Introduction to DCE/OODCE

Laks Prabhala

ECS Developers' Workshop
30 May 1995

Agenda

Introduction to Developers Workshop	**08:30 - 08:45**
DCE/OODCE Introduction	**08:45 - 09:30**
Interface Definition Language	**09:30 - 10:00**
• use of pointers, structures etc..	
Writing a ECS Server & Client	**10:15 - 11:15**
• registration; event logging; exceptions; threads; mss interface, # of instances etc..	
• Event Logging	
Interfacing with MSS	**11:15 - 11:45**
Special Topics	**11:45 - 12:30**
• Bulk Transfers - Common Facilities/ RPC Pipes	
• Customizing a Directory/Namespace - Production/Testing/IV&V etc..	
• Error Handling/Exception Handling	
• Gateway Status	
Distributed Object Paradigm - Objects, Objects & Objects	**1:30 - COB**

Writing a Simple Distributed Application/Specialized Topics [Code Walkthrough]

- Threads - basics, asynchronous/callbacks/locks etc..
- Security - Authorization ACLs, Key Mgmt, RefMons, Policy
- Writing Iterators in distributed objects
- Customizing the Client Stub code as generated by OODCE
- Factories/Manager Objects and Dynamic Registration in OODCE
- Distributing the Application Objects - Strategy and guidelines
- Message Passing Status
- Using the Directory Services as a global db for application information
- Using Security for non-rpc applications

Why bother...

Because Distributed Computing [objects, Client/Server] has

- **Multiple Operating Modes**
- **Simultaneous execution/Parallel executions [local and network]**
- **Many possible states**
- **Complex processing flows**
- **Security enforcements difficult**
- **Performance estimations difficult**
- **Complex design tradeoffs**
- **Testing and debugging difficult**
- **Interoperability - must run on multiple platforms and configurations**
- **High level of resource availability [replications; backups etc..]**

Think and Code “Open & Distributed”

DISTRIBUTION

- ➢ Do not use file paths, network addresses as constants
- ➢ Minimize data transfer ex: pointer use
- ➢ Use RPC to distribute complex computation to faster CPU
- ➢ Be ready to process exceptions due to network errors

SCALABILITY

- ➢ Thing about 2002
- ➢ More servers, more DAACs, more clients, many users
- ➢ LAN - WAN switch overs
- ➢ RMA - Replicated servers

PORTABILITY

- ➢ Use DCE RPC (IDL) defined data types
- ➢ Use ANSI/POSIX standards
- ➢ Use ECS Types
- ➢ DOCUMENT VERY WELL for conditionalizing specific platforms

Basic DCE Configuration Components

DCE User Machine Configuration	DCE Admin Machine Configuration	DCE Server Machine Configuration
(DFS Server)	DFS Admin Client	DFS Server [user + admin]
DFS Client	Security Service Admin Client	DTS Server [user + admin]
Security Service Client	DTS Admin Client	CDS Server [user + admin]
DTS Client	Directory Service Admin Client	Security Server [user + admin]
Directory Service Client	DCE User	DCE User
DCE RPC		
DCE Threads		

Basic DCE Development and Admin Components

UUIDGEN

idl compiler

DCE + OODCE Include Files

DCE + OODCE Libraries

rpccp*

cdscp*

acl_edit

dtscp*

rgy_edit*

sec_admin*

dts tools

passwd_import

* dcecp replaces “almost-all” in DCE 1.1

DCE & OODCE

What is DCE ?

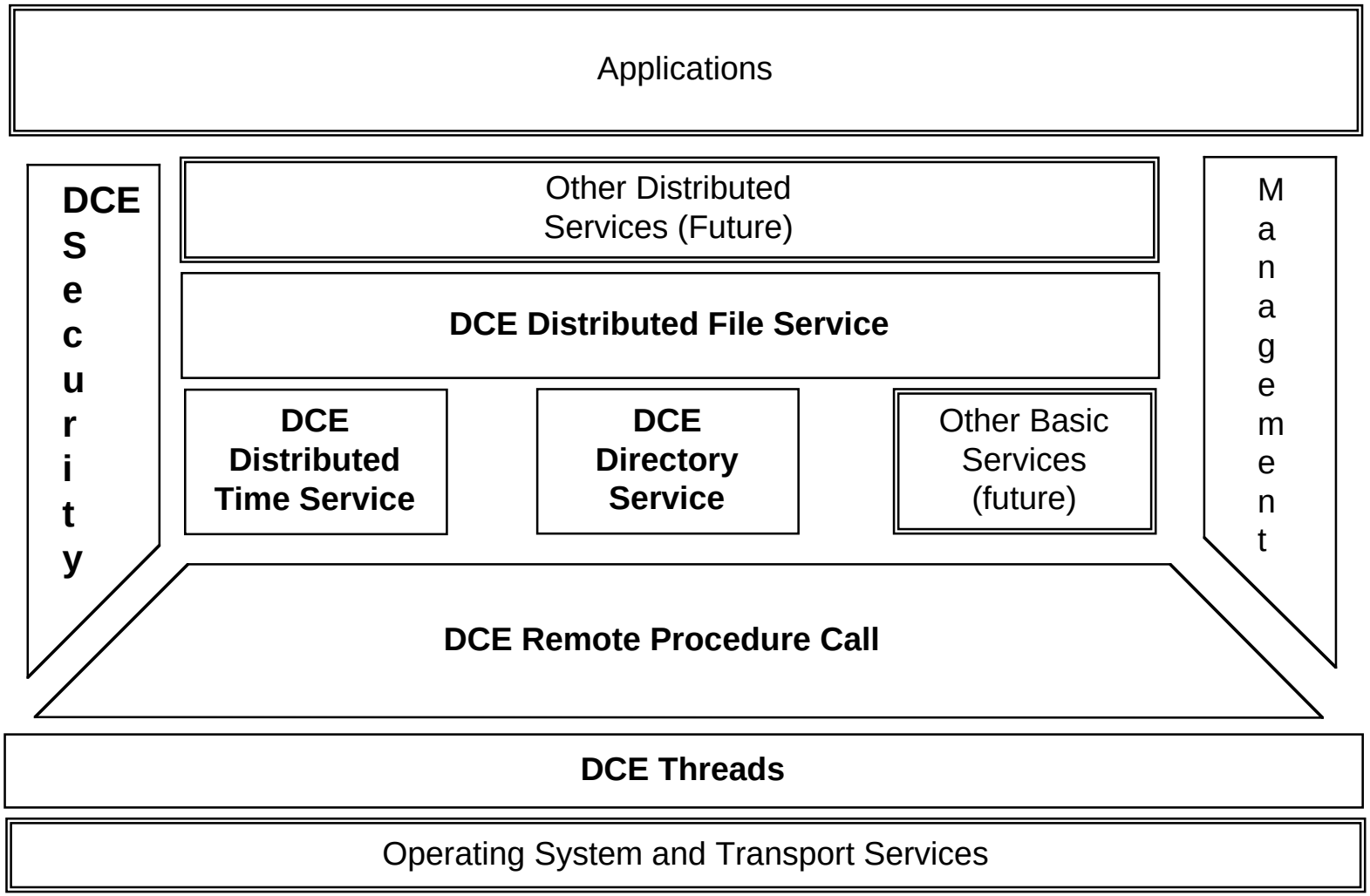
- DCE is a heterogeneous and Interoperable
  - environment
  - product
  - standard
  - current released version from OSF is DCE 1.0.3a

What is OODCE ?

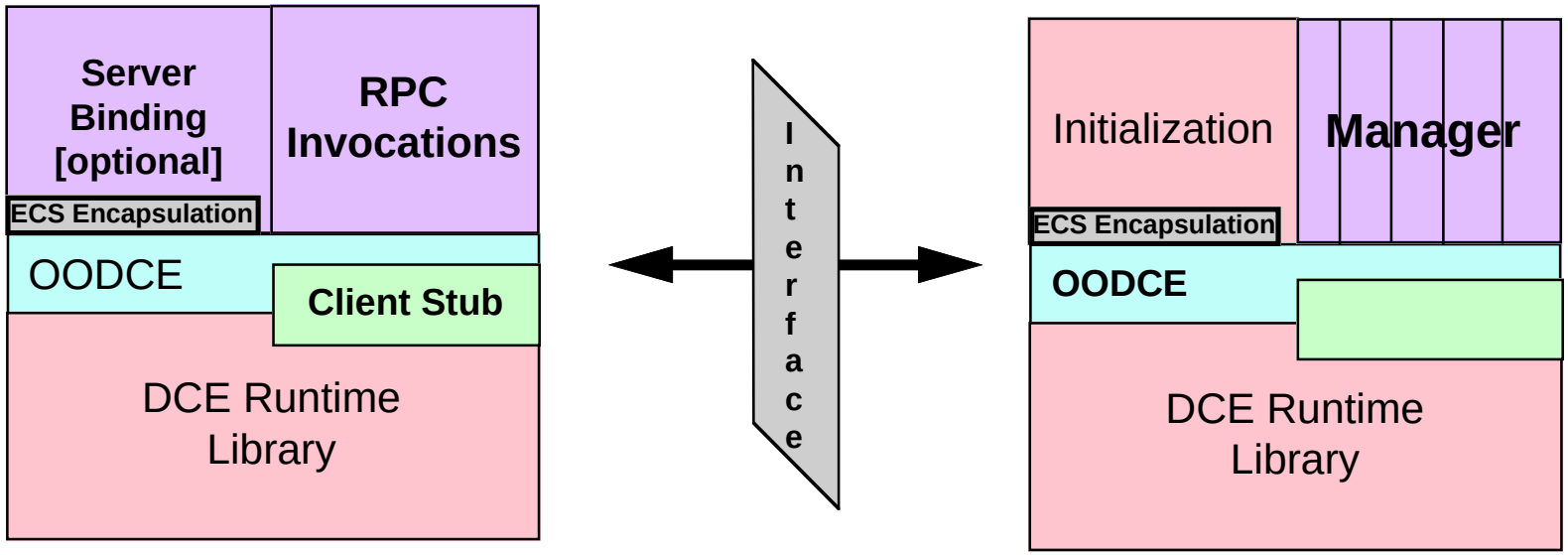
  - It is a development library to access DCE Services
  - Does not change the wire protocol or DCE server implementations
  - eases development of DCE applications in C++
  - Product from Hewlett Packard currently based on DCE 1.0.3

istributed Computing Environment

DCE Architecture



Distributed Application Components (1)



Distributed Application Components (2)

Interface

- **Contract of Client and a Server**
- **Client invokes the interface - Server guarantees to accept and process those invocations, returns results or raises exceptions**

Manager

- **contains the actual routines of RPC implementations**

Stubs

- **Marshalling and Unmarshalling**
  - **translating RPC's parameter and results to and from a machines native format into a Common format transfer across the wire**

ECS Encapsulation

- **adding new functionality to DCE/OODCE ex: MSS interfaces/Logging**

Runtime Library

- **interfaces all other DCE Services [security, directory etc..]**

RPC

Why RPCs ?

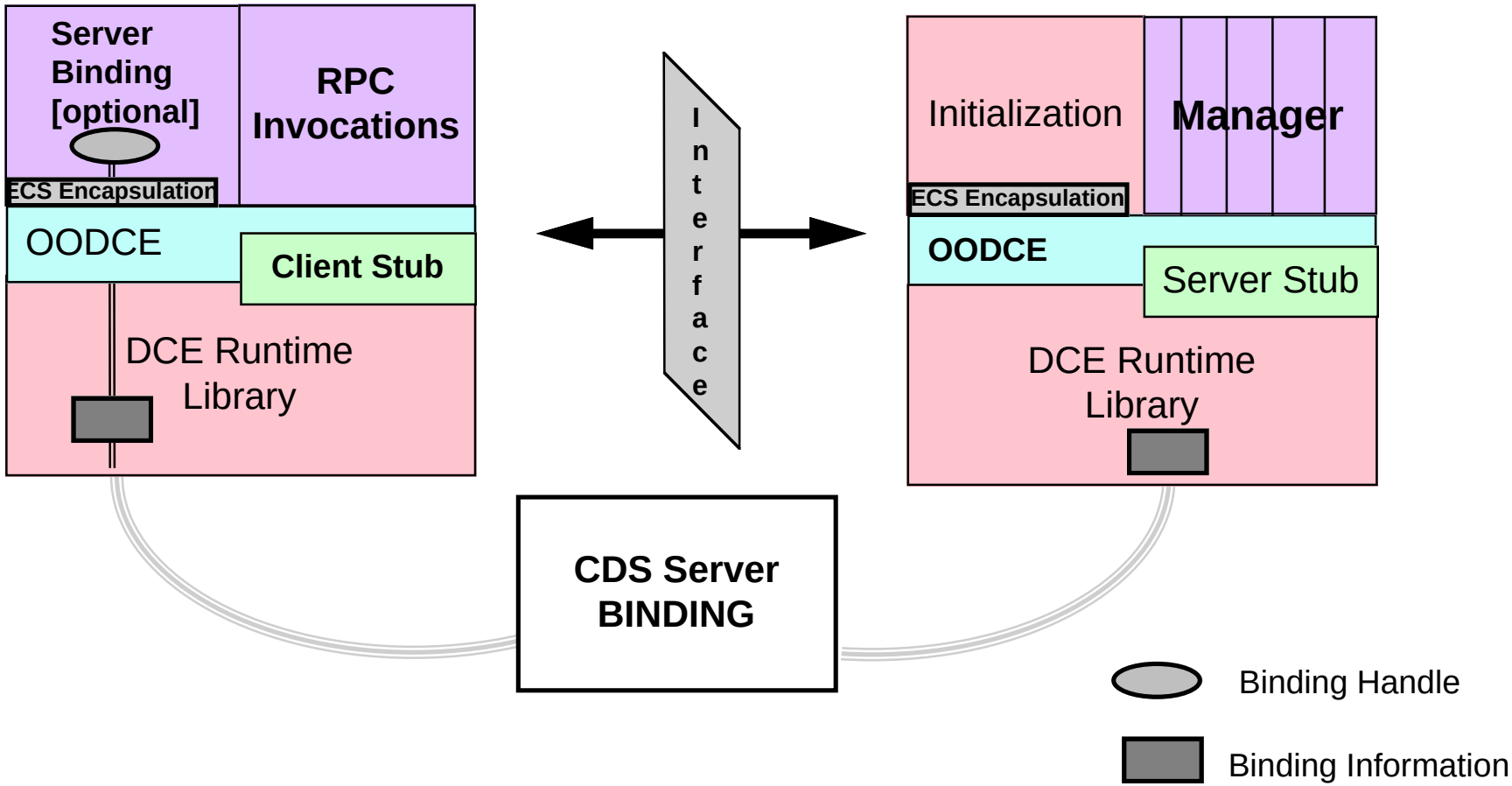
- **Transport Independence**
- **Data Transformation**
- **Secure Communications**

RPC (2)

RPC Call Semantics

- **at-most-once (default)**
  - **non-idemponent ex: deposit(..)**
- **Idempotent**
  - **add (x, y)**
- **Maybe [implicitly idempotent]**
  - **no return values, not guaranteed**
- **Broadcast [implicitly idempotent]**
  - **only on a LAN**

Directory Service (1)



Directory Service (2)

Servers export their binding information to CDS

- **[optional] global replicated data store using XDS/XOM**

Client import Servers binding information from CDS

Object UUID
Protocol Sequence
Host Network Address
Endpoint
Network Options
Client/Server Information

```
rpccp> show entry /.../csmscell.hitc.com/ep3/auth_rpc_server

>>> no matching objects found

binding information
<interface id>  af508fe4-5625-11cd-8c31-0800009701903, 1.0
<string binding> ncacn_ip_tcp : 155.157.31.90 [ ]
<string binding> ncacn_ip_udp : 155.157.102.23 [ ]
```

A typical CDS lookup

Will be supplied at a later date.

Directory Service (CDS + GDS)

Consistent Global Namespace

- **Directories, objects, soft links**
- **Integrated with Security Service**

Cell Directory Service (CDS)

- **Replication**
- **Caching**

Global Directory Service (GDS)

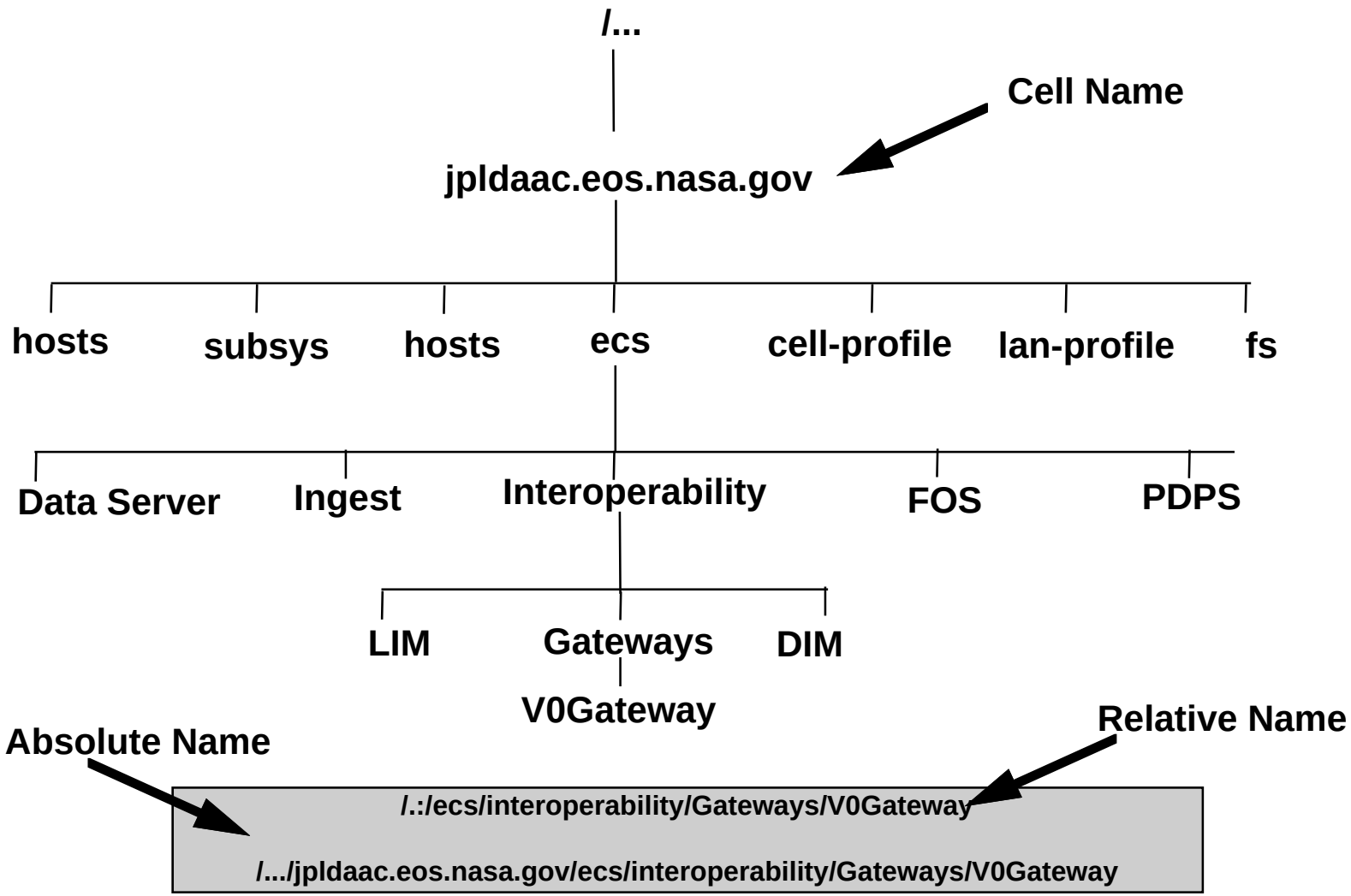
- **X.500 or DNS**
- **Accessed via CDS**

RPC Name Service Interface (NSI)

- **Servers, Groups, Profile entries**

XDS & XOM extensions/APIs

Typical DAAC Cell Namespace



Security Service

Authentication

- **Kerberos (version 5)**
- **DES Encryption**
- **Password, tickets and sessions keys**

Authorization

- **Priviledge Attribute Certificate (PAC)**
- **Principals, Groups, Organizations**

Data Integrity Verification

Data Privacy Verification

User Management

ACL Management

Security Protection Levels

Symbol	Description
rpc_c_protect_level_default	Use the default protection level
rpc_c_protect_level_none	Do not provide any protection
rpc_c_protect_level_connect	Verify identities when the client first connects
rpc_c_protect_level_call	Verify identities on each call to the server *
rpc_c_protect_level_pkt	Verify identities whenever data is sent
rpc_c_protect_level_pkt_integrity	Insure that data sent has not been modified
rpc_c_protect_level_pkt_privacy	Encrypt the data that is sent

* Only applies to connectionless protocols. If used with connection-based protocol (eg ncan_ip_tcp) it will automatically be converted to rpc_c_protect_level_pkt

Authentication Services

Type	
rpc_c_authn_none	
rpc_c_authn_dce_secret	
rpc_c_authn_dce_public	Use public key authentication (DCE 1.2)

Authorization Services

Type	
rpc_c_authz_none	
rpc_c_authz_name	
	Authorization based on PAC

ECS Login Service

Subsystems (FUI group, V0/V1 development group) develop the GUI

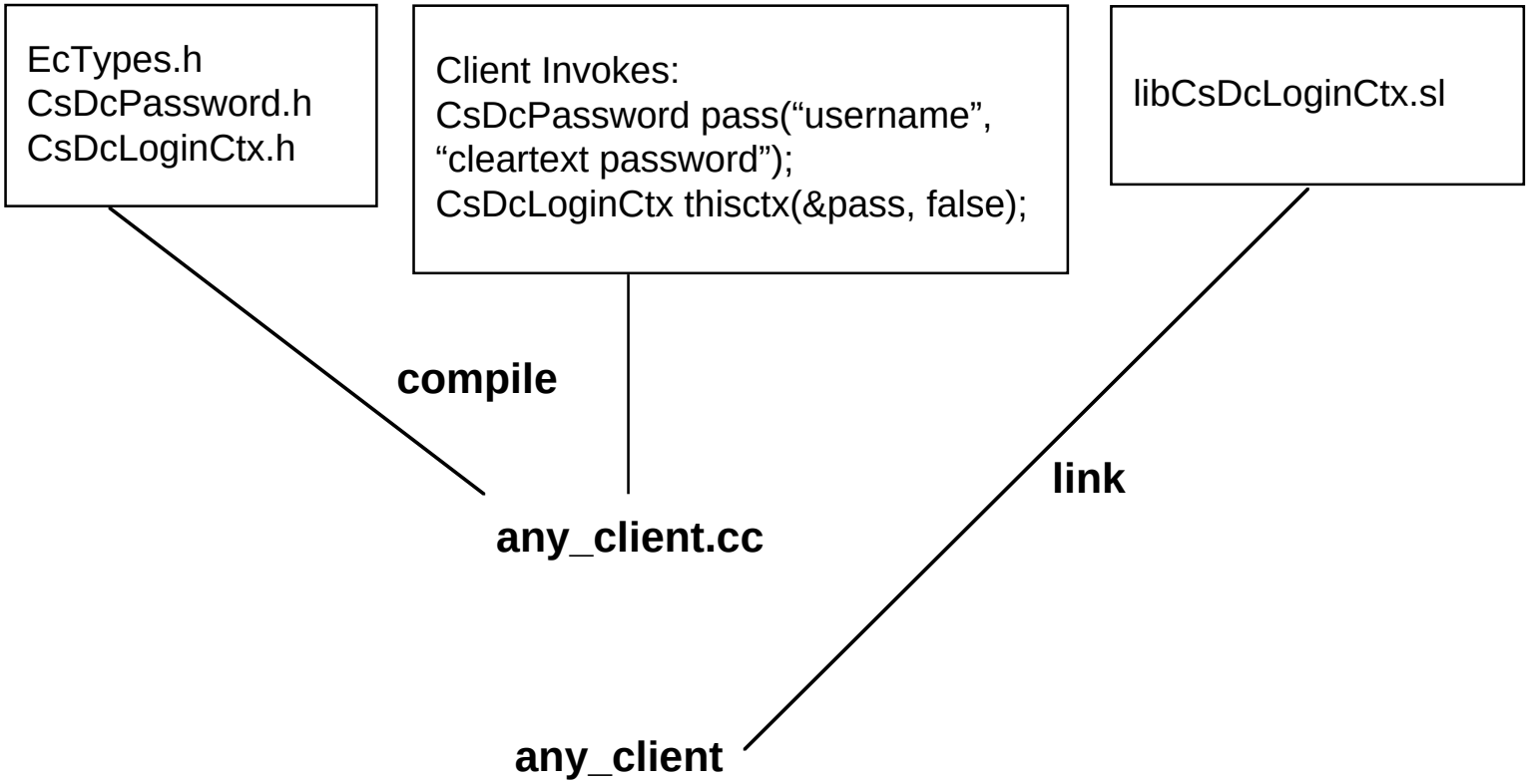
Underlying APIs for User login will be provided by CSS

- **2 classes defined currently**
  - **CsDcPassword & CsDcLoginCtx**

[optional] Additional Classes/APIs provided for

- **changing password**
- **listing principals groups and organization information**
- **Tools provided to create/modify/delete and list users/groups and organizations**

User Authentication



Threads (1)

Pthreads with extensions (POSIX.4a Draft 4)

User Level (non-kernel)

Explicit thread creation and termination

Syncronization Methods

- mutexes (locks)
- condition variables
- join

Automatic use of threads by servers

Jacket routines

Threads (2)

Why use threads ?

- **Servers can handle concurrent requests from multiple clients**
  - **ex: LIM getting multiple V0/V1 client browse requests**

    Ingest server getting multiple DAN requests

- **Client can do productive work while waiting for RPC responses from Servers**
  - **ex: database work thread, user interface thread & rpc thread**

Distributed Time Service (DTS)

Distributed Clock Synchronization

- **ex: all the event logs in ECS will be synchronized**

Secure Protocol

Coordinated Universal Time (UTC)

Inaccuracy Factor

- **can adjust for known inaccuracies ex: using hardware time sources**

Support for external time provider

DCE 1.1

Administration	• dcecp - replaces “almost all” ; rgy_edit, sec_admin, rpccp, cdscp, dtscp • cell aliases • centralized administration - secure remote host & server administration
Ease of Programming	• IDL compiler stub size improvement; pointers, array, I18N • ACL library • Backing Store library - store and retrieve static server data (objects) • dced server registration library - local configuration data

Security	• Extended Registry Attributes
	• Delegation - support for authenticated Intermediaries in operation
	  • impersonation and traced
	• Hierarchical Cells
	  • transitive trust

Security	• Auditing • Pre-authentication • Password Management • GSS API - allows use of DCE security in a non-RPC application
Other	• DTS remote administration • GDS uses GSS API, Is thread safe • DFS/NFS gateway - NFS access to DFS - Gateway Concept

DCE 1.2

Administration	• SNMP support • DCE Managed Object definition • DCE SNMP MIB • DCE SNMP Agent
Administration	• Centralized multicell administration • support of hierarchical namespace policies • active user monitoring
Ease of Programming	• X/Open Federated Naming (XFN) • IDL C++ Support • Single threaded client
Coexistence with Legacy Technologies	• Netware co-existence • ONC co-existence • Kerberos V5 support • public key support • multiple security environment support
Scalability	• small clients • improvements to security & machines - 100,000 users & hosts • DFS improvements

More Info

http://www.osf.org/dce/index.html

http://www.info.pilgrim.umass.edu

http://www.citi.umich.edu

or

see me for joining the WADCUG (Washington Area Distributed Computing Users Group)