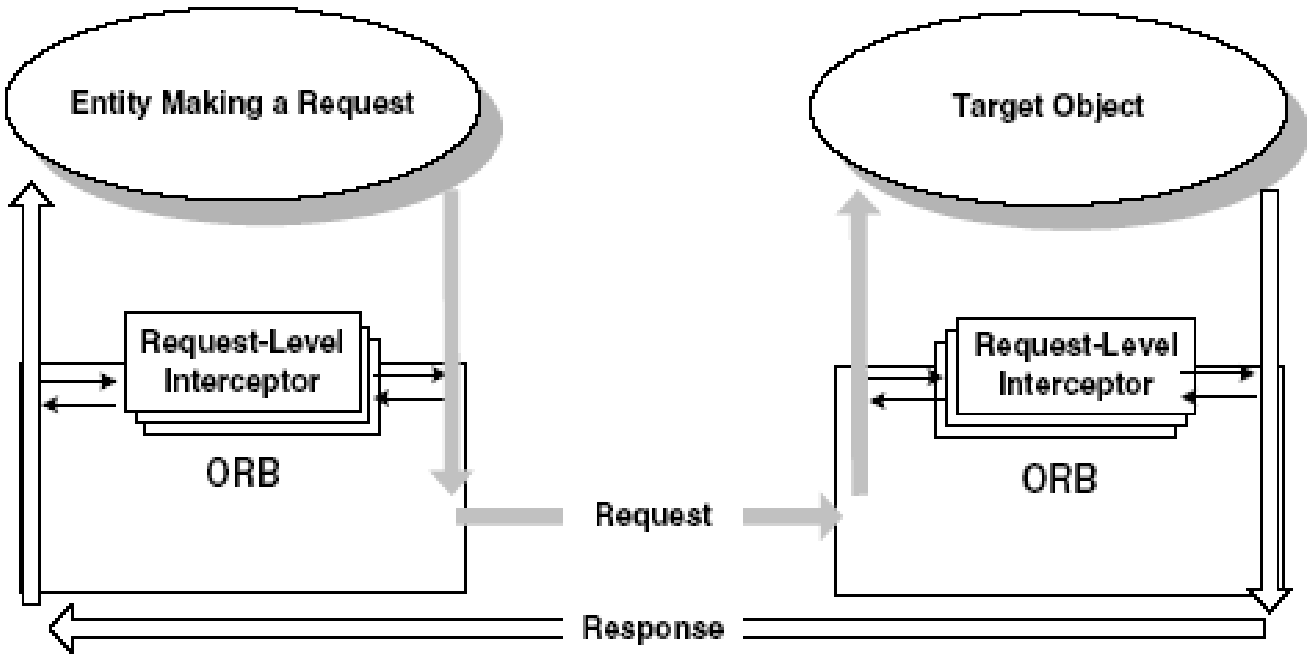
CORBA – Interceptors

Ralf Kahrl, Andreas Knauer, Christopher Kristes

Übersicht

- Architektur
- Verwendungsmöglichkeiten
- Problematiken
- Implementierung Client
- Implementierung Server
- Ausblick ‚Security‘

Architektur



Verwendungsmöglichkeiten

- Authorisierungsdaten auswerten
- Überwachungslogs erstellen
- Vor oder nach Operationsaufrufen Ereignishandler anstoßen
- Ver- und Entschlüsselung

Problematiken

- Interceptoren können nur vom ORB aufgerufen werden
- Endlosschleifen bei gegenseitigem Objektaufruf
- Systemobjekte damit keine dynamische Bindung
- Interceptoren ändern den gesamten ORB
- Keine selektive Verarbeitung

Interface ClientInterceptor

```
interface ClientInterceptor : Root {
  Status initialize_request (in LWRequest req,
                             in CORBA::Environment env);
  Status after_marshal (in LWRequest req,
                        in CORBA::Environment env);
  Status output_message (in CORBA::Buffer message,
                         in CORBA::Environment env);
  Status input_message (in CORBA::Buffer message,
                        in CORBA::Environment env);
  Status before_unmarshal (in LWRequest req,
                           in CORBA::Environment env);
  Status finish_request (in LWRequest req,
                         in CORBA::Environment env);
};
```

Interface ServerInterceptor

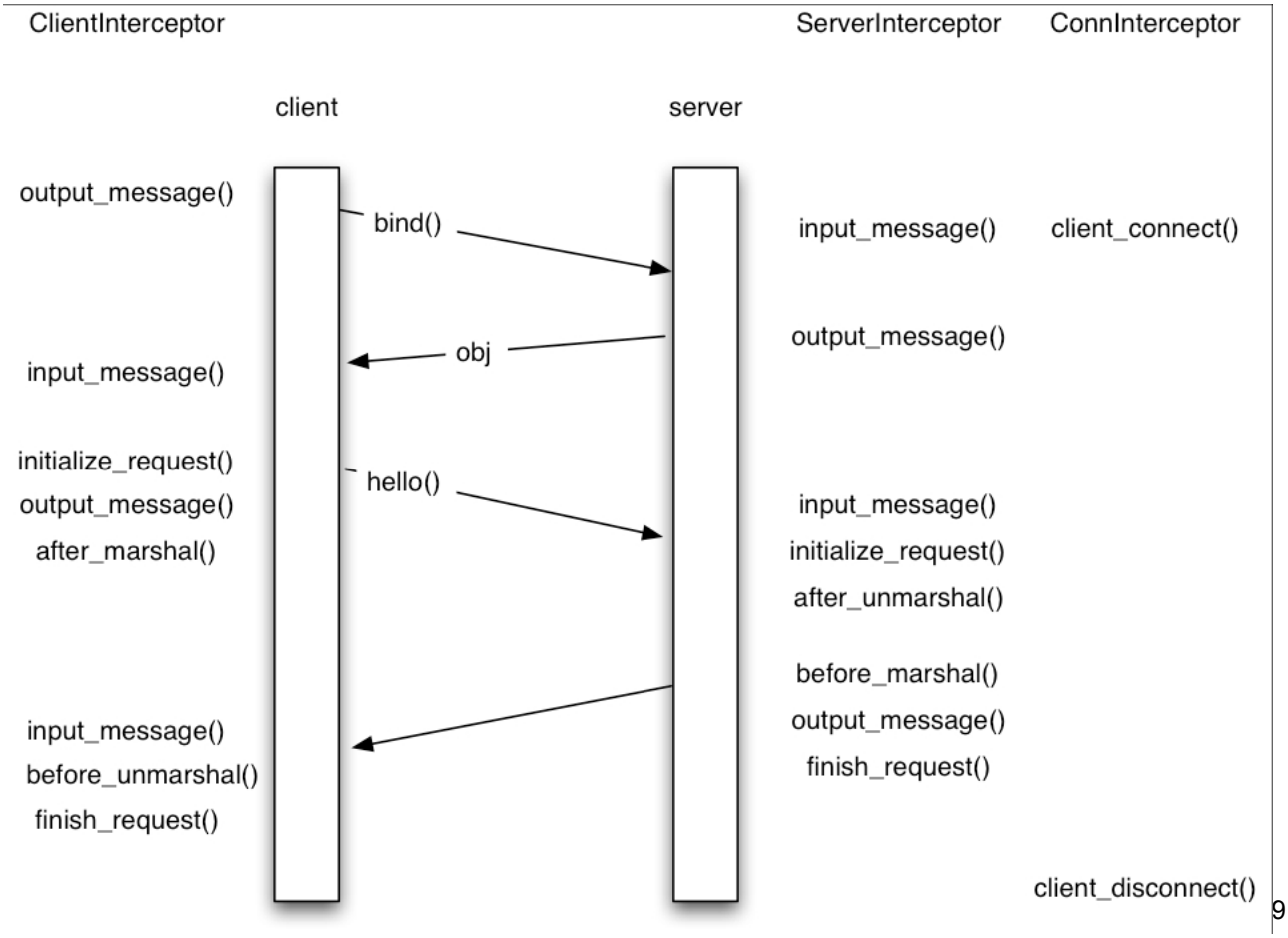
```
interface ServerInterceptor : Root {
   Status input_message (in CORBA::Buffer message,
                         in CORBA::Environment env);
   Status initialize_request (in LWServerRequest req,
                              in CORBA::Environment env);
   Status after_unmarshal (in LWServerRequest req,
                           in CORBA::Environment env);
   Status before_marshal (in LWServerRequest req,
                          in CORBA::Environment env);
   Status finish_request (in LWServerRequest req,
                          in CORBA::Environment env);
   Status output_message (in CORBA::Buffer message,
                          in CORBA::Environment env);
};
```


Interfaces weiterer Interceptoren

```
interface InitInterceptor : Root {
   Status initialize (in CORBA::ORB orb,
                      in CORBA::ORBId id,
                      inout ArgList arg_list);
};

interface BOAInterceptor : Root {
   Status bind (in string repoid, in CORBA::ORB::ObjectTag tag);
   Status restore (in Object obj);
   Status create (in Object obj);
};

interface ConnInterceptor : Root {
   Status client_connect (in string addr);
   Status client_disconnect (in string addr);
};
```



Client-Interceptor_impl

```
using namespace std;

class MyClientInterceptor : public Interceptor::ClientInterceptor {
public:
   Interceptor::Status initialize_request (Interceptor::LWRequest_ptr req,
   CORBA::Environment_ptr env)  {
         CORBA::String_var s = req->operation();
         cout << "client: init request for: " << s.in() << endl;
         return Interceptor::INVOKE_CONTINUE;
   }

   Interceptor::Status after_marshal (Interceptor::LWRequest_ptr req,
   CORBA::Environment_ptr env)  {
         CORBA::String_var s = req->operation();
         cout << "client: after marshal for: " << s.in() << endl;
         return Interceptor::INVOKE_CONTINUE;
   }
   Interceptor::Status output_message (CORBA::Buffer *buf, CORBA::Environment_ptr env)
   {
         cout << "client: output message (" << buf->wpos() - buf->rpos() <<
                  " bytes)" << endl;

         for (CORBA::ULong i = buf->rpos(); i < buf->wpos(); ++i)
             (buf->buffer())[i] ^= 0x37; // simple encryption ...
         return Interceptor::INVOKE_CONTINUE;
    }
...
```

Client-Interceptor_impl

```
Interceptor::Status input_message (CORBA::Buffer *buf, CORBA::Environment_ptr env) {
   cout << "client: input message (" << buf->wpos() - buf->rpos() << " bytes)" << endl;

   for (CORBA::ULong i = buf->rpos(); i < buf->wpos(); ++i)
       (buf->buffer())[i] ^= 0x37; // simple decryption ...
   return Interceptor::INVOKE_CONTINUE;
}

Interceptor::Status before_unmarshal (Interceptor::LWRequest_ptr req,
   CORBA::Environment_ptr env){
   CORBA::String_var s = req->operation();
   cout << "client: before unmarshal for: " << s.in() << endl;
   return Interceptor::INVOKE_CONTINUE;
}

Interceptor::Status finish_request (Interceptor::LWRequest_ptr req,
   CORBA::Environment_ptr env){
   CORBA::String_var s = req->operation();
   cout << "client: finish request for: " << s.in() << endl;
   return Interceptor::INVOKE_CONTINUE;
}
```


Client Interceptor Aktivierung

```
int main (int argc, char *argv[]) {
    MyClientInterceptor icept;
    icept.activate (0);

    CORBA::ORB_var orb = ...
    ...
    ...
}
```

Server Interceptor_impl

```
class MyServerInterceptor : public Interceptor::ServerInterceptor {
public:
    Interceptor::Status input_message (CORBA::Buffer *buf, CORBA::Environment_ptr
   env)  {
   cout << "server: input message (" << buf->wpos() - buf->rpos() << " bytes)" <<
   endl;

   for (CORBA::ULong i = buf->rpos(); i < buf->wpos(); ++i)
       (buf->buffer())[i] ^= 0x37;         // simple decryption ...

   return Interceptor::INVOKE_CONTINUE;
    }

    Interceptor::Status initialize_request (Interceptor::LWServerRequest_ptr req,
   CORBA::Environment_ptr env) {

   CORBA::String_var s = req->operation();
   cout << "server: init request for: " << s.in() << endl;
   return Interceptor::INVOKE_CONTINUE;
    }
…
```


Server Interceptor_impl

```
Interceptor::Status after_unmarshal (Interceptor::LWServerRequest_ptr req,
   CORBA::Environment_ptr env) {
   CORBA::String_var s = req->operation();
   cout << "server: after unmarshal for: " << s.in() << endl;
   return Interceptor::INVOKE_CONTINUE;
}

Interceptor::Status before_marshal (Interceptor::LWServerRequest_ptr req,
   CORBA::Environment_ptr env) {
   CORBA::String_var s = req->operation();
   cout << "server: before marshal for: " << s.in() << endl;
   return Interceptor::INVOKE_CONTINUE;
}
 Interceptor::Status finish_request (Interceptor::LWServerRequest_ptr req,
   CORBA::Environment_ptr env) {
   CORBA::String_var s = req->operation();
   cout << "server: finish request for: " << s.in() << endl;
   return Interceptor::INVOKE_CONTINUE;
}
```

Server Interceptor_impl

```
Interceptor::Status output_message (CORBA::Buffer *buf,
   CORBA::Environment_ptr env) {
   cout << "server: output message ("
        << buf->wpos() - buf->rpos() << " bytes)" << endl;

   for (CORBA::ULong i = buf->rpos(); i < buf->wpos(); ++i)
        (buf->buffer())[i] ^= 0x37; // simple encryption ...

  return Interceptor::INVOKE_CONTINUE;
}
```


Server ConnectionInterceptor

```
class MyConnInterceptor : public Interceptor::ConnInterceptor {
public:
    Interceptor::Status client_connect (const char *addr){
       cout << "server: connect from: " << addr << endl;
       return Interceptor::INVOKE_CONTINUE;
    }

    Interceptor::Status client_disconnect (const char *addr){
       cout << "server: disconnect from: " << addr << endl;
       return Interceptor::INVOKE_CONTINUE;
    }
};
```

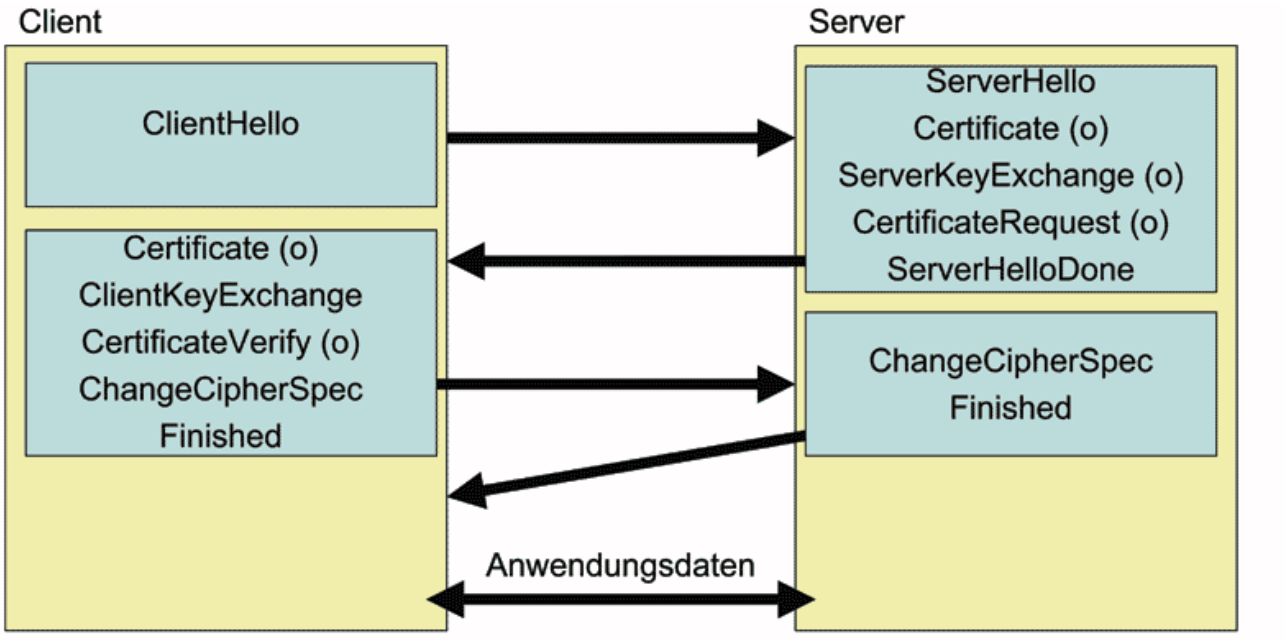
Server Interceptor Aktivierung

```
int main (int argc, char *argv[]) {
    MyServerInterceptor sv_icept;
    sv_icept.activate (0);

    MyConnInterceptor orb_icept;
    orb_icept.activate (0);

    ...
}
```


ssl-Verbindung



ssl Interceptor Mico

```
class AccessChecker : public Interceptor::ServerInterceptor {
public:
Interceptor::Status
initialize_request (Interceptor::LWServerRequest_ptr req,
                        CORBA::Environment_ptr env)   {
        CORBA::Object_var obj = req->target ();
        CORBA::Object_var o =
            the_orb->resolve_initial_references ("PrincipalCurrent");
        CORBA::PrincipalCurrent_var pc = CORBA::PrincipalCurrent::_narrow (o);
        CORBA::Principal_var p = pc->get_principal();

        CORBA::Any_var a = p->get_property ("ssl-x509-subject:CN");
        const char *name;

        if (!(a >>= name) || strcmp (name, "Roemer")) {
            // permission denied ...
            env->exception (new CORBA::NO_PERMISSION);
            return Interceptor::INVOKE_ABORT;
        }
        // ok ...
        return Interceptor::INVOKE_CONTINUE;
    }
};
```


ssl Mico Beispielaufruf

- ./server -ORBIIOPAddr $ADDR -ORBSSLcert s_cert.pem -ORBSSLkey s_key.pem -ORBSSLverify 0
- ./client `cat sec_hello.ior` -ORBSSLcert c_cert.pem -ORBSSLkey c_key.pem -ORBSSLverify 0