EBU6501 - Middleware

Week 3, Day 1: Security Concepts for Middleware and Web Vulnerabilities

Gokop Goteng & Ethan Lau

Original version created by: Dr. Gokop Goteng

Lecture Aim and Outcome

- Aim
  - How to identify and prevent security threats and vulnerabilities in middleware and web-based applications
- Outcome
  - At the end of this lecture students should be able to:
    - Know the security threats in middleware and web applications
    - Know how to implement preventive measures against security threats
    - Know the security features of different middleware

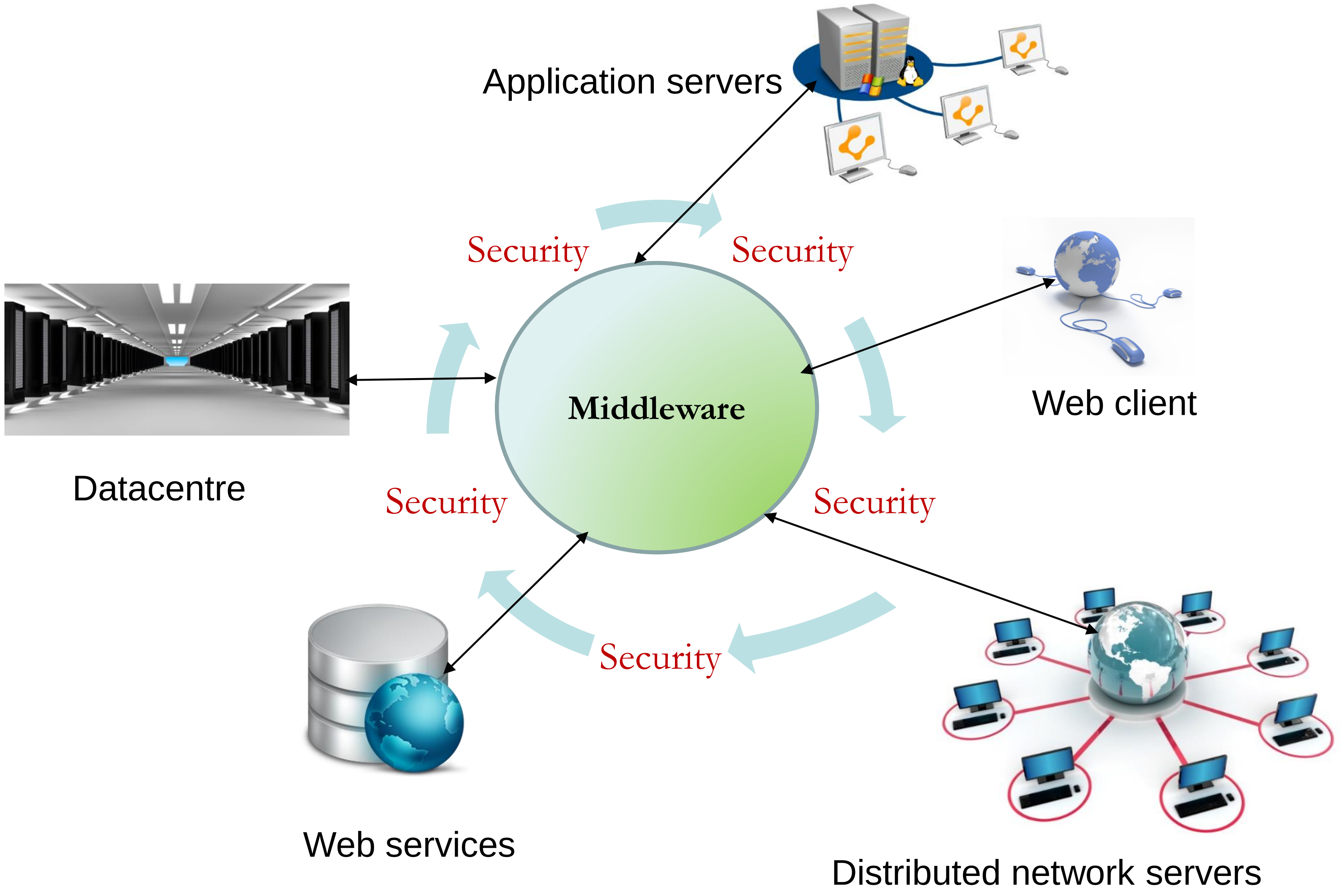
Lecture Outline

- **Security Concepts for Middleware**
- **Web Services Security Concepts**
- **Layers of Security**
- **Web Security Vulnerabilities**
- Case study: Security Implementations in Middleware
  - Globus
  - CREAM-CE

Middleware (recap)

- Middleware - a software layer residing on top of the operating system that **connects** different software components or applications.
- Provides **interoperability** and other services like the distribution of functionality, scalability, load balancing and fault tolerance.
- Functionalities of middleware (three general categories): **application-specific, information-exchange, management and support**.
- Application-specific middleware delivers services - distributed-database services, distributed transaction processing, and specialised services for mobile computing and multimedia.
- Information-exchange middleware - information management.
- The management and support - communicating with servers, manage security, handle failures, and monitor performance.

Security Concepts for Middleware



Security concepts:

1. Authentication mechanisms and credential management
2. Authorization and access control management
3. Shared data security and integrity
4. Secure one-to-one and group communication
5. Heterogeneous security/environment requirements support
6. Secure mobility management
7. Capability to operate in devices with low resources
8. Automatic configuration and management of these facilities.

Web Services Security Concepts

- **Loosely coupled** connectivity
  - Using http (hypertext transport protocol)
  - Multiple clients and servers interact independently
  - Distributed connections
- Methods of securing web services:
  - **Authentication**
  - **Authorisation**
  - **Confidentiality**
  - **Integrity**

Authentication

- Ensuring that it is the same person who she/he claims to be
- How?
  - Something one has
    - Credentials issued by a trusted authority such as
      - Smart card
  - Something one knows
    - Password.
  - Something one is
    - Biometric information (fingerprint)
- A strong authentication process consists of at least two of the above
  - For example having an ATM card (something you have) and entering a PIN (something you know)

Authorisation

- Access control
  - Granting access to specific resources based on an authenticated user's entitlements.
  - Entitlements are defined by one or several attributes.
  - An attribute is the **property or characteristic** of a **user**
    - Admin role, quest role, authorisation request, etc

Confidentiality

- **Privacy**
- Keeping **information secretive**.
  - Treat web service request, email, identity of the sending and receiving parties in a confidential manner.
  - To achieve **confidentiality and privacy**
    - **Encrypt** the content of a message
    - Do not reveal sending and receiving parties' identities
    - Use public key infrastructures (PKI) for encryption

Integrity

- Message in transit **should not be altered**
  - Sender should digitally sign the message.
  - A digital signature is used to validate the signature.
  - The timestamp in the signature prevents anyone from replaying this message after the expiration.
  - Exchanging security tokens in a trusted environment

Layers of Security

- **Transport-layer security**
  - Secure Socket Layer (SSL), also known as Transport Layer Security (TLS):
    - **Authentication** between communicating two trusted parties
    - **Confidentiality** through data encryption
    - Message integrity by checking that the data is not corrupted
    - **Secure key exchange between client and server**.
- **Application-layer security**
  - Application-level security complements **transport-level security**.
  - Application-level security is based **on XML frameworks** defining confidentiality, integrity, authenticity; message structure; trust management and federation.
  - **Data confidentiality** is implemented **by XML Encryption**.
    - XML Encryption defines how digital content is encrypted and decrypted, how the encryption key information is passed to a recipient, and how encrypted data is identified to facilitate decryption.
  - **Data integrity and authenticity** are implemented by **XML Signature**.
    - XML Signature binds the sender's identity (or "signing entity") to an XML document. Signing and signature verification can be done using asymmetric or symmetric keys.
- **Middleware-layer security**
  - Middleware layer security ensures that the **communicating security layers are secure**
  - **Single Sign-On (SSO) systems** are used for authentication across the layers
  - Certificate-based SSO are common in middleware security systems
  - Virtual organisation membership services (VOMS) are used fore authentication/authorisation for different users belonging to different organisations

Web Security Vulnerabilities

- Web security Vulnerabilities are areas of weakness in web security that **hackers or intruders exploit /** access to systems
- **Vulnerabilities:**
  - **Injection flaws**
    - Injection flaws result from **failure to filter** untrusted input.
    - It can happen when you pass unfiltered data to the SQL server (SQL injection), to the Lightweight Directory Access Protocol (LDAP) server (LDAPInjection), etc.
    - The attacker can "inject" commands to these entities, resulting in loss of data and hijacking clients' browsers.
- **Prevention:**
- Adopting **highly skilful programming** and **encryption techniques** plus vigorous testing procedures
- **Updating browsers** regularly

Web Security Vulnerabilities

- **Vulnerabilities:**
  - **Broken Authentication**
    - Password that is not encrypted
    - URL that exposes the session ID
    - Prevention:
      - Use a tested framework (e.g. J2EE) or implement your code to prevent this happening
  - **Cross Site Scripting**
    - Simple input on a form that contains malicious links
    - Posting cookies to hackers
    - Prevention
      - Do a thorough data cleaning and sanitisation for all inputs
      - Do not return HTML tags to the client
  - **Insecure Direct Object Reference**
    - Resetting passwords from an insecure environment
    - Exposing codes during download to unauthorised users
    - Prevention
      - Secure source codes and password resetting environment
      - Virtual key-boards usage

Web Security Vulnerabilities

- Vulnerabilities:
  - **Security Misconfigurations**
    - Using default passwords and keys on production systems
      - MySQL has a default username and password
    - Using outdated applications
    - Prevention
      - Automate security configurations
  - **Sensitive data exposure**
    - Not using SSL (Secure Socket Layer) in Tomcat security tag of deployment descriptor
    - Prevention
      - Enforce confidentiality and data integrity security features
      - Use SSL and encryption applications
  - **Problem with access control level**
    - Failure to implement correct authorisation system
    - Prevention
      - Automate authorisation system
      - Ensure authorisation is always enforced on the server side

Web Security Vulnerabilities

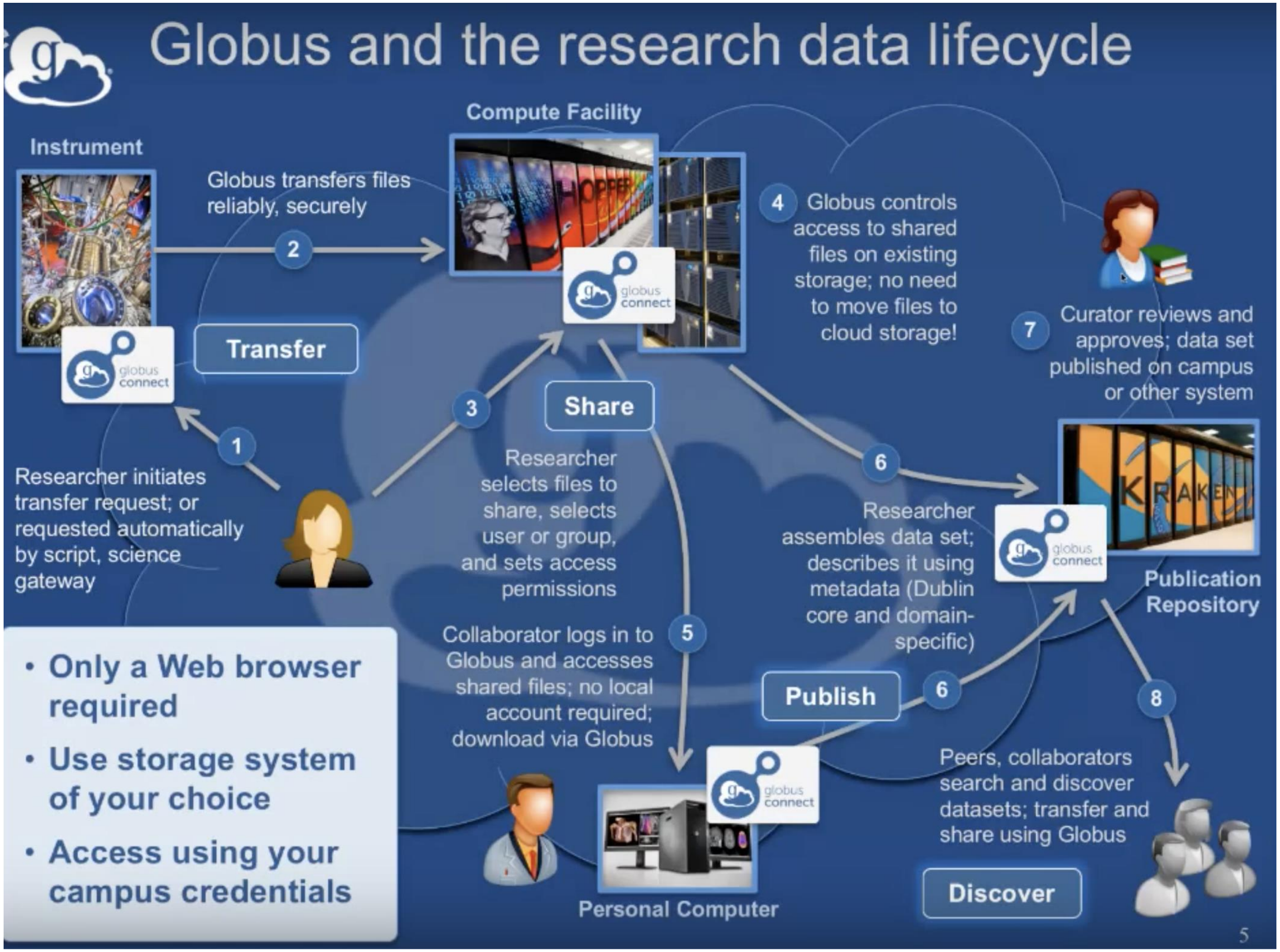
- Vulnerabilities:
  - **Cross Site Request Forgery (CSRF)**
    - A third party browser that is not authentic can deceive you to enter sensitive details
    - For example your banker’s site may be compromised and you may enter your details which may be available to the attacker
    - This is some times called the “Confused Deputy” problem
    - Prevention
      - Do not click on URLs that are suspicious
  - **Unvalidated Redirects and Forwards**
    - Web programmers usually redirect URLs when a company changes its website or when working with third parties
    - Prevention
      - Do not redirect or forward URLs in your applications

Use Web Security Vulnerability Scanners!

- Automated applications that scan the entire website for vulnerabilities
- Reliable and recommended
- Examples
  - Microsoft Safety Scanner
  - Acunetix Web Vulnerability Scanner
  - Netsparker

Security in Globus

- Globus is a middleware that is developed at Argonne Lab and managed by the Globus Alliance Forum (GAF)
- Globus security feature is called the “Grid Security Infrastructure (GSI)
- Globus use proxy delegations, public key infrastructure (PKI), certificate authorities (CAs), Secure Socket Layer (SSL) / Transport Layer Security (TLS) technologies



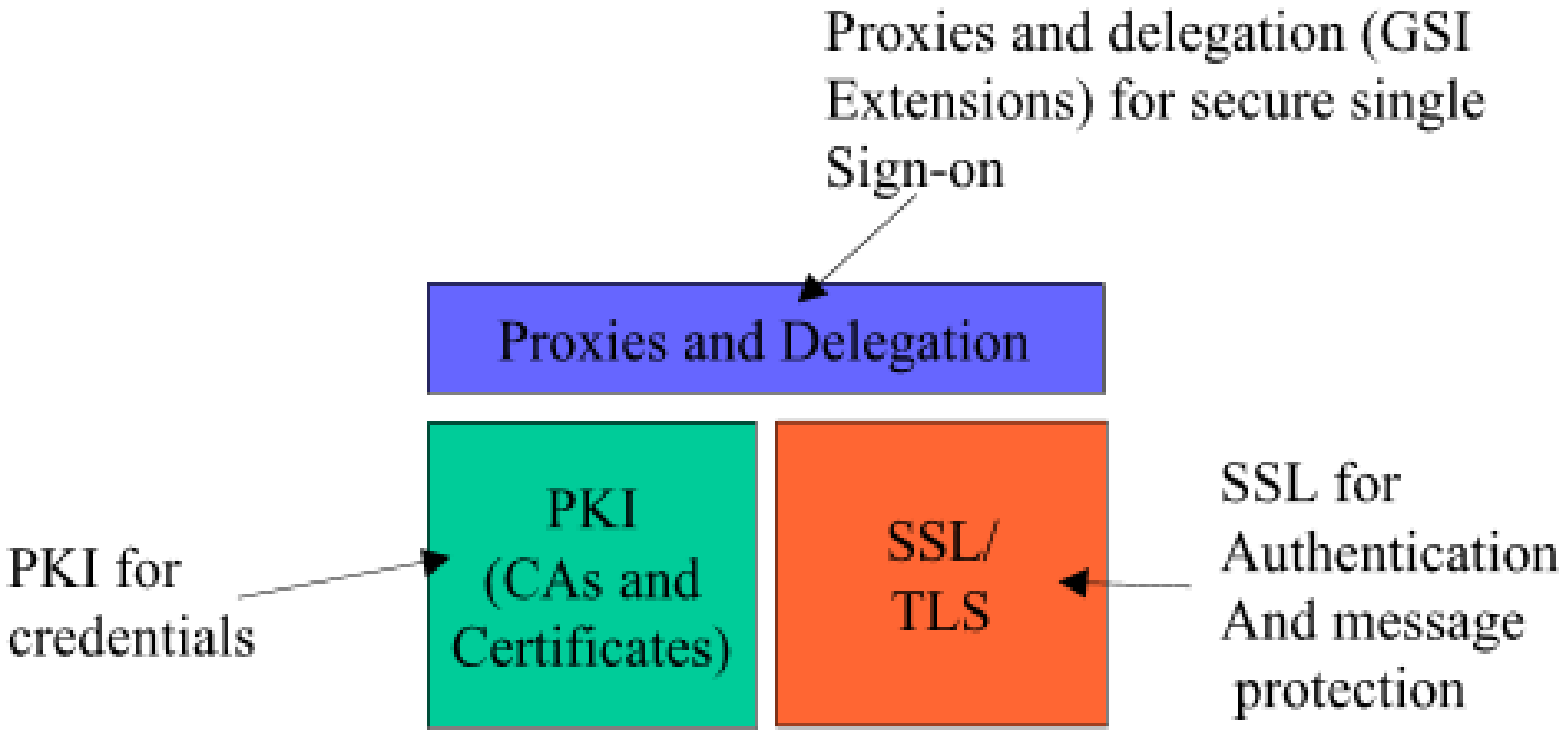
Globus and the research data lifecycle

Compute Facility
Instrument
Globus transfers files reliably, securely
2
Transfer
4 Globus controls access to shared files on existing storage; no need to move files to cloud storage!
7 Curator reviews and approves; data set published on campus or other system
3
Share
1
Researcher initiates transfer request; or requested automatically by script, science gateway
Researcher selects files to share, selects user or group, and sets access permissions
6
Researcher assembles data set; describes it using metadata (Dublin core and domain-specific)
Publication Repository
Collaborator logs in to Globus and accesses shared files; no local account required; download via Globus
5
Publish
6
8
Peers, collaborators search and discover datasets; transfer and share using Globus
Discover
Personal Computer
5

- Only a Web browser required
- Use storage system of your choice
- Access using your campus credentials

Queen Mary University of London

Grid Security Infrastructure (GSI) in Globus


Proxies and delegation (GSI Extensions) for secure single Sign-on
Proxies and Delegation
PKI for credentials
PKI (CAs and Certificates)
SSL/ TLS
SSL for Authentication And message protection

Source: Globus Toolkit Documentation

Getting Grid Security Certificate in Globus

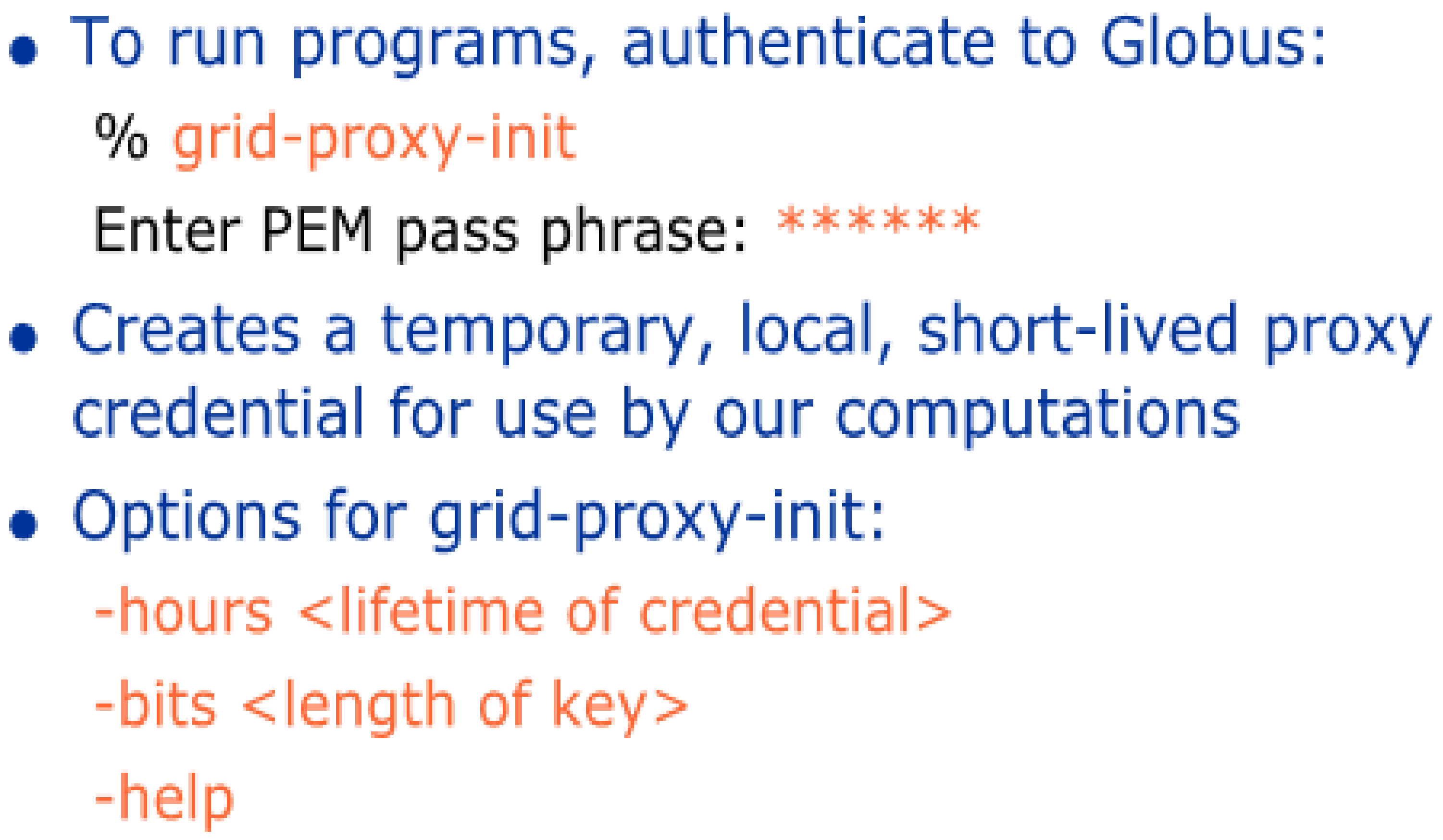
- The program grid-cert-request is used to create a public/private key pair and unsigned certificate in ~/.globus/:
  - usercert_request.pem: Unsigned certificate file
  - userkey.pem: Encrypted private key file
    - Must be readable only by the owner
- Mail usercert_request.pem to ca@globus.org
- Receive a Globus-signed certificate

  Place in ~/.globus/usercert.pem

Source: Globus Toolkit Documentation

Queen Mary
University of London

Logging onto Globus



- To run programs, authenticate to Globus:

```
% grid-proxy-init
Enter PEM pass phrase: ******
```

- Creates a temporary, local, short-lived proxy credential for use by our computations
- Options for grid-proxy-init:

```
-hours <lifetime of credential>
-bits <length of key>
-help
```

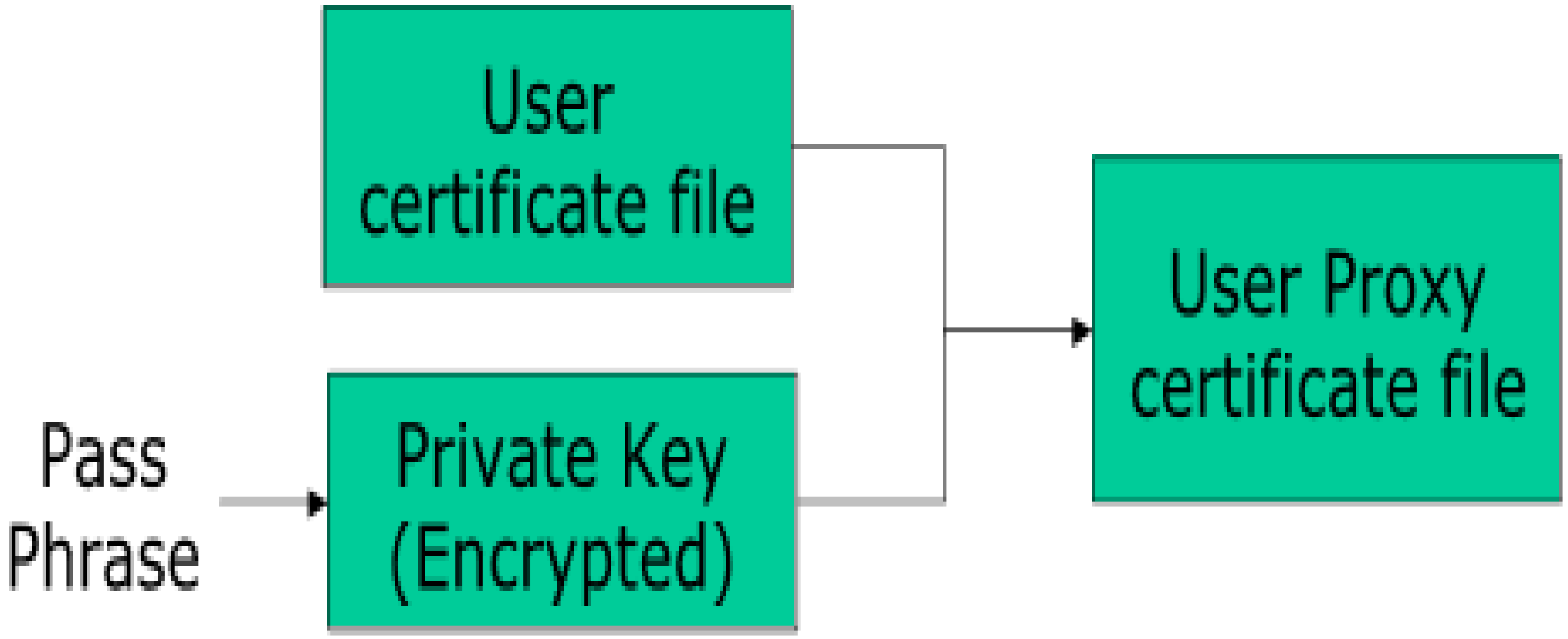
Source: Globus Toolkit Documentation

“grid-proxy-init” Details in Globus

- grid-proxy-init creates the local proxy file.
- User enters pass phrase, which is used to decrypt private key.
- Private key is used to sign a proxy certificate with its own, new public/private key pair.
  - User’s private key not exposed after proxy has been signed
- Proxy placed in /tmp, read-only by user
- NOTE: No network traffic!
- grid-proxy-info displays proxy details

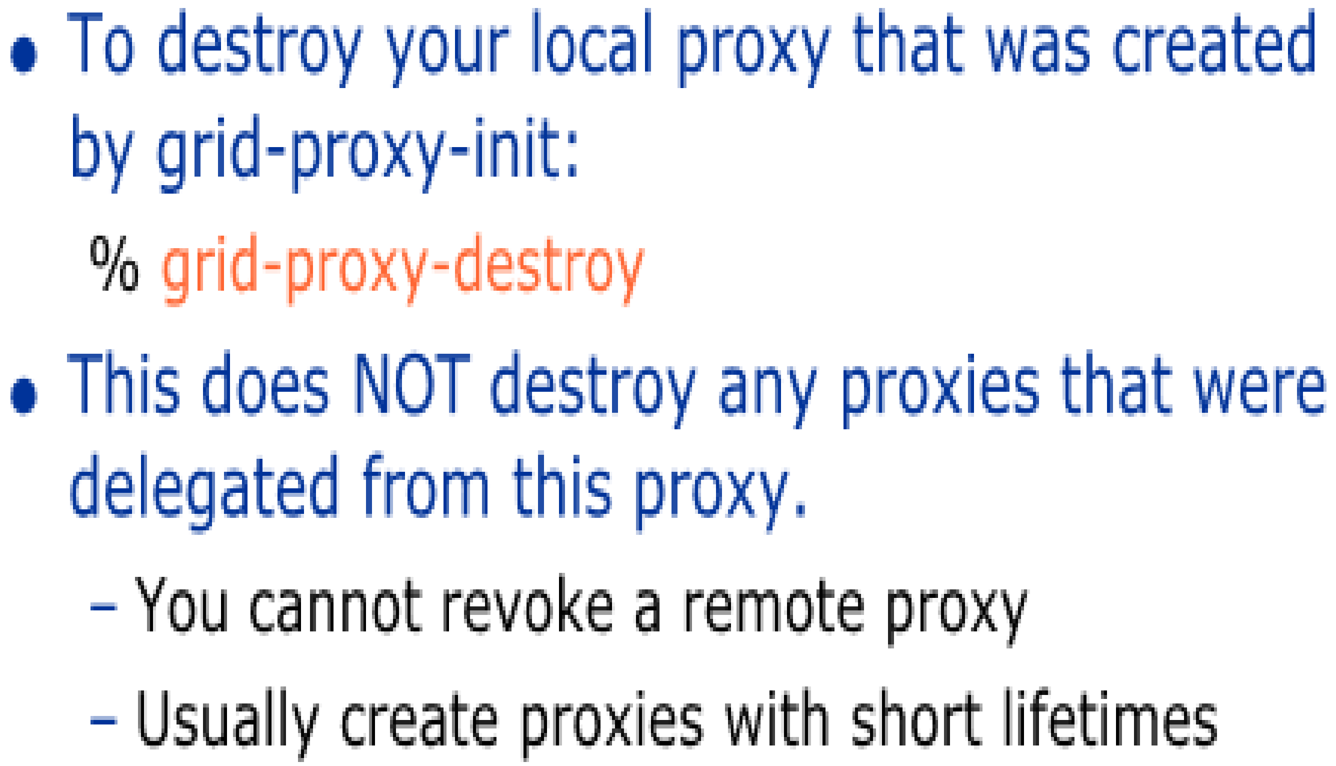
Source: Globus Toolkit Documentation

Sign-On with “grid-proxy-init”in Globus



Source: Globus Toolkit Documentation

Destroy Grid Proxy in Globus



- To destroy your local proxy that was created by grid-proxy-init:

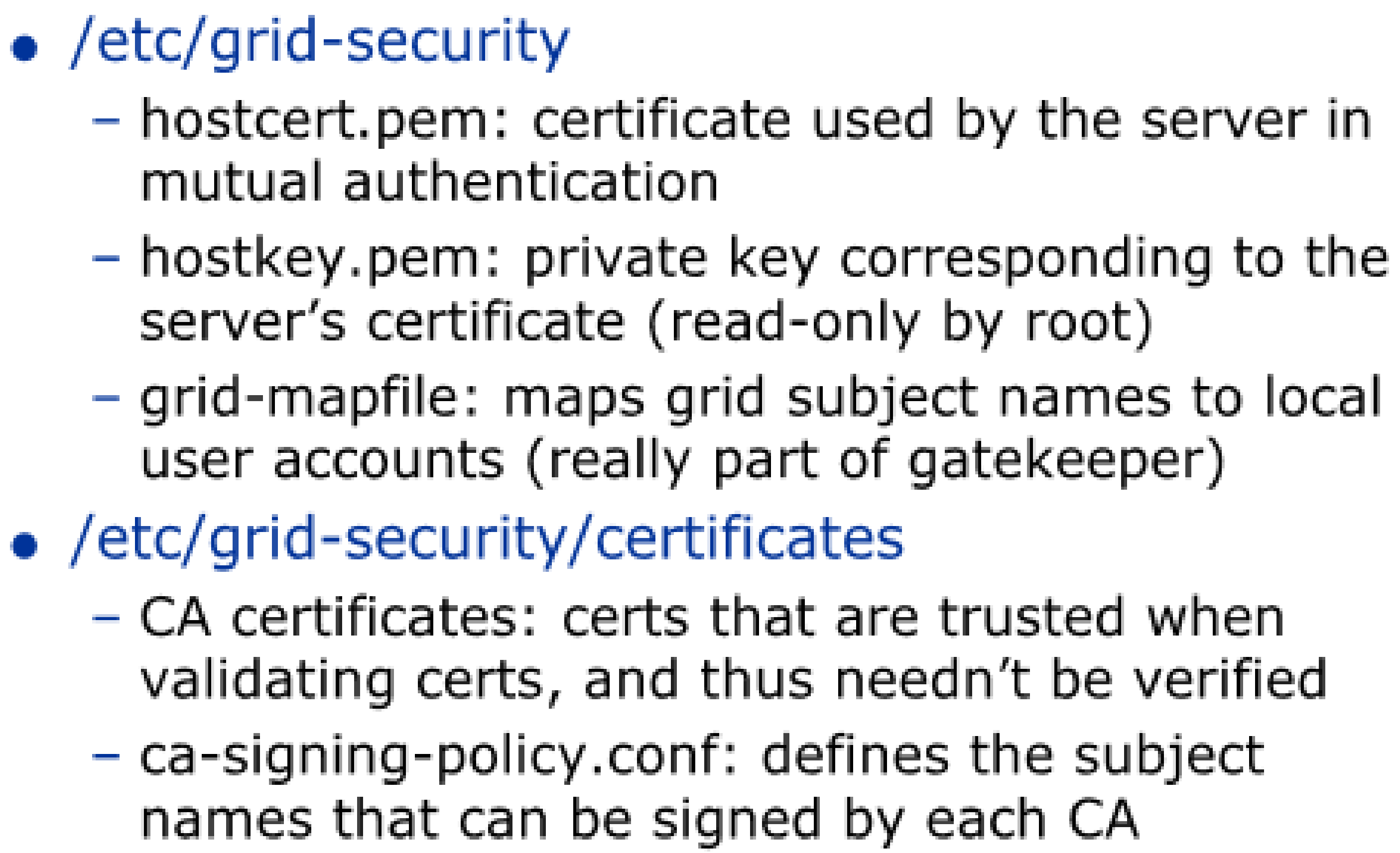
```
% grid-proxy-destroy
```

- This does NOT destroy any proxies that were delegated from this proxy.
  - You cannot revoke a remote proxy
  - Usually create proxies with short lifetimes

Source: Globus Toolkit Documentation

Queen Mary
University of London

Important GSI Files in Globus



- /etc/grid-security
  - hostcert.pem: certificate used by the server in mutual authentication
  - hostkey.pem: private key corresponding to the server's certificate (read-only by root)
  - grid-mapfile: maps grid subject names to local user accounts (really part of gatekeeper)
- /etc/grid-security/certificates
  - CA certificates: certs that are trusted when validating certs, and thus needn't be verified
  - ca-signing-policy.conf: defines the subject names that can be signed by each CA

Source: Globus Toolkit Documentation

Important GSI Files in Globus

- $HOME/.globus
  - usercert.pem: User's certificate (subject name, public key, CA signature)
  - userkey.pem: User's private key (encrypted using the user's pass phrase)
- /tmp
  - Proxy file(s): Temporary file(s) containing unencrypted proxy private key and certificate (readable only by user's account)
    - Same approach Kerberos uses for protecting tickets

Source: Globus Toolkit Documentation

“grid-mapfile” and “group-mapfile” in GSI

- **Grid-mapfile maps individual users** to their **proxy** and **certificates** for authentication and authorisation
- **Group-mapfile maps individuals belonging** to **particular groups** for authentication and authorisation

Case Study: National Grid Service using Globus

- Computational Resource Execution And Management for Computing Element (**CREAM-CE**) uses the concept of **Globus and gLite Middleware**
- Implementation of CREAM CE and Testing at the Science and Technology Facilities Council (STFC) within the National Grid Service (NGS) in the UK

An Overview of CREAM CE

Queen Mary
University of London

Introduction

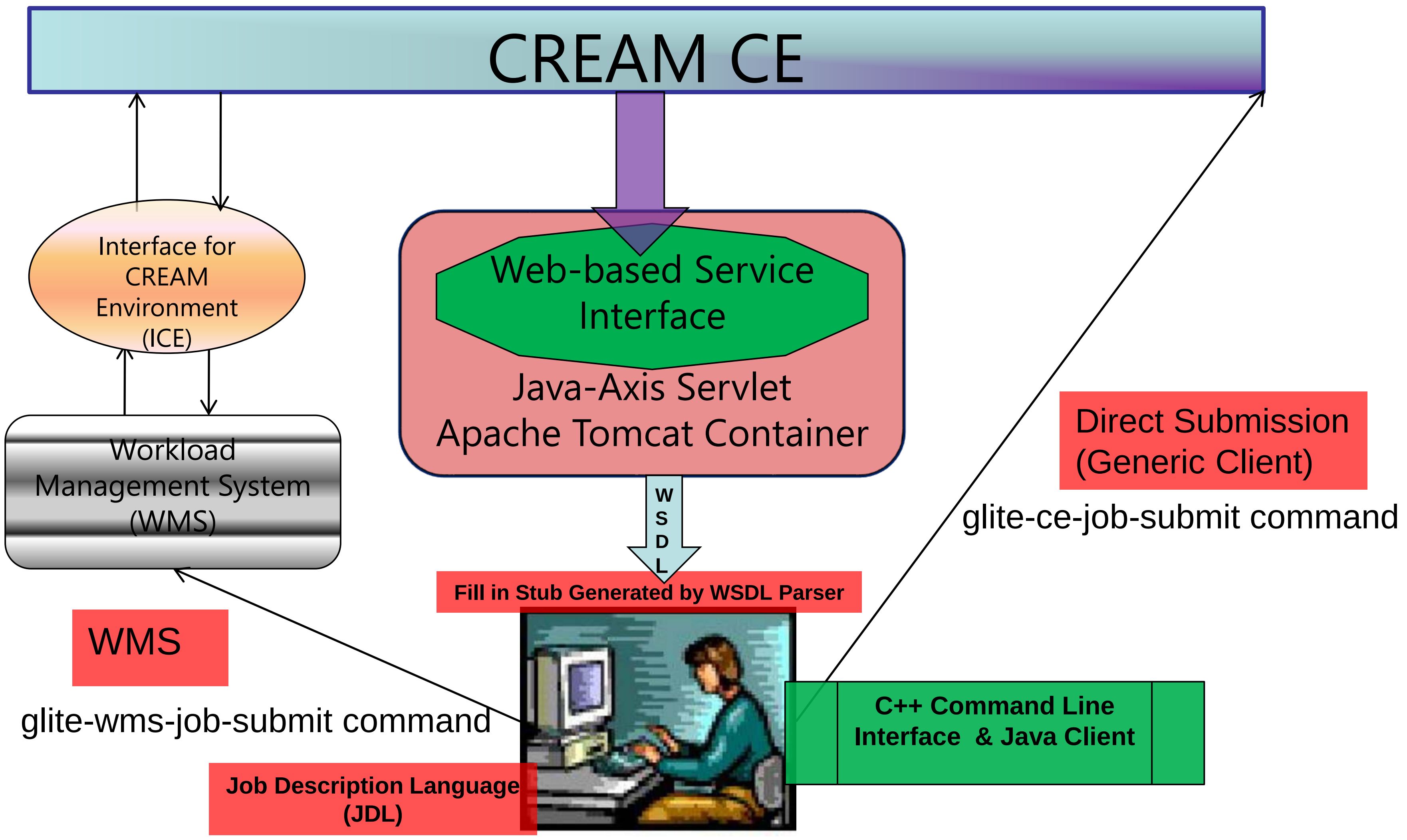
- CREAM-CE (Computing Resource Execution And Management-Computing Element)
  - Is a **gLite middleware** that provides services for job management operations at the Computing Element level
  - Accepts jobs submission requests described with the same **JDL (Job Description Language)** used by **WMS (Workload Management System)**
  - Supports LSF (Load Sharing Facility), PBS (Portable Batch System)/Torque, GE (Grid Engine) & Condor batch systems
- CEMon (Computing Element Monitor): Consists of independent java CEMonitor which notifies users when job changes state.

Queen Mary
University of London

CREAM CE Main Functionalities

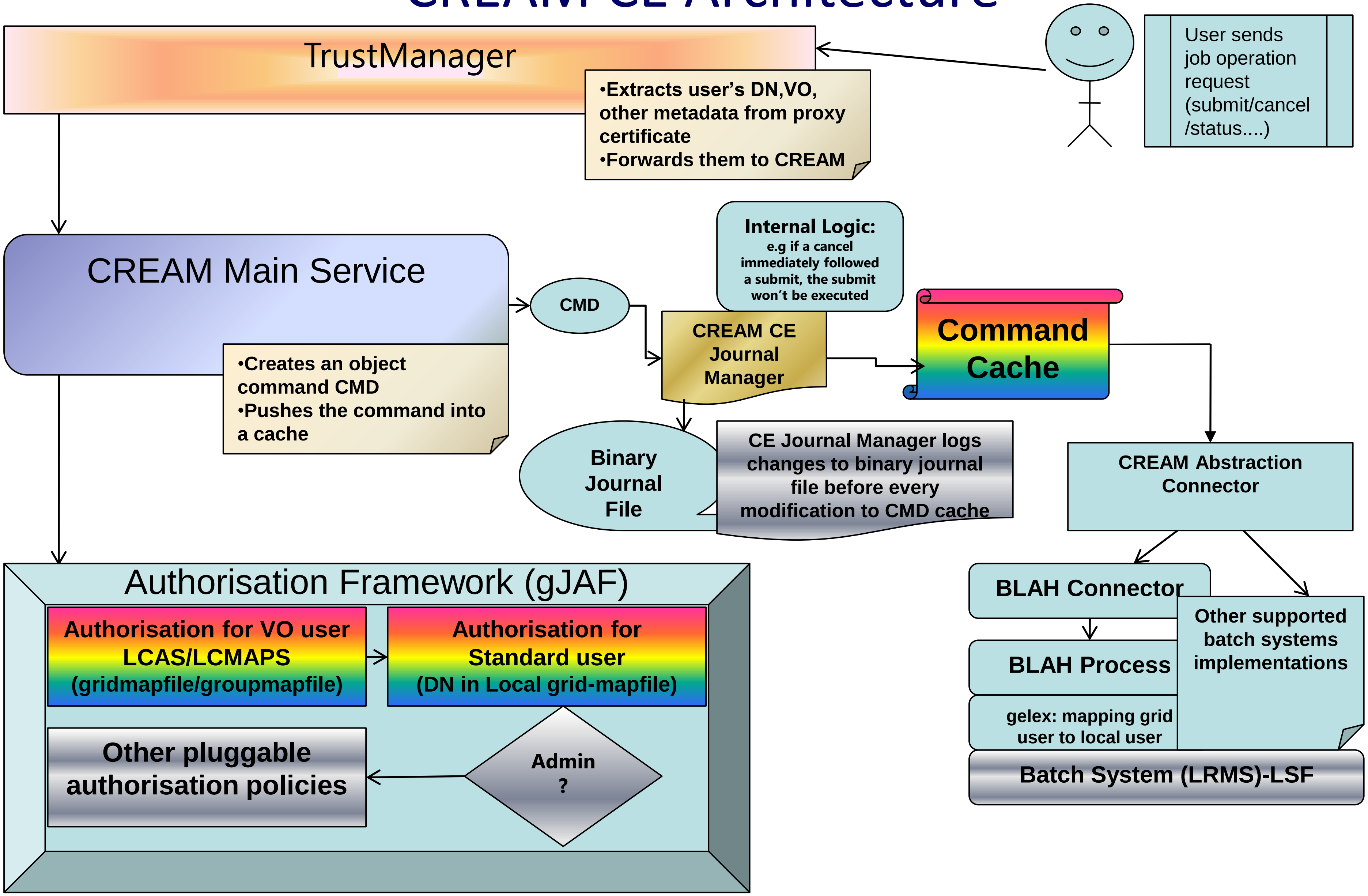
- Job submission
  - Direct staging of files that are GLITE WMS JDL compliant
  - Support for batch and MPI jobs
- Job listing & Job cancellation
- Job suspension & resumption
- Job info based on submission time/job status
- Job purge for terminated jobs
- Disable/enable new submissions by Admin & Super users
  - glite-ce-disable-submission
  - glite-ce-enable-submission

CREAM CE Interface


CREAM CE
Interface for CREAM Environment (ICE)
Web-based Service Interface
Java-Axis Servlet
Apache Tomcat Container
WSDL
Workload Management System (WMS)
Fill in Stub Generated by WSDL Parser
Direct Submission (Generic Client)
glite-ce-job-submit command
WMS
glite-wms-job-submit command
Job Description Language (JDL)
C++ Command Line Interface & Java Client

Queen Mary
University of London

CREAM CE Architecture

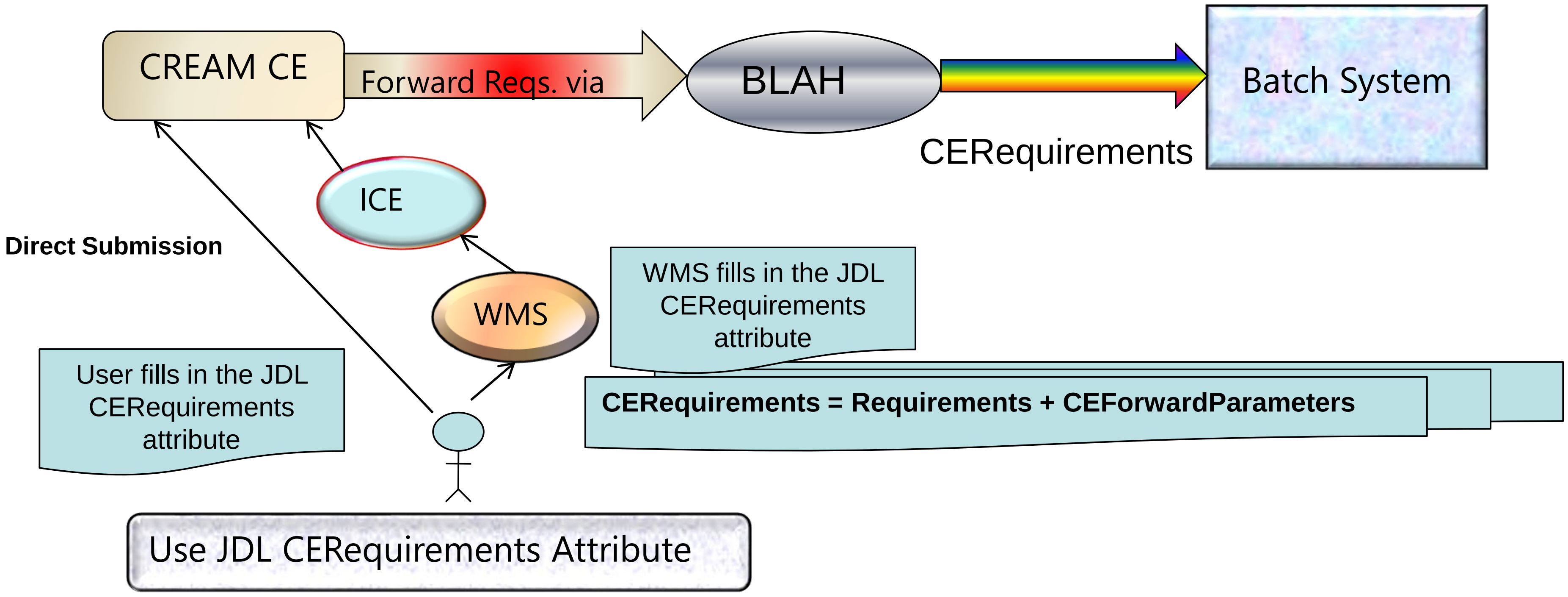

TrustManager
•Extracts user's DN,VO, other metadata from proxy certificate
•Forwards them to CREAM
User sends job operation request (submit/cancel /status....)
CREAM Main Service
•Creates an object command CMD
•Pushes the command into a cache
CMD
Internal Logic: e.g if a cancel immediately followed a submit, the submit won't be executed
CREAM CE Journal Manager
Command Cache
Binary Journal File
CE Journal Manager logs changes to binary journal file before every modification to CMD cache
CREAM Abstraction Connector
BLAH Connector
Other supported batch systems implementations
BLAH Process
gelex: mapping grid user to local user
Batch System (LRMS)-LSF
Authorisation Framework (gJAF)
Authorisation for VO user LCAS/LCMAPS (gridmapfile/groupmapfile)
Authorisation for Standard user (DN in Local grid-mapfile)
Other pluggable authorisation policies
Admin ?

Queen Mary University of London

LCAS-Local Centre Authorisation Service
LCMAPS-Local Credential Mapping Service

The BLAHP Component & Forward Requirements to Batch System

- The BLAHP (Batch Local ASCII Helper Protocol) is used by CREAM CE to manage batch jobs

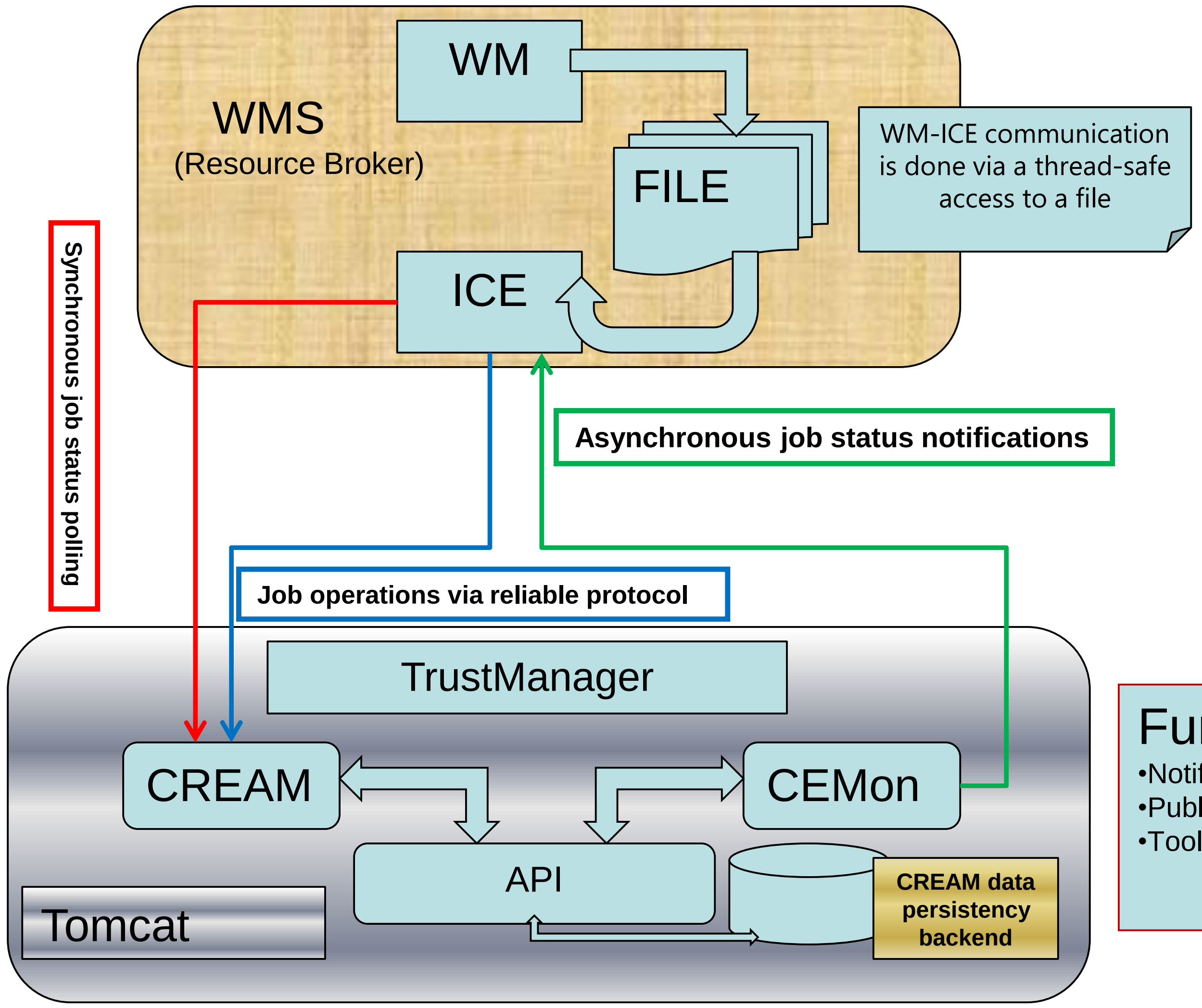

CREAM CE
Forward Reqs. via
BLAH
CERequirements
Batch System
ICE
Direct Submission
WMS
WMS fills in the JDL CERequirements attribute
CERequirements = Requirements + CEForwardParameters
User fills in the JDL CERequirements attribute
Use JDL CERequirements Attribute

- BLAHPD daemon is used
  - To translate BLAHP commands to batch system actions
  - Interprets the result in BLAHP format
  - BLPARSER is the main component that gets information on the status of the job via the batch log files
    - Blparser must be installed on a machine where the batch log files are available or can have access to them.

WMS-Workload Management Service
ICE-Interface for Computing Element

Queen Mary
University of London

CREAM CE, WMS & ICE


WM
WMS
(Resource Broker)
FILE
WM-ICE communication is done via a thread-safe access to a file
ICE
Synchronous job status polling
Asynchronous job status notifications
Job operations via reliable protocol
TrustManager
CREAM
CEMon
API
CREAM data persistency backend
Tomcat

Function of CEMonitor:

- Notify user when job changes state
- Publishes information as Topics
- Tools subscribe to the topics to know job status

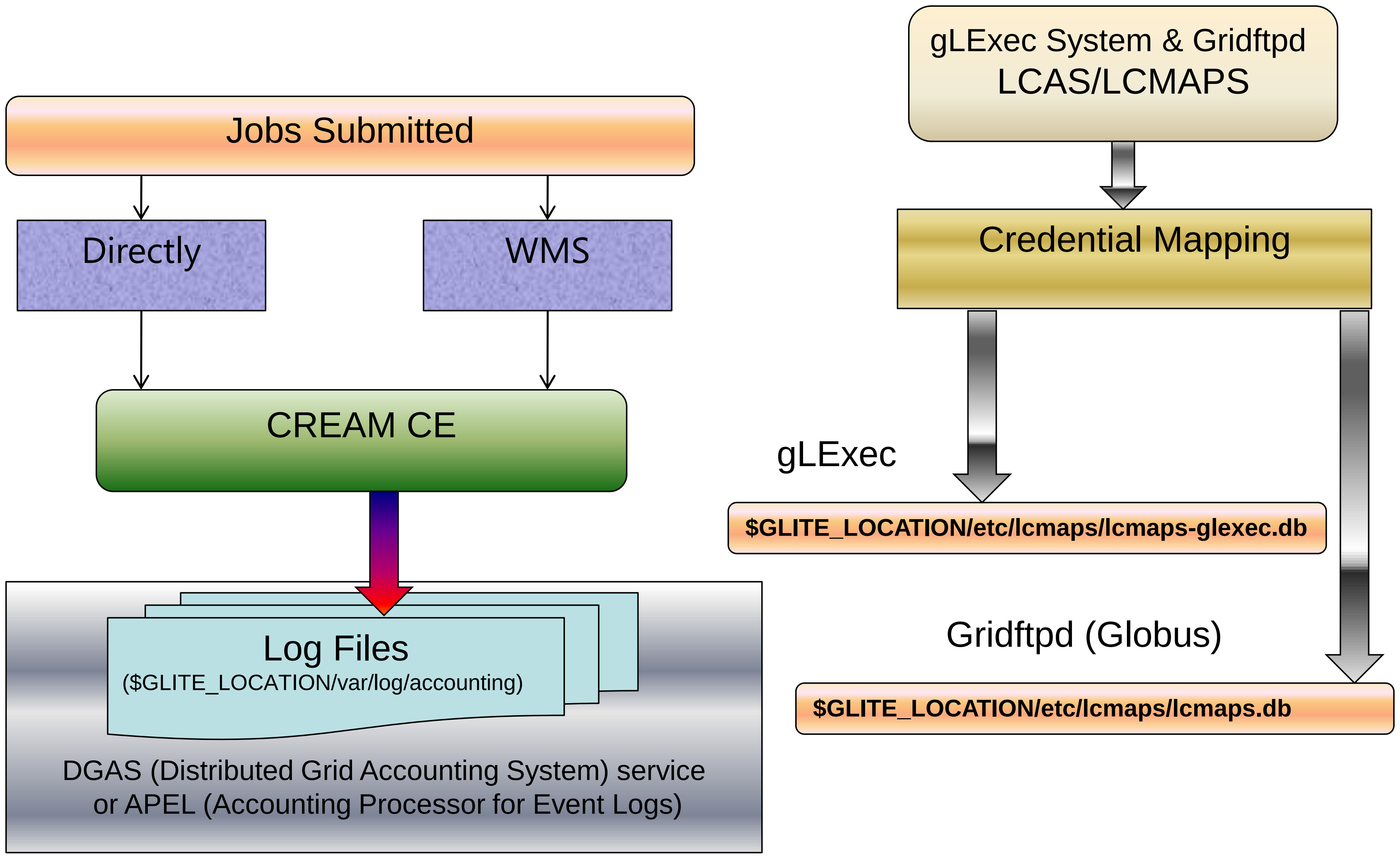
Queen Mary
University of London

CREAM CE Installation

- Setup yum repository for your specific batch system e.g LSF
  - wget http://grid-deployment.web.cern.ch/grid-deployment/glite/repos/3.1/glite-LSF_utils.repo -O /etc/yum.repos.d/glite-TORQUE_utils.repo
- Install tomcat
  - yum install tomcat5
- Install CREAM metapackage
  - yum install glite-CREAM
- Install your specific batch system metapackage
  - yum install glite-LSF_utils

Queen Mary
University of London

Accounting & Credential Mapping


Jobs Submitted
Directly
WMS
CREAM CE
Log Files
($GLITE_LOCATION/var/log/accounting)
DGAS (Distributed Grid Accounting System) service
or APEL (Accounting Processor for Event Logs)
gLExec System & Gridftpd
LCAS/LCMAPS
Credential Mapping
gLExec
$GLITE_LOCATION/etc/lcmaps/lcmaps-glexec.db
Gridftpd (Globus)
$GLITE_LOCATION/etc/lcmaps/lcmaps.db

Queen Mary
University of London

Management Control Mechanisms

- Start and stop service
  - /etc/init.d/tomcat start/stop
- Adding a VO
  - Reconfigure CREAM with the VO
- Ban a user. Put the DN of the user in the file:
  - /opt/glite/etc/glite-ce-cream/banned.lst
- Trace specific job
- Drain CREAM CE
- Self-limiting CREAM behaviour
  - $GLITE_LOCATION/bin/glite_cream_load_monitor

Queen Mary
University of London

The Pros of CREAM CE

- **Interoperability: Web Service** interface with clients written in any programming language.
- **Testing:**
  - The **CheckCreamConf** script performs configuration test to confirm if the installation/configuration is successful.
- CREAM allows **WMS & direct** modes of job submissions.
- Self-limiting CREAM behaviour and Draining command

The Cons of CREAM CE

- No Globus submission
- Complex configuration of authorisation mechanism
  - Good news! Argus (the new single glite authorisation service) is being tested for release to replace the present authorisation mechanism.

Study References

- CREAM-CE website
  - http://grid.pd.infn.it/cream/
- The Art of Software Security Assessment: Identifying and Preventing Software Vulnerabilities by John McDonald, Mark Down and Justin Schuh
- Globus website
  - https://globus.org/