Inside the GDS

The Engine, Activities, Data Resource Implementations and Role Mapping

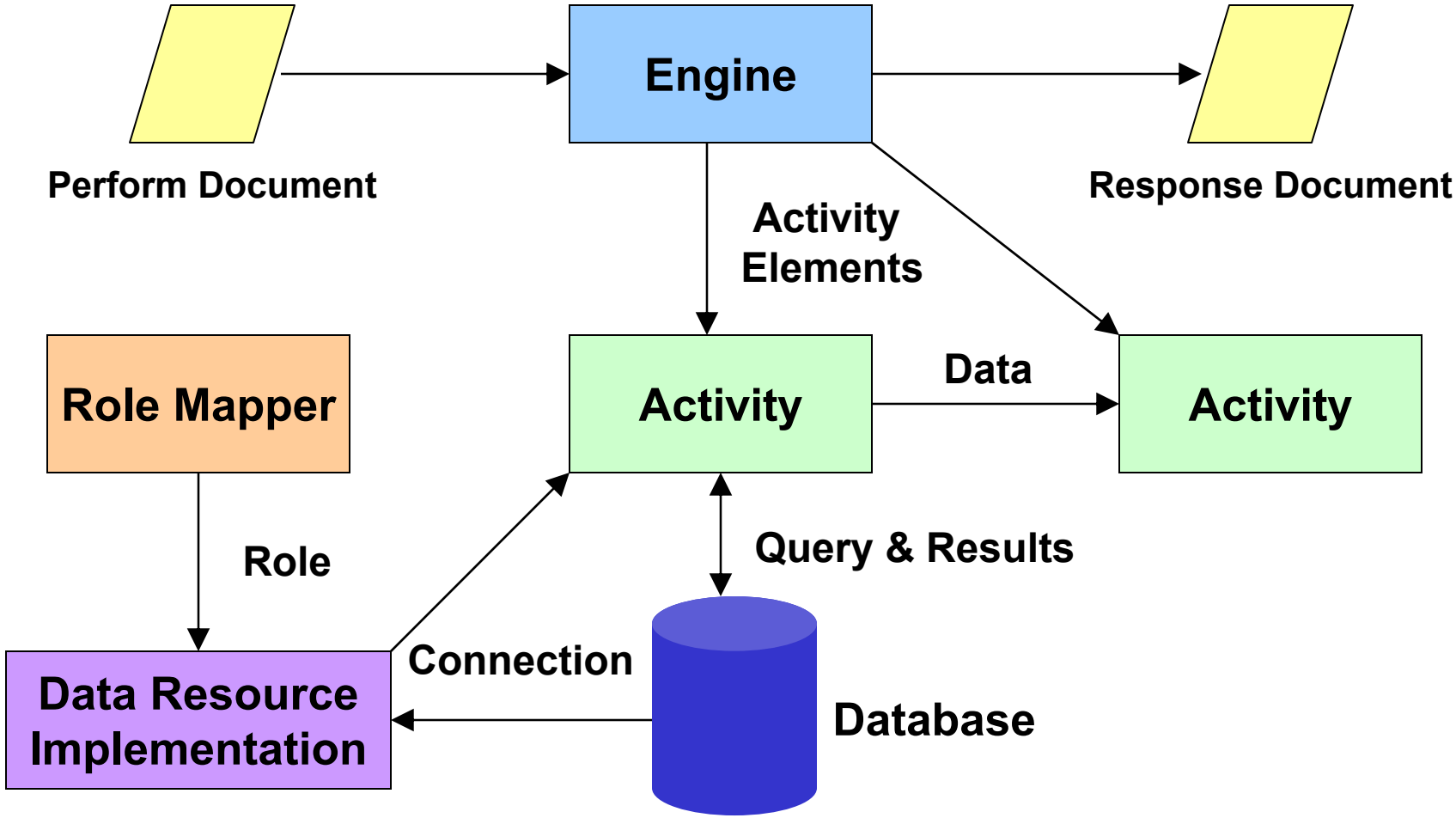
EPCC, University of Edinburgh
Tom Sugden (tom@epcc.ed.ac.uk)

First International Summer School on Grid Computing, Vico Equense, Italy

Overview

- Low-level components of a Grid Data Service:
  - Engine
  - Activities
  - Data Resource Implementations
  - Role Mappers
- Extensibility of the OGSA-DAI architecture
  - Interfaces and Implementations

Internal Architecture



The Engine

- The Engine is the central component of a GDS.
- Dictates the behaviour of the service when documents are submitted.
  - Parses perform document
  - Identifies required activities
  - Instantiates the activity implementations
  - Processes the activities
  - Combines outputs to form a response document
  - Returns response document to GDS

The Engine Interface

- Allows different implementations to be developed to alter the behaviour of the service.

<Engine>
+ invoke(performDocument: Document, context: Map) : Document **+ terminate(): void**

- **`Invoke`** invokes a request.
- **`Terminate`** terminates the request currently processing, if one exists.

Engine Construction

- When a GDS is created, it instantiates an Engine using details from its configuration
- The Engine constructor takes a Context object known as the Engine Context

Engine Context

Activity Map
Schema Map
Data Resource Implementation

Engine Invocation

- *GridDataService::perform* takes an XML document as input parameter
- GDS calls the Engine's **`invoke`** method.

```
invoke( performDocument: Document,
        invocationContext: Map ): Document
```

- The perform document describes actions for GDS to perform
- The invocation context contains the distinguished name from the user certificate

The OGSA-DAI R3 Engine

- Executes one request at a time.
- Validates perform documents against their schema.
- Terminates a request when:
  - all activities have completed
  - an error occurs
  - the terminate method is called
- When a request is terminated, all data relating to that request is discarded.
- Status changes processed by the engine are published as service data.

Processing a Perform Document

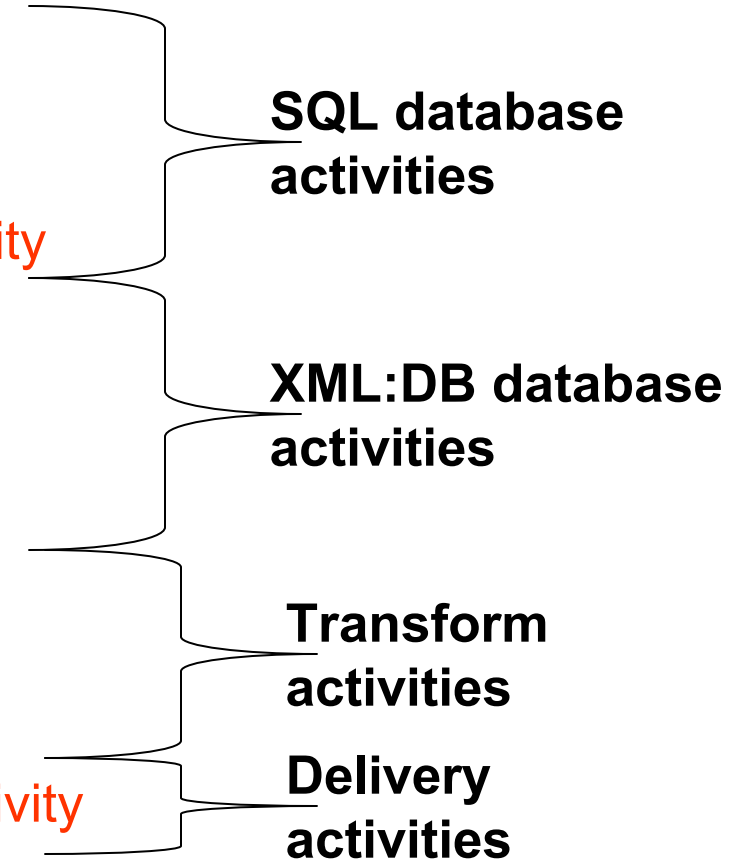
- The Engine validates perform documents using the schema map from the Engine Context
  - activity element → activity schema
- Instantiates activity implementations using the activity map from the Engine Context
  - activity element → activity implementation class
- Creates an Activity Handler to process the activity

Activities

- An Activity dictates an action to be performed by a GDS
  - Query a data resource
  - Transform data
  - Deliver results
- Each Activity has a corresponding:
  - Activity Element **sqlQueryActivity**
  - XSD schema **sql_query_statement.xsd**
  - Java implementation **SQLQueryStatementActivity**

Provided Activity Implementations

- OGSA-DAI R3 provides
  - SQLQueryStatementActivity
  - SQLStoredProcedureActivity
  - SQLUpdateActivity
  - RelationalResourceManagementActivity
  - XPathStatementActivity
  - XUpdateStatementActivity
  - XMLCollectionManagementActivity
  - XMLResourceManagementActivity
  - XSLTransformActivity
  - GZIPCompressionActivity
  - ZIPArchiveActivity
  - Deliver[To|From][URL|GDT|GFTP]Activity



Activity Handlers

- Activity Handlers process Activities
  - Manage inputs and outputs
  - Perform the processing
  - Monitor the status
- Decouples activity processing behaviour from the engine and activities
  - SimpleHandler

    Generates output only when it is required
  - RunAheadHandler

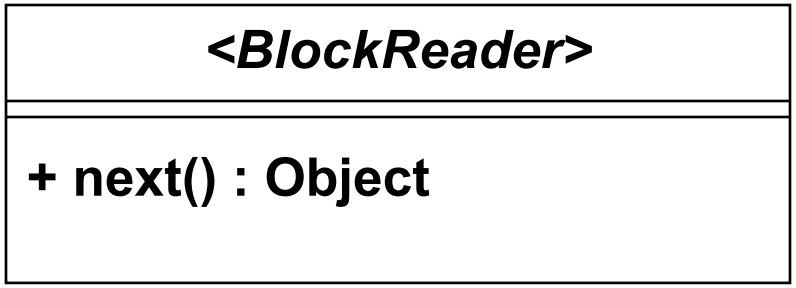
    Generates output before it is required

Activity Inputs and Outputs

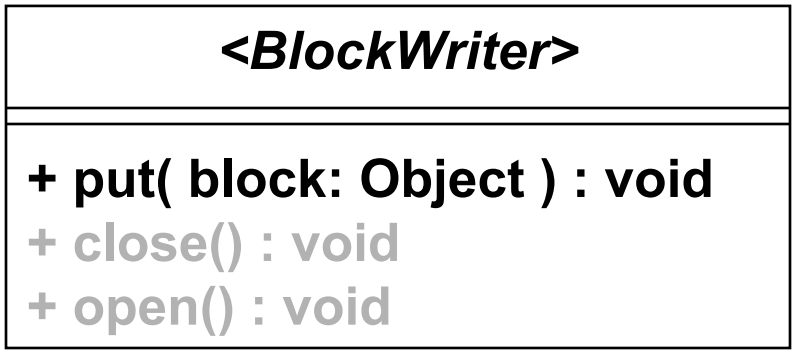
- Activities read and write blocks of data
  - Allows efficient streaming between activities to reduce memory overhead
- A block is an Object
  - Usually a String or byte array
- Currently input and outputs are untyped
- Interfaces for reading and writing
  - BlockReader
  - BlockWriter

BlockReader and BlockWriter Interface

- Activity inputs

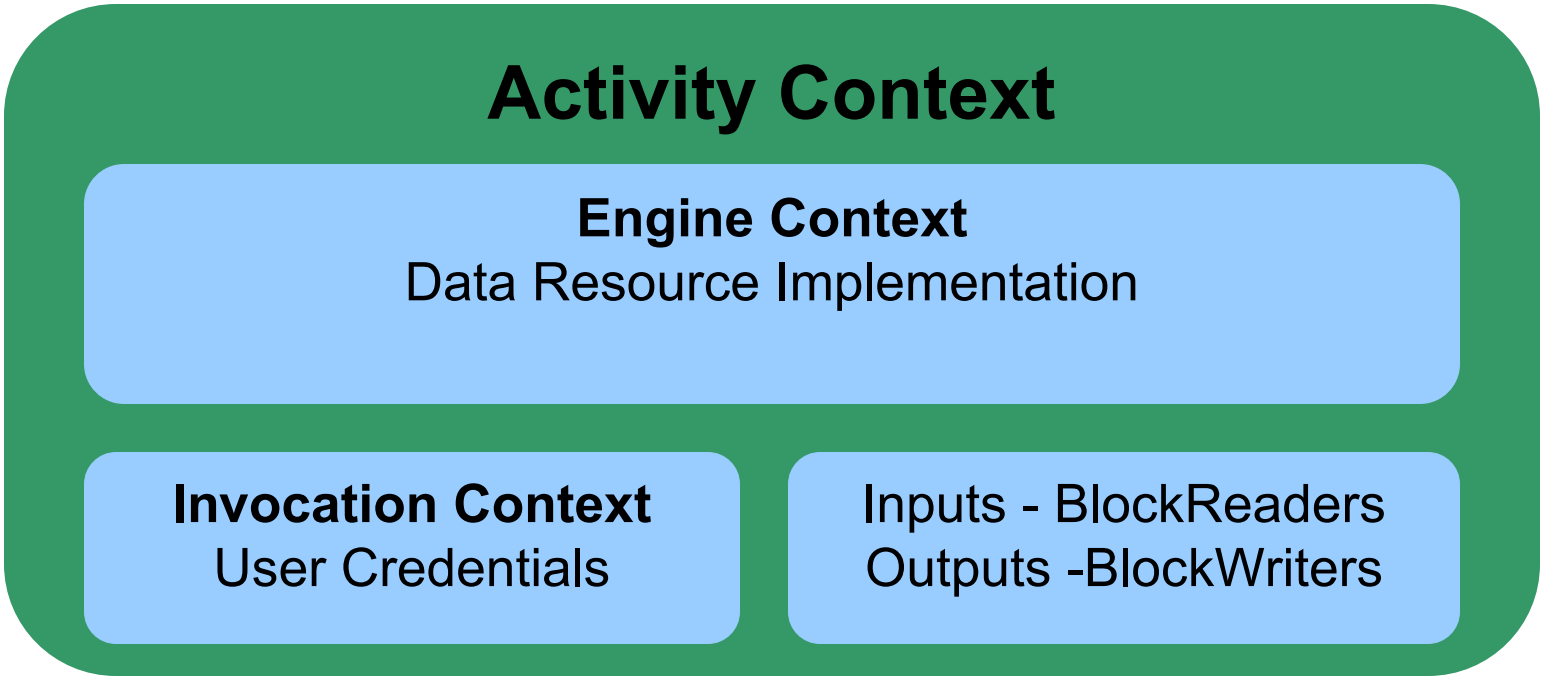


- Activity output



The Activity Context

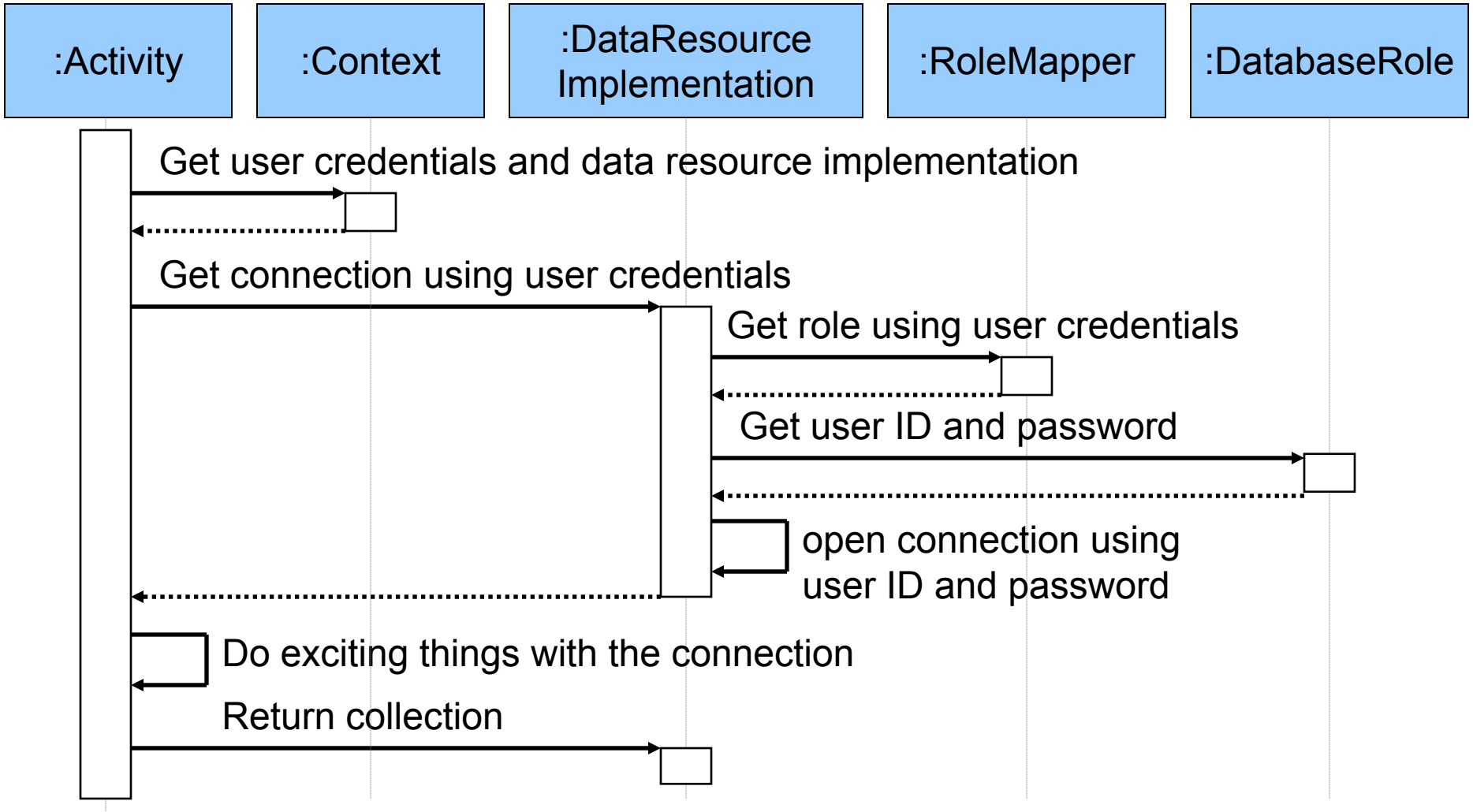
- Activities are initialised using the Activity Context



Accessing Data Resources

- Activities often interact with data resources
  - Query a database, update a table row, etc
- Data resources often require user validation
  - User ID and password
- An Activity can use its Context information to access and interact with a data resource
  - Data Resource Implementation
  - User Credentials

Accessing Data Resource Sequence Diagram



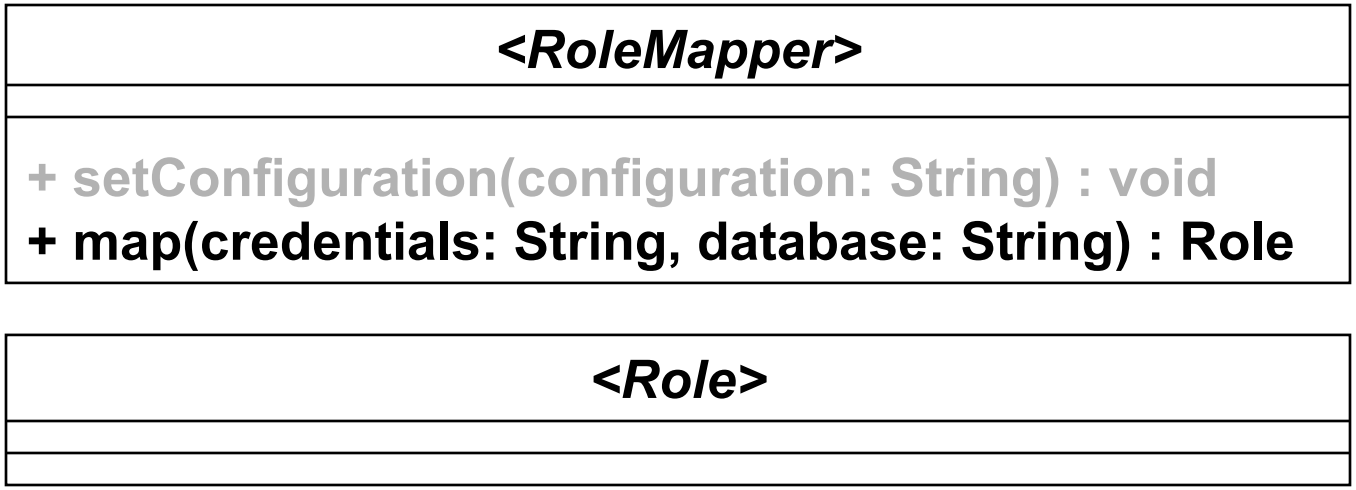
Data Resource Implementations

- Data Resource Implementations govern access to data resources
  - Open/Close connections
  - Validate user credentials using a RoleMapper
  - Facilitates connection pooling
- All Data Resource Implementations extend an abstract base class

uk.org.ogsadai.porttype.gds.dataresources.
DataResourceImplementation

The Role and RoleMapper Interfaces

- A RoleMapper maps user credentials to a Role



- OGSA-DAI provides
  - **SimpleFileRoleMapper – reads database roles from a file**
  - **DatabaseRole – encapsulates username and password**

Role Mapping

- The SimpleFileRoleMapper loads the Role Map file referenced from the GDSFConfig
- This file maps X509 Certificate User Credentials to username and password combinations
  - An X509 Certificate is a type of digital document used in Web Services to attest to the identity of an individual or other entity

```
…
<Database name="MyDatabase">
      <User dn="J.Smith@somewhere.co.uk,OU=Group,O=Org,O=Another"
            userid="jsmith" password="carrotcake" />
</Database>
…
```

Conclusion

- The Engine is the core of a GDS.
- The Engine uses Activities to perform actions.
- Activities use Data Resource Implementations to access data resources.
- OGSA-DAI R3 includes many activities for querying, updating, transforming and delivering data.
- Architecture is designed for extensibility:
  - New Activities
  - New Role Mappers
  - New Data Resource Implementations