CORBA Services

The Object Management Architecture

Object Services
- Naming Service
- Event Service
- Lifecycle Service
- Externalization

Common Facilities
- User Interface
- Information Management
- Task Management
- System Management

Domain Services
- CORBAmed
- CORBAfinancials
- CORBAm manufacturing
- CORBA tel

Application Objects
Collection Service

• Grouping of objects
  – Sets, queues, sequences
  – Iterators for these collections
  – Factory classes for object creation

• Collection interface
  – Set, Heap, Stack, Queue, SortedSet are subclasses

• Iterator interface
  – EqualityIterator, SequentialIterator are subclasses

• Operations interface
  – Base class for operations on objects
Concurrent Service

• Framework for managing concurrent object access
  – Analogous to multithreading support in C++/Java
  – Facilities for interfacing with transaction service

• Concurrency Service assumes usage of locks
  – Read, write-locks, multi-possession, two-phase locks
  – Conflicts with existing locks are resolved by a first-come first-served queuing model

• Interfaces to represent resources:
  – LockSet, TransactionalLockSet
  – LockSetFactory interface for object creation
Event Service

• Flexible framework for asynchronous object interactions
  – DII provides basic form of asynchrony
  – Event Service targeted to notification, rather than interaction

• Event Service allows to set up event channels

• CORBA objects may act as
  – Event consumers
  – Event suppliers

• Push or pull event communication

• Interfaces:
  – PushConsumer, PullConsumer, PushSupplier, PullSupplier, EventChannel
Externalization Service

• Means for Object conversion
  – Allow export over general media (network streams, disk storage, etc.)
  – Constitute data back into object references (potentially in a different ORB / process)

• Pluggable data formats for externalized objects
  – Standard serialized format for objects is provided
  – Streaming model for externalizing and internalizing objects

• Interfaces:
  – Stream, StreamFactory, FileStreamFactory, etc;
  – Objects must extend Streamable interface

• Externalization Service uses Life Cycle Service
Licensing Service

• Controlled access to objects and services under a licensing model
  – Conceptually an extension of security service

• Operation
  – Client requests licensed service, proofs ownership of license
  – Service provider checks with license manager
  – Service provider may request notification on license expiration of may poll for changed in license state

• Interfaces
  – LicenseServerManager, ProducerSpecificLicenseService

• License Service depends on Security Service
Life Cycle Service

- Standard protocols for distributed objects:
  - Creation, copying, movement, deletion of remote objects
- Service defined around the concept of object factories
- Interfaces
  - `LifeCycleObject`
  - `FactoryFinder` interface for locating object factories
- Life Cycle Service references Naming Service
  - When dealing with connected graphs of objects, the Relationship Service structures are referenced
Naming Service

• Most commonly used service in CORBA
  – Provides the principle way for clients to find objects on the network
  – Remote object references can be bound to names
  – Clients may access object references by providing the name

• A NamingContext represents a directory or subdirectory of named objects
  – This interface can be used to bind, lookup, or unbind objects and subcontexts within the naming directory
  – Names of objects within a NamingContext are composed of NameComponent arrays
  – Browsing can be done with a BindingIterator
Notification Service

• Extends asynchronous msg exchange of Event Service
  – Allows multiple event suppliers to send events to multiple event consumers
  – Supports pull and push models
  – Allows event channels to be federated
  – Allows clients to attach filters to each proxy in an event channel

• QoS properties:
  – Per-channel, per-proxy, per-event

• CORBA Notification Service uses:
  – StructuredEvent, EventChannel, EventType classes
  – StructuredPushConsumer, StructuredPushSupplier classes
Persistent Object Service

- Common framework to interact with persistency eng.:
  - Relational databases, object databases, etc.
  - Middleware between CORBA objects and database protocols (ODMG)

- Persistency…
  - Managed at object level or data member level
  - Objects typically control their own persistent state

- Interfaces:
  - $\text{PO}$ (persistent object), $\text{PID}$ (persistent object identifier), $\text{POM}$ (persistent object manager)

- Persistent object service depends on:
  - Externalization Service, Life Cycle Service
Property Service

- Defines name/value pairs that can be assigned to objects
  - Without being explicitly defined by their IDL interfaces
  - Can represent any application-specific attributes
- Property Service does not specify how properties are associated with objects
  - Implementation detail
  - Properties are represented as string name and an Any value
- Interfaces:
  - PropertySet, PropertySetDef (inquire metadata about properties - read/write, read-only, etc.)
  - PropertiesIterator, PropertySetFactory
Query Service

- General query mechanism for distributed objects
  - Collections of objects can be searched to generate subcollections
  - Subsets of objects within a collection can be deleted or updated

- Query Service’s facilities can be mapped to persistent storage facilities
  - Relational databases, object databases

- Interfaces:
  - Collection objects (with an Iterator), CollectionFactory
  - QueryManager, QueryEvaluator
  - Result of a query typically is a Collection object
Relationship Service

• Allows for explicit specification of relationships among objects
  – Defined in terms of type, roles within the relationship, and the cardinality of each role
  – Objects fulfill a role when they participate in a relationship
  – Agent/proxy relationship: one agent, multiple proxies

• Interfaces:
  – Relationship, Role
  – RelationshipFactory, RoleFactory, RelationshipIterator
  – CosGraphs, CosContainment, CosReference
Security Service

• Provides the tools to secure distributed applications
  – Authentication, access control for users
  – Secure communication channels

• High-level security framework
  – Implementations are free to use any cryptographic framework
  – Layers security measures on top of ORB object-to-object model

• Interfaces:
  – PrincipleAuthenticator, Credentials object assigned to each user
  – Current object identifies security measures for current execution context
  – Extensions to the org.omg.CORBA.Object interface
Time Service

• Ability to enquire accurate time value + estimated error
  – Uses Universal Coordinated time representation
  – Time intervals of 100 nanoseconds since Oct 15, 1582
  – Times are relative to Greenwich Time Zone

• Time-based events, linear positioning of events
  – Implementation of time service is responsible for communication with accurate time source (Cesium clock, radio time broadcast, etc.)

• Interfaces:
  – TimeService object, UTO (Universal time objects)
  – TimerEventT, TimerEventService, TimerEventHandler

• Timer event portion depends on Event service
Trading Object Service

- **Market trading context**
  - Objects describe services offered to the system
  - Clients issue description of desired service
  - Trading service performs matching

- **Interfaces:**
  - **Lookup** interface to advertise needs of importers
  - **Register** interface to advertise properties of a service
  - **OfferIterator** to iterate through multiple offers (hits)
  - **Admin** interface to query for all outstanding offers and queries and to control matching process
Transaction Service

• Defines interfaces to allow distributed objects to create and engage in transactional interactions

• ACID properties
  – *Atomic* - any and all actions carried out as part of a transaction are committed or undone/cancelled
  – *Consistent* - actions within a transaction produce results that remain consistent
  – *Isolated* - transactions do not see each other’s effects until they are committed. If they are rolled back, their effects are not seen by other contexts
  – *Durable* - if a transaction completes successfully, its effects are made persistent
Transaction Service (contd.)

- Transaction can involve a series of remote method calls
  - Whole transaction is rolled back when a significant error is encountered
  - Transaction contexts are propagated along the way

- Service provides framework for notification and management of transaction boundaries
  - Little help with implementation of rollback operations

- Interfaces:
  - Current interface: start and end of transactions
  - Control interface: manipulation of ongoing transactions
  - Terminator, Coordinator, Resource objects

- Depends on Concurrency and Persistent Object Services