
    ASequence (ULong max);
    ASequence (ULong max, ULong length, Short *data, Boolean release = FALSE);

    ...

    ULong maximum () const;
    Boolean release () const;

    void length (ULong);
    ULong length () const;

    T &operator [] (ULong index);
    const T &operator [] (ULong index) const;

    ...
}
```

Fixed Point Types

Fixed Class Example

```
class Fixed
{
  public:

    // Constructors

    Fixed (Long val);
    Fixed (ULong val);
    Fixed (LongLong val);
    Fixed (ULongLong val);
    ...
    Fixed (const char *);

    // Conversions

    operator LongLong () const;
    operator LongDouble () const;
    Fixed round (UShort scale) const;
    Fixed truncate (UShort scale) const;

    // Operators

    Fixed &operator = (const Fixed &val);
    Fixed &operator += (const Fixed &val);
    Fixed &operator -= (const Fixed &val);
    ...
}

Fixed operator + (const Fixed &val1, const Fixed &val2);
Fixed operator - (const Fixed &val1, const Fixed &val2);
...
```

Proxies

Proxy Interface Class Example

```
class AnInterface;
typedef AnInterface *AnInterface_ptr;
class AnInterface_var;


class AnInterface : public virtual Object
{
  public:

    typedef AnInterface_ptr _ptr_type;
    typedef AnInterface_var _var_type;

    static AnInterface_ptr _duplicate (AnInterface_ptr obj);
    static AnInterface_ptr _narrow (Object_ptr obj);
    static AnInterface_ptr _nil ();

    virtual ... AnOperation (...) = 0;

  protected:

    AnInterface ();
    virtual ~AnInterface ();

    ...
}
```

Proxy Var Class Example

```
class AnInterface_var : public _var
{
  protected:

    AnInterface_ptr ptr;

  public:

    AnInterface_var () { ptr = AnInterface::_nil (); }
    AnInterface_var (AnInterface_ptr p) { ptr = p; }

    ...

    ~AnInterface_var ()
    {
      release (ptr);
    }

    AnInterface_var &operator = (AnInterface_ptr p)
    {
      release (ptr);
      ptr = p;
      return (*this);
    }

    AnInterface_var &operator = (const AnInterface_var &var)
    {
      if (this != &var)
      {
        release (ptr);
        ptr = AnInterface::_duplicate (AnInterface_ptr (var));
      }
      return (*this);
    }

    operator AnInterface_ptr & () { return (ptr); }
    AnInterface _ptr operator -> () const { return (ptr); }

    ...
}
```

Proxy Class Example

```
public interface AnInterfaceOperations
{
  ... AnOperation (...) throws ...;
}

public interface AnInterface extends AnInterfaceOperations ... { }

abstract public class AnInterfaceHelper
{
  public static void insert (Any a, AnInterface t) {...}
  public static AnInterface extract (Any a) {...}
  public static AnInterface read (InputStream is) {...}
  public static void write (OutputStream os, AnInterface val) {...}
  ...

  public static AnInterface narrow (org.omg.CORBA.Object obj) {...}
  public static AnInterface narrow (java.lang.Object obj) {...}
}

final public class AnInterfaceHolder implements Streamable
{
  public AnInterface value;
  public AnInterfaceHolder () { }
  public AnInterfaceHolder (AnInterface initial) {...}

  ...
}
```

Servants

Servant Base Class

```
class ServantBase
{
  public:

    virtual ~ServantBase ();

    virtual InterfaceDef_ptr _get_interface () throw (SystemException);
    virtual Boolean _is_a (const char *logical_type_id) throw (SystemException);
    virtual Boolean _non_existent () throw (SystemException);

    virtual void _add_ref ();
    virtual void _remove_ref ();

    ...
}
```

Servant Class Example

```
class POA_AnInterface : public virtual ServantBase
{
  public:

    virtual ... AnOperation (...) = 0;

    ...
}

template <class T> class POA_AnInterface_tie : public POA_AnInterface
{
  public:

    POA_AnInterface_tie (T &t) : _ptr (t) { }

    ...

    ... AnOperation (...) { return (_ptr->AnOperation (...); }
}
```

Servant Base Class

```
class Servant
{
  public:

    virtual IDL::traits<CORBA::InterfaceDef>::ref_type _get_interface ();
    virtual bool _is_a (const std::string &logical_type_id);
    virtual bool _non_existent ();

    ...

  protected:

    virtual ~Servant ();
}
```

Servant Class Example

```
class _AnInterface_Servant_Base : public virtual Servant
{
  public:

    virtual ... AnOperation (...) = 0;

    ...
}

class AnInterface_Servant : public virtual CORBA::servant_traits<AnInterface>::base_typ
{
  public:

    virtual ... AnOperation (...) override;
}
```

Servant Base Class

```
abstract public class Servant
{
  final public Delegate _get_delegate () { ... }
  final public void _set_delegate (Delegate delegate) { ... }
  ...
}
```

Servant Class Example

```
abstract public class AnInterfacePOA implements AnInterfaceOperations
{
  public AnInterface _this () { ... }
  ...
}

public class AnInterfacePOATie extends AnInterfacePOA
{
  private AnInterfaceOperations _delegate;

  public AnInterfacePOATie (AnInterfaceOperations delegate)
  { _delegate = delegate; }

  public AnInterfaceOperations _delegate ()
  { return (_delegate); }

  public void _delegate (AnInterfaceOperations delegate)
  { _delegate = delegate; }

  public ... AnOperation (...) { return (_delegate.AnOperation (...); }
}
```

Value Types

Value Mapping Example

```
class AValue : public virtual ValueBase
{
  public:

    virtual void ShortItem (Short) = 0;
    virtual Short ShortItem () const = 0;

    virtual void LongItem (Long) = 0;
    virtual Long LongItem () const = 0;

    ...

    virtual ... AnOperation (...) = 0;
}

class OBV_AValue : public virtual AValue
{
  public:

    virtual void ShortItem (Short) { ... };
    virtual Short ShortItem () const { ... };

    virtual void LongItem (Long) { ... };
    virtual Long LongItem () const { ... };

    ...

    virtual ... AnOperation (...) = 0;
}

class ValueFactoryBase
{
  private:

    virtual ValueBase *create_for_unmarshal () = 0;

    ...
}

class AValue_init : public ValueFactoryBase
{
  public:

    virtual AValue *AConstructor (...) = 0;

    ...
}
```

Object Adapter

Object Adapter Configuration

```
local interface POA
{
  POA create_POA (in string adapter_name,
                  in POAManager manager,
                  in CORBA::PolicyList policies);

  ThreadPolicy create_thread_policy (in ThreadPolicyValue value);
  LifespanPolicy create_lifespan_policy (in LifespanPolicyValue value);
  ServantRetentionPolicy create_servant_retention_policy (in ServantRetentionPolicyValu
  RequestProcessingPolicy create_request_processing_policy (in RequestProcessingPolicyV

  ...
};

local interface POAManager
{
  enum State { HOLDING, ACTIVE, DISCARDING, INACTIVE };
  State get_state ();

  void activate () raises (AdapterInactive);
  void hold_requests (in boolean wait_for_completion) raises (AdapterInactive);
  void discard_requests (in boolean wait_for_completion) raises (AdapterInactive);

  void deactivate (in boolean etherealize_objects,
                   in boolean wait_for_completion);
};
```

Thread Policy

Thread Policy Values

- SINGLE_THREAD_MODEL - calls to servants and managers are serialized
- MAIN_THREAD_MODEL - calls to servants are using single main thread
- ORB_CTRL_MODEL - calls use arbitrary threading model

Object Identity Policies

ID Uniqueness Policy Values

- UNIQUE_ID - servants have exactly one object ID
- MULTIPLE_ID - servants have at least one object ID

ID Assignment Policy Values

- USER_ID - object ID is assigned by application
- SYSTEM_ID - object ID is assigned by object adapter

Implicit Activation Policy Values

- IMPLICIT_ACTIVATION - assign object ID on demand
- NO_IMPLICIT_ACTIVATION - do not assign object ID on demand

Object Activation

```
ObjectId activate_object (in Servant servant) raises (ServantAlreadyActive, WrongPolicy

void activate_object_with_id (in ObjectId oid, in Servant servant)
raises (ObjectAlreadyActive, ServantAlreadyActive, WrongPolicy);

void deactivate_object (in ObjectId oid) raises (ObjectNotActive, WrongPolicy);

Object create_reference (in CORBA::RepositoryId ifc) raises (WrongPolicy);
Object create_reference_with_id (in ObjectId oid, in CORBA::RepositoryId ifc);

Object servant_to_reference (in Servant servant) raises (ServantNotActive, WrongPolicy)
Servant reference_to_servant (in Object reference) raises (ObjectNotActive, WrongAdapte
```

Current Object Interface

```
local interface Current
{
  POA get_POA () raises (NoContext);
  ObjectId get_object_id () raises (NoContext);
  Object get_reference () raises (NoContext);
  Servant get_servant () raises (NoContext);
};
```

Servant Lookup Policies

Servant Retention Policy Values

- RETAIN - keep track of active servants
- NON_RETAIN - do not keep track of active servants

Request Processing Policy Values

- USE_ACTIVE_OBJECT_MAP_ONLY - only deliver to tracked servants
- USE_DEFAULT_SERVANT - alternatively deliver to default servant
- USE_SERVANT_MANAGER - alternatively activate servants on demand

Servant Activator Interface

```
local interface ServantActivator : ServantManager
{
  Servant incarnate (in ObjectId oid,
                     in POA adapter)
  raises (ForwardRequest);

  void etherealize (in ObjectId oid,
                    in POA adapter,
                    in Servant servant,
                    in boolean cleanup_in_progress,
                    in boolean remaining_activations};
};
```

Servant Locator Interface

```
local interface ServantLocator : ServantManager
{
  native Cookie;

  Servant preinvoke (in ObjectId oid,
                     in POA adapter,
                     in CORBA::Identifier operation,
                     out Cookie cookie)
  raises (ForwardRequest);

  void postinvoke (in ObjectId oid,
                   in POA adapter,
                   in CORBA::Identifier operation,
                   in Cookie cookie,
                   in Servant servant);
};
```

Lifespan Policy

Lifespan Policy Values

- TRANSIENT - object references have lifetime of object adapter
- PERSISTENT - object references have potentially unlimited lifetime

Request Forward Exception

```
exception ForwardRequest
{
  Object forward_reference;
};
```

Messaging

Synchronization Scope Policy

- SYNC_NONE
- SYNC_WITH_TRANSPORT
- SYNC_WITH_SERVER
- SYNC_WITH_TARGET

Routing Policy

- ROUTE_NONE
- ROUTE_FORWARD
- ROUTE_STORE_AND_FORWARD

Asynchronous Messaging Mapping Example

```
interface StockManager
{
  attribute string stock_exchange_name;
  boolean add_stock (in string symbol, in double quote);
  void remove_stock (in string symbol, out double quote) raises (InvalidStock);
};


void sendc_get_stock_exchange_name (
  in AMI_StockManagerHandler ami_handler);
void sendc_set_stock_exchange_name (
  in AMI_StockManagerHandler ami_handler,
  in string attr_stock_exchange_name);

void sendc_add_stock (
  in AMI_StockManagerHandler ami_handler,
  in string symbol,
  in double quote);

void sendc_remove_stock (
  in AMI_StockManagerHandler ami_handler,
  in string symbol);


AMI_StockManagerPoller sendp_get_stock_exchange_name ();
AMI_StockManagerPoller sendp_set_stock_exchange_name (
  in string attr_stock_exchange_name);

AMI_StockManagerPoller sendp_add_stock (
  in string symbol, in double quote);
```

```
AMI_StockManagerPoller sendp_remove_stock (
  in string symbol);


interface AMI_StockManagerHandler : Messaging::ReplyHandler
{
  void get_stock_exchange_name (
    in string ami_return_val);
  void get_stock_exchange_name_excep (
    in Messaging::ExceptionHolder excep_holder);

  void set_stock_exchange_name ();
  void set_stock_exchange_name_excep (
    in Messaging::ExceptionHolder excep_holder);

  void add_stock (in boolean ami_return_val);
  void add_stock_excep (
    in Messaging::ExceptionHolder excep_holder);

  void remove_stock (in double quote);
  void remove_stock_excep (
    in Messaging::ExceptionHolder excep_holder);
};


valuetype AMI_StockManagerPoller : Messaging::Poller
{
  void get_stock_exchange_name (
    in unsigned long timeout,
    out string ami_return_val);
  void set_stock_exchange_name (
    in unsigned long timeout);

  void add_stock (
    in unsigned long timeout,
    out boolean ami_return_val);

  void remove_stock (
    in unsigned long timeout,
    out double quote) raises (InvalidStock);
};
```

Components

Component Features

- attributes - denote configurable properties
- supported interface - inherited in all interfaces
- facets - interfaces provided to the outside
- receptacles - interfaces required from the outside
- sources - events produced to the outside
- sinks - events consumed from the outside

Component Definition Example

```
module DiningPhilosophers
{
  interface IFork
  {
    void pick_up () raises (ForkNotAvailable);
    void release ();
  };

  component AFork
  {
    provides IFork fork;
  };

  eventtype PhilosopherStatus
  {
      public string name;
      public PhilosopherState state;
      public boolean has_left_fork;
      public boolean has_right_fork;
  };

  component APhilosopher
  {
    attribute string name;

    // Receptacles for forks
    uses Fork left;
    uses Fork right;

    // Source for status
    publishes PhilosopherStatus status;
  };

  component AnObserver
  {
    // Sink for status
    consumes PhilosopherStatus status;
  };

  ...
};
```

Navigation Interfaces

```
module Components
{
  typedef string FeatureName;
  typedef sequence<FeatureName> NameList;

  valuetype PortDescription
  {
    public FeatureName name;
    public CORBA::RepositoryId type_id;
  };

  ...

  valuetype FacetDescription : PortDescription
  {
    public Object facet_ref;
  };
  typedef sequence<FacetDescription> FacetDescriptions;
  interface Navigation
  {
    FacetDescriptions get_all_facets ();
    Object provide_facet (in FeatureName name) raises (InvalidName);
    FacetDescriptions get_named_facets (in NameList names) raises (InvalidName);

    ...
  };

  ...

  valuetype PublisherDescription : PortDescription
  {
    public SubscriberDescriptions consumers;
  };
  typedef sequence<PublisherDescription> PublisherDescriptions;

  valuetype ConsumerDescription : PortDescription
  {
    public EventConsumerBase consumer;
  };
  typedef sequence<ConsumerDescription> ConsumerDescriptions;

  PublisherDescriptions get_all_publishers ();
  PublisherDescriptions get_named_publishers (in NameList names) raises (InvalidName);

  ConsumerDescriptions get_all_consumers ();
  ConsumerDescriptions get_named_consumers (in NameList names) raises (InvalidName);

  ...
};
```

Assembly Interfaces

```
uses AnInterface AReceptacle;
consumes AnEvent ASink;

void connect_AReceptacle (in AnInterface connection)
  raises (AlreadyConnected, InvalidConnection);
AnInterface disconnect_AReceptacle ()
  raises (NoConnection);
AnInterface get_connection_AReceptacle ();

AnEventConsumer get_consumer_ASink ();

module Components
{
  ...

  interface Receptacles
  {
    Cookie connect (in FeatureName name, in Object connection)
      raises (InvalidName, InvalidConnection, AlreadyConnected, ExceededConnectionLimit
    Object disconnect (in FeatureName name, in Cookie ck)
      raises (InvalidName, InvalidConnection, CookieRequired, NoConnection);
    ConnectionDescriptions get_connections (in FeatureName name)
      raises (InvalidName);

    ...
  };

  ...
  valuetype Cookie
  {
    private CORBA::OctetSeq cookieValue;
  };

  valuetype SubscriberDescription
  {
    public Cookie ck;
    public EventConsumerBase consumer;
  };
  typedef sequence<SubscriberDescription> SubscriberDescriptions;

  interface Events
  {
    void connect_consumer (in FeatureName emitter_name, in EventConsumerBase consumer)
      raises (InvalidName, AlreadyConnected, InvalidConnection);
    EventConsumerBase disconnect_consumer (in FeatureName source_name)
      raises (InvalidName, NoConnection);
    EventConsumerBase get_consumer (in FeatureName sink_name) raises (InvalidName);

    Cookie subscribe (in FeatureName publisher_name, in EventConsumerBase subscriber)
      raises (InvalidName, InvalidConnection, ExceededConnectionLimit);
    EventConsumerBase unsubscribe (in FeatureName publisher_name, in Cookie ck)
      raises (InvalidName, InvalidConnection);

    ...
  };

  ...
};
```

JAX-RS

Service Example

```
@Path ("example")
public class ExampleResource {
  @GET
  @Path ("{key}")
  @Produces ("application/value+xml")
  public KeyValueType getKeyValue (@PathParam ("key") String key) {...}

  @DELETE
  @Path ("{key}")
  public void deleteKeyValue (@PathParam ("key") String key) {...}
}
```

JAX-WS

Mapping Example Java Input

```
package com.example;

@WebService
public interface StockQuoteProvider {
  float getPrice (String tickerSymbol)
```

```
    throws TickerException;
}

// Example adjusted from JAX-WS 2.2 Specification.
```

Mapping Example WSDL Output (Document Style)

```
<types>
  <xsd:schema targetNamespace="...">
    <xsd:element name="getPrice" type="tns:getPriceType"/>
    <xsd:element name="getPriceResponse" type="tns:getPriceResponseType"/>
    <xsd:element name="TickerException" type="tns:TickerExceptionType"/>
    ...
  </xsd:schema>
</types>

<message name="getPrice">
  <part name="getPrice" element="tns:getPrice"/>
</message>
<message name="getPriceResponse">
  <part name="getPriceResponse" element="tns:getPriceResponse"/>
</message>
<message name="TickerException">
  <part name="TickerException" element="tns:TickerException"/>
</message>

<portType name="StockQuoteProvider">
  <operation name="getPrice">
    <input message="tns:getPrice" wsam:action="..."/>
    <output message="tns:getPriceResponse wsam:action="..."/>
    <fault message="tns:TickerException wsam:action="..."/>
  </operation>
</portType>

<!-- Example adjusted from JAX-WS 2.2 Specification. -->
```

Mapping Example WSDL Output (RPC Style)

```
<types>
  <xsd:schema targetNamespace="...">
    <xsd:element name="TickerException" type="tns:TickerExceptionType"/>
    ...
  </xsd:schema>
</types>

<message name="getPrice">
  <part name="tickerSymbol" type="xsd:string"/>
</message>
<message name="getPriceResponse">
  <part name="return" type="xsd:float"/>
</message>
<message name="TickerException">
  <part name="TickerException" element="tns:TickerException"/>
</message>

<portType name="StockQuoteProvider">
  <operation name="getPrice">
    <input message="tns:getPrice"/>
    <output message="tns:getPriceResponse"/>
    <fault message="tns:TickerException"/>
  </operation>
```

```
</portType>

<!-- Example adjusted from JAX-WS 2.2 Specification. -->
```

Django

Models

```
class Author (models.Model):
  name = models.CharField (max_length = 255)
  born = models.DateField (null = True, blank = True)
  died = models.DateField (null = True, blank = True)
  bio = models.TextField (null = True, blank = True)

    def __unicode__ (self):
      return self.name

    class Meta:
      ordering = ["name"]

class Book (models.Model):
    name = models.CharField (max_length = 255)
    date = models.DateField ()
    authors = models.ManyToManyField (Author, related_name = "books")
    description = models.TextField (null = True, blank = True)

    def __unicode__ (self):
        authors = [author.name for author in self.authors.all ()]
        return ", ".join (authors) + ": "  + self.name

    class Meta:
        ordering = ["name", "date"]
```

Views

```
def index (request):
  "Display index page."
  authors = Author.objects.all ()
  return render_to_response ("index.html", {"authors" : authors})

def author (request, author_id):
  "Display author page."
  author = get_object_or_404 (Author, id = author_id)
  return render_to_response ("author.html", {"author" : author})

urlpatterns = patterns (
  "views",
  url (r"^$", "index", name="index"),
  url (r"^author/(?P<author_id>\d+)$", "author", name="author")
)
```

Templates

```
<html>
  <head>
    <title>Author Index</title>
  </head>
  <body>
    <ul>
      {% for author in authors %}
        <li><a href="{$ url author author.id %}">{{ author.name }}</a></li>
      {% endfor %}
    </ul>
  </body>
</html>
```

Forms

```
class BookForm (forms.Form):
  name = forms.CharField (max_length = 1024, label = u"Book Title")
  date = forms.DateField (initial = date.today (), label = u"Issue Date")
  author = forms.ChoiceField (
    choices = [(author.id, author.name) for author in Authors.all ()],
    label = u"First Author")
  description = forms.CharField (
    widget = forms.Textarea,
    required = False,
    label = u"Description")

def book_view (request):
  "Book view."

  if request.method == "POST":
    form = BookForm (request.POST)
    if form.is_valid ():
      book = Book (
        name = form.cleaned_data ["name"],
        date = form.cleaned_data ["date"],
        authors = [get_object_or_404 (Authors, form.cleaned_data ["author"])]
        description = form.cleaned_data ["description"])
      book.save ()
      return redirect ("author", author_id = book.authors [0])

  form = BookForm ()
  return render_to_response ("book.html", {"form" : form})

<form action="{% url book %}" method="post">
  <table>
    {{ form.as_table }}
    <tr>
      <th></th>
      <td><input type="submit" value="Enter" /></td>
    </tr>
  </table>
</form>
```

Hadoop

Architecture

- cluster file system (HDFS)
  - single name node
    - maps names to blocks
    - maps blocks to data nodes
    - single point of failure (metadata backed up)
  - multiple data nodes
    - store blocks (size configurable per file)
    - potentially replicated (count configurable per file)
    - replica placement balances rack and node utilization
  - interface similar to standard file system
    - files and directories
    - writing data only once
    - mapping to C, Java, REST
- cluster map reduce implementation
  - single JobTracker node
    - schedules tasks to nodes
    - retrying failed tasks
  - multiple TaskTracker nodes

Map Reduce Word Count Example

```
class WordCount {

  class TokenizerMapper
    extends Mapper<Object, Text, Text, IntWritable> {

    // Local variables for the map function.
    // Here to avoid frequent allocation.
    IntWritable one = new IntWritable (1);
    Text word = new Text ();

    void map (Object key, Text value, Context context) {
      StringTokenizer itr = new StringTokenizer (value.toString ());
      while (itr.hasMoreTokens ()) {
        word.set (itr.nextToken ());
        context.write (word, one);
      }
    }
  }

  class IntSumReducer
    extends Reducer<Text,IntWritable,Text,IntWritable> {

    // Local variables for the reduce function.
    // Here to avoid frequent allocation.
    IntWritable result = new IntWritable ();

    void reduce (Text key, Iterable<IntWritable> values, Context context) {
      int sum = 0;
      for (IntWritable val : values) {
        sum += val.get ();
      }
      result.set (sum);
      context.write (key, result);
    }
  }

  public static void main (String [] args) {

    ...

    Job job = new Job (conf, "Word Count Job");
    job.setJarByClass (WordCount.class);
    job.setMapperClass (TokenizerMapper.class);
    job.setCombinerClass (IntSumReducer.class);
    job.setReducerClass (IntSumReducer.class);
    job.setOutputKeyClass (Text.class);
    job.setOutputValueClass (IntWritable.class);

    FileInputFormat.addInputPath (job, new Path ("input"));
    FileOutputFormat.setOutputPath (job, new Path ("output"));

    System.exit (job.waitForCompletion (true) ? 0 : 1);
  }
}

// Example adjusted from sources, see references.
// Apache License applies to the example.
```

Pastry

Application Interface

Pastry Endpoint API

```
interface Node
{
  Endpoint registerApplication (Application application, java.lang.String instance);
  ...
}

interface Endpoint
{
  Id getId ();

  // Get a list of nodes that can route towards a node
  NodeHandleSet localLookup (Id id, int num, boolean safe);
  // Get the list of neighbor nodes
  NodeHandleSet neighborSet (int num);

  // Get the range of keys we are responsible for
  IdRange range (NodeHandle handle, int rank, Id lkey);
  // Get a list of nodes that can store a replica
  NodeHandleSet replicaSet (Id id, int maxRank);

  // Send a message to a node
  void route (Id id, Message message, NodeHandle hint);
  ...
}
```

Pastry Application API

```
interface Application
{
  void deliver (Id id, Message message);
  boolean forward (RouteMessage message);
  void update (NodeHandle handle, boolean joined);
}
```

Chimera

Application Interface

Chimera Initialization API

```
// Initialization for listening on given port
ChimeraState *chimera_init (int port);

// Join a network using a known bootstrap host
void chimera_join (ChimeraState *state, ChimeraHost *bootstrap);

// Manage objects representing hosts
ChimeraHost *host_get (ChimeraState *state, char *hostname, int port);
void host_release (ChimeraState *state, ChimeraHost *host);

// Upcall for host membership change notification
typedef void (*chimera_update_upcall_t) (Key *key, ChimeraHost *host, int joined);
void chimera_update (chimera_update_upcall_t func);
```

Chimera Communication API

```
// Register a message type
void chimera_register (ChimeraState *state, int type, int ack);

// Create a key from a string
void key_makehash (void *log, Key *hashed, char *s);

// Send a message to host responsible for key
void chimera_send (ChimeraState *state, Key key, int type, int len, char *data);

// Upcall for message forward event
typedef void (*chimera_forward_upcall_t) (Key **key, Message **msg, ChimeraHost **host)
void chimera_forward (chimera_forward_upcall_t func);
```

```
// Upcall for message delivery event
typedef void (*chimera_deliver_upcall_t) (Key *key, Message *msg);
void chimera_deliver (chimera_deliver_upcall_t func);
```

JGroups

Protocol Modules

Transport Protocol Modules

- UDP - uses IP multicast to deliver multicast messages
- TCP - uses mesh of TCP connections to deliver multicast messages
- TUNNEL - tunnels transport to specialized router

Discovery Protocol Modules

- PING - uses IP multicast over existing UDP transport
- MPING - uses IP multicast over separate UDP transport
- BPING - uses IP broadcast
- TCPPING - uses list of member addresses
- TCPGOSSIP - uses specialized router
- FILE_PING - uses shared directory to keep track of members
- JDBC_PING - uses shared database to keep track of members
- RACKSPACE_PING - uses Rackspace Cloud File Storage
- S3_PING - uses Amazon Simple Storage Service

Merge Protocol Modules

- MERGE2 - group coordinator multicasts presence and membership view
- MERGE3 - all members multicast presence and membership view

Failure Detection Modules

- FD - uses periodic ping in logical ring
- FD_ALL - uses multicast heartbeat
- FD_SOCK - uses TCP socket ring
- FD_PING - uses external executable to ping members
- FD_ICMP - uses internal library method to ping members
- VERIFY_SUSPECT - verify suspect members additionally

Reliable Message Transmission Modules

- NAKACK - uses negative acknowledgments and sequence numbering, old version
- NAKACK2 - uses negative acknowledgments and sequence numbering, new version
- UNICAST - uses positive acknowledgments and sequence numbering, for unicast messages
- UNICAST2 - uses negative acknowledgments and sequence numbering, for unicast messages

Miscellaneous Modules

- UFC - rate limiting flow control for unicast
- MFC - rate limiting flow control for multicast
- FRAG - message fragmentation
- STABLE - atomic delivery in group
- SEQUENCER - totally ordered delivery through coordinator

- AUTH - member authentication
- ENCRYPT - message body encryption
- COMPRESS - message body compression

Channels

Channel Class

```
public class JChannel ...
{
  // Initialization can accept either options or configuration file
  public JChannel (String props) throws Exception;

  // Join a group with a given name
  public void connect (String cluster) throws Exception;
  public void disconnect ();

  // View is the current list of members
  public View getView ();

  public void send (Message msg) throws Exception;
  public void send (Address dst, byte[] buf) throws Exception;
  public void send (Address dst, Serializable obj) throws Exception;
  public void receive (Message msg);

  // Asynchronous notification about messages and membership is available
  public void setReceiver (Receiver r);

  ...
}
```

Receiver Interface

```
public interface MessageListener
{
  void receive (Message msg);

  // Group members can share state
  void getState (OutputStream output) throws Exception;
  void setState(InputStream input) throws Exception;
}

public interface MembershipListener
{
  // Notification about membership view change
  public void viewAccepted (View view);

  // Indication of suspect member before actual removal
  public void suspect (Object suspected);

  // Notifies about temporary suspension during view update
  public void block ();
  public void unblock ();
}

public interface Receiver extends MessageListener, MembershipListener;
```

Bigtable

Table Model

- sparse rectangular table with strings as row and column keys
- rows kept in lexicographical order with dynamic partitioning
- columns grouped into families for access control
- cell is an uninterpreted array of bytes
- cell can have timestamped versions

Operations

```
Table *T = OpenOrDie ("/bigtable/webtable");

// Select row for modification.
RowMutation row (T, "row.key");

// Adjust row by adding and deleting columns.
row.Set ("family:column.key.one", "value");
row.Delete ("family:column.key.two");

// Perform the operation atomically.
Operation op;
Apply (&op, &row);

// Prepare for iteration.
Scanner scanner (T);
ScanStream *stream;
```

```
stream = scanner.FetchColumnFamily ("family");
stream->SetReturnAllVersions ();
scanner.Lookup ("row.key");

// Iterate over returned values.
for ( ; !stream->Done () ; stream->Next ())
{
  printf ("%s %s %lld %s\n",
          scanner.RowName (),
          stream->ColumnName (),
          stream->MicroTimestamp (),
          stream->Value ());
}

// Example adjusted from literature, see references.
```

- locate row by key and time
- iterate over rows with lexicographic ordering of keys
- limit results by column family
- limit results by timestamp
- run script on server
- ...

- row operations atomic
- transactions over single rows
- garbage collection of old versions by number or by time

Scalability

- Using 1786 four core nodes to host GFS
- 500 nodes also act as table servers
- 500 nodes also act as clients
- 500 GB table
  - random 1kB reads 120 MB/s total
  - sequential 1kB reads 1235 MB/s total
  - random 1kB writes 1000 MB/s total

- Google Crawler, 800 TB table, 1 trillion cells
- Google Earth, 70 TB table, 9 billion cells

Memcached

Architecture

- clients access data
  - data as key value pairs
  - key is a string (250 B limit)
  - value is array of bytes (1 MB limit)
  - client side compression supported

- servers cache data
  - standardized protocols
    - can use TCP or UDP
    - transparent binary protocol
    - text protocol limits keys and values

  - servers do not know of each other
  - server selected by client side hashing
  - least recently used strategy for eviction

Usage Example

```
// Initialization of memcache client.
mem = new Memcache ()
mem.add_server ("10.0.0.1:12345")
mem.add_server ("10.0.0.2:12345")
mem.add_server ("10.0.0.3:12345")

// Construct key for database query.
sql = "SELECT * FROM user WHERE name = ?"
key = "SQL:" + hash (sql) + name

// Try to fetch value from memcache.
if (defined (result = mem.get (key))) return (result)

// Fetch value from database and populate memcache otherwise.
result = execute_query (sql, name)
mem.set (key, result, lifetime)
return (result)

// Example adjusted from documentation, see references.
```

- getting and setting values of keys
- atomic setting of values for new keys
- incrementing and decrementing of integer values
- invalidating values by keys

JavaSpaces

Space Interface

```
interface JavaSpace
{
  Lease write (Entry e, Transaction tx, long lease);

  Entry read (Entry template, Transaction tx, long timeout);
  Entry readIfExists (Entry template, Transaction tx, long timeout);

  Entry take (Entry template, Transaction tx, long timeout);
  Entry takeIfExists (Entry template, Transaction tx, long timeout);

  EventRegistration notify (
    Entry template,
    Transaction tx,
    RemoteEventListener listener,
    long lease,
    MarshalledObject handback);

  Entry snapshot (Entry e);
}
```

- write - write an entry into space
- read - read (but preserve) a matching entry from space (blocking)
- take - read (and remove) a matching entry from space (blocking)
- notify - notify about a matching entry in space

Entry Class Example

```
import net.jini.core.entry.Entry;

public class Message implements Entry
{
  // Attributes that form the content of an entry
  public String text;

  // An obligatory parameterless constructor
  public Message () { }
}
```

Usage Example

```
Message m = new Message ();
m.text = "FOO";
space.write (m, null, Lease.FOREVER);

Message t = new Message ();
Message r = (Message) space.take (t, null, Long.MAX_VALUE);
```