Architecture of the CORBA Component Model

Object Services: Naming Service

What Are Object Services

- Infra structure services, independent from specific application
- Defined in IDL
- Can be implemented independent from a specific CORBA implementation (cf. ORB services)
- Different functional groups:
  - Registering and finding objects (Naming, Trading)
  - Asynchronous communication (Event, Notification)
  - Object life cycle (life cycle, externalization)
  - Time service
  - Licensing service
  - ...

Naming Service

- Defined in formal/02-09-02
- Associates names with object references
- Inserting a name: binding
- Deleting a name: unbinding

Yet Another Level of Indirection?

Finding the „initial“ application object is now deferred to finding the name service, and knowing the name of the object

- Why not use just IORs?
  - bulky, due to their size
  - not human readable
  - simplifies restart of the server, and relocation of the service:
    - Contact information (host, port, object key) may change, name stays
- ORB provides standard interface to locate name service:

```
orb->resolve_initial_references("NameService);
```

ORB Configuration

- Proprietary approaches:
  - ORBacus: Configuration file with Java properties
    - Java: System property ooc.config points to file
    - C++: Environment variable ORBACUS_CONFIG
    - ORBacus 4.0: ooc.orb.service.NameService=...
  - omniORB: Configuration file, environment variables, Windows registry
    - InitRef = NameService = ...
    - Registry: HKEY_LOCAL_MACHINE\Software\omniORB
  - JDK 1.2/1.3
    - Initial references via port 900 using tnameserv
    - command line: -ORBInitialHost/-ORBInitialPort
- Interoperable Naming Service (INS):
  - Command line argument –ORBInitRef NameService=...

Names

```
module CosNaming {
   typedef string Istring;
   struct NameComponent {
       Istring id;
       Istring kind;
   };
   typedef sequence<NameComponent> Name;
};
```

- Name defines path relative to naming context
- String version: / separates name components, . separates id and kind:

  outercontext/innercontext.ctx/MyServer.service

Bindings

```
module CosNaming {
    enum BindingType {nobject, ncontext};
    struct Binding {
        Name binding_name;
        BindingType binding_type;
    };
};
```

Inserting Names

```
module CosNaming{
  interface NamingContext {
   // Ausnahmen hier weggelassen
   void bind(in Name n, in Object obj)
     raises(NotFound, CannotProceed,
            InvalidName, AlreadyBound);
   void rebind(in Name n, in Object obj)
     raises(NotFound, CannotProceed,
  InvalidName);

  void bind_context(in Name n,
      in NamingContext nc)raises ...;
  void rebind_context(in Name n,
      in NamingContext nc)raises...
```

Deleting Names

```
void unbind(in Name n)
  raises(NotFound, CannotProceed,
InvalidName);
```

Resolving Names

```
Object resolve(in Name n)
      raises(NotFound, CannotProceed,
 InvalidName);

 void list(in unsigned_long how_many,
           out BindingList bl,
           out BindingIterator bl);
```

Creating/Deleting Contexts

```
  NamingContext new_context();
  NamingContext bind_new_context(in Name n)
      raises(NotFound, AlreadyBound,
             CannotProceed, InvalidName);

  void destroy()raises(NotEmpty);
};
```

Exceptions

- Alle defined in interface NamingContext
- NotFound: NameComponent is not present
  - NotFoundReason why;
    - missing_node
    - not_context
    - not_object
- CannotProceed: Implementation cannot continue resolving
  - NamingContext ctx; Name rest_of_name
- InvalidName: Name does not follow implementation limitations, or is empty
- AlreadyBound: Name already exists
- NotEmpty: Context has contents

BindingIterator

```
module CosNaming{
   interface BindingIterator{
       boolean next_one(out Binding b);
       boolean next_n(in unsigned long how_many,
                      out BindingList bl);
       void destroy();
   };
};
```

- Implementations can delete iterators themselves, so OBJECT_NOT_EXIST may happen

INS: Interoperable Naming Service

- Standard command line arguments
- Shorter URLs for object references
  - Argument for string_to_object
  - Values of initial references on command line
- Extensions of the NamingContext interface

Object URLs

- corbaloc: Contact information (Host, port, object key)
  - corbaloc::ccm1/NameService
  - Default value for port: 2809
    - corbaloc::ccm1:2809/NameService
  - Default value for protocol
    - corbaloc:iiop:1.1@ccm1:2809/NameService
  - Protocol rir: resolve_initial_references
    - corbaloc:rir:/SecurityCurrent
  - Alternative contact addresses:
    - corbaloc::ccm1,pao:4567/NameService

Object URLs (2)

- corbaname: String version of name service entry
  - contact information and name separated by #
    - corbaname::ccm1#myctx/myobj
  - default object key: NameService
    - corbaname::ccm1/NameService#myctx/myobj
  - Combination with rir protocol:
    - corbaname:rir:#myctx/myobj