Java GSS-API: Concrete Classes or Interfaces?

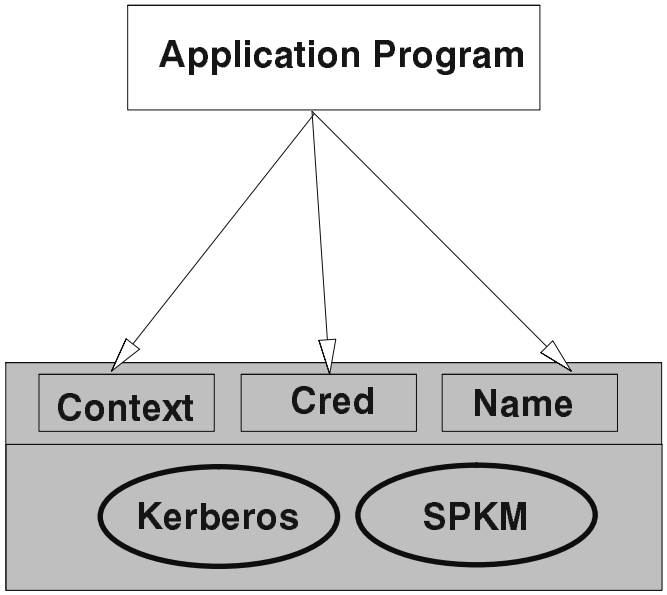
Mayank Upadhyay
Sun Microsystems
mdu@eng.sun.com

CAT Working Group
44th IETF, 1999

Design Goals

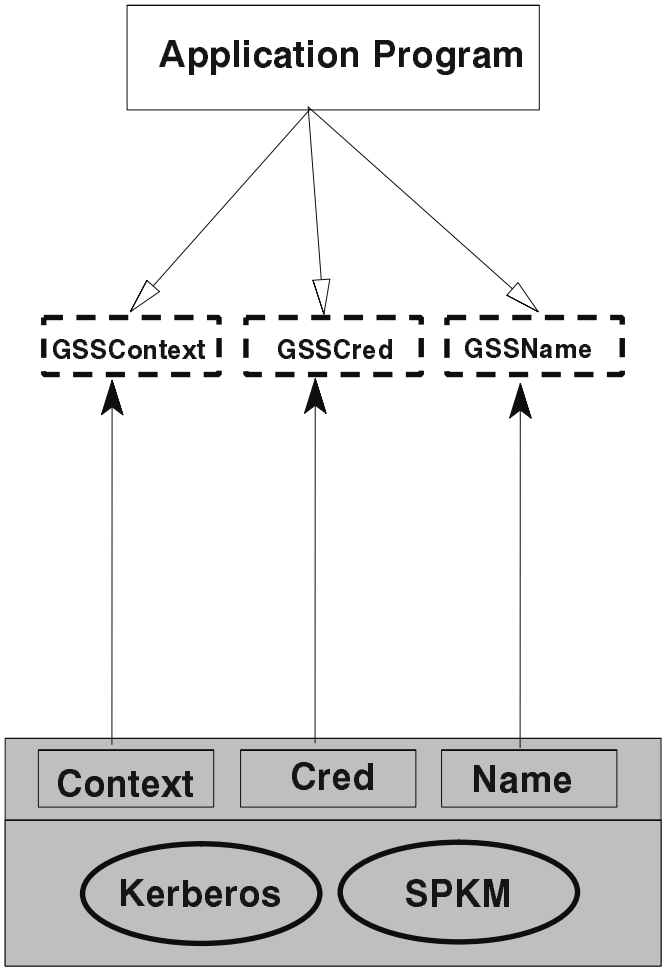
- Multi mechanism support within one program lifetime
  - NFS client-server environment with many mechanisms
- Application programmer ease of use
  - Object oriented design
  - Simple object creation
  - Overloading of methods to assume default values

Concrete Classes Approach

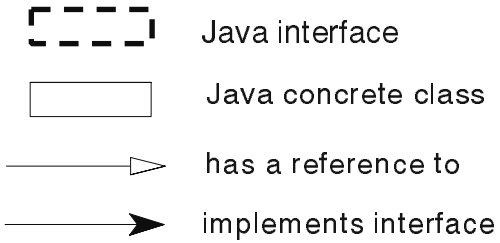


GSS-API Implementation

Interfaces Approach



GSS-API Implementation



A Simple Context Initiator using Concrete Classes

```
import org.ietf.JGSS.*;
// package consists of concrete classes

...
...
    OID mechOID         = new OID("1.2.840.113554.1.2.2");

    GSSName peer        = new GSSName("someservice",
                            GSSName.HOSTBASED_SERVICE);

    GSSCredential cred = new GSSCredential(null /* Default Name */,
                             GSSCredential.INDEFINITE,
                             mechOID,
                             GSSCredential.INITIATE_ONLY);

    GSSContext context = new GSSContext(peer, mechOID, cred,
                             GSSContext.INDEFINITE);


    ...
    ...
    // context.init() loop etc. follows
```

- Desired GSS-API classes picked up via CLASSPATH
- Only one implementation of GSS-API can be instantiated in one lifetime of application
- Consistent with model used by Java Platform core API and its various implementations in JDK's and SDK's

A Simple Context Initiator using Interfaces

```
import org.ietf.JGSS.*;
// package consists of interfaces
...
...

     String className   = System.getProperty("org_ietf_JGSS_Factory);
     // Which class claims to be the factory class for the
     // GSS implementation present on this system?

     GSSFactory factory = Class.forName(className).newInstance();

     OID mechOID        = factory.getOID("1.2.840.113554.1.2.2");

     GSSName peer       = factory.getName("someservice",
                             GSSName.HOSTBASED_SERVICE);

     GSSCredential cred = factory.getCredential(null,
                             GSSCredential.INDEFINITE,
                             mechOID, GSSCredential.INITIATE_ONLY);

     GSSContext context = factory.getContext(peer, mechOID, cred,
                             GSSContext.INDEFINITE);

     ...
     ...


    // context.init() loop etc. follows
```

Using Just Interfaces will Allow Multiple GSS Implementations in One Application Lifetime

```
// Say implementation #1 supports mechanisms #1a and #1b

GSSFactory factory1 = Class.forName(className1).newInstance();
OID mech1a = factory1.getOID("1.2.840.113554.1.2.2");
OID mech1b = factory1.getOID("1.3....");
GSSName myName1 = factory1.getName("joe", GSSName.USER_NAME);
GSSCredential cred1 = factory1.getCredential(myName1,
           GSSCredential.INDEFINITE, mech1a, GSSCredential.INITIATE_ONLY);
cred1.add(myName1, GSSCredential.INDEFINITE, mech1b,
          GSSCredential.INITIATE_ONLY);

// Say implementation #2 supports mechanism #2

GSSFactory factory2 = Class.forName(className2).newInstance();
OID mech2 = factory2.getOID("1.4.....");
GSSName myName2 = factory2.getName("joe", GSSName.USER_NAME);
GSSCredential cred2 = factory2.getCredential(myName2,
           GSSCredential.INDEFINITE, mech2, GSSCredential.INITIATE_ONLY);

// After a bunch of calls the objects start getting mixed around and BOOM!

cred2.add(myName1, GSSCredential.INDEFINITE, mech1a,
          GSSCredential.INITIATE_ONLY);

context = factory1.getContext(myName2, mech1a, cred1,
                    GSSContext.INDEFINITE);
```

CLASSPATH's and Concrete Classes: The Applet Issue

Applet Developer	**Web Browser**
Has org.ietf.JGSS classes with a custom/proprietary mechanism	Has org.ietf.JGSS with some fixed set of mechanisms

Problem:

Applet developer ships his/her org.IETF.JGSS package across the wire with the applet code but which one does the browser end up using?

Solution:

IDEALLY: Ship the custom mechanism across the wire and let it plug under the browser's GSS implementation.

INTERIM (until an SPI is defined):

Applet developer calls the custom GSS implementation package com.xyz.JGSS
Applet code imports com.xyz.JGSS