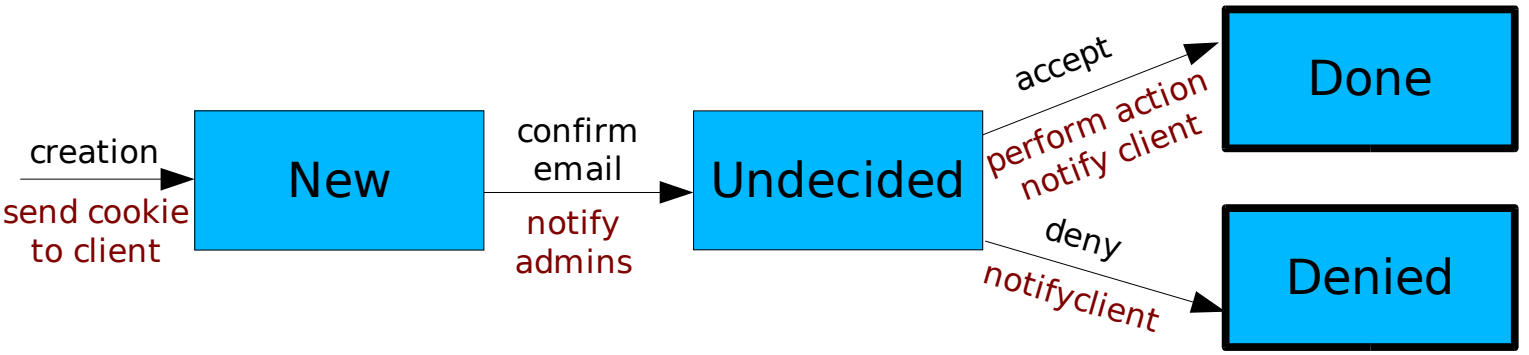
VO Management with VOMS

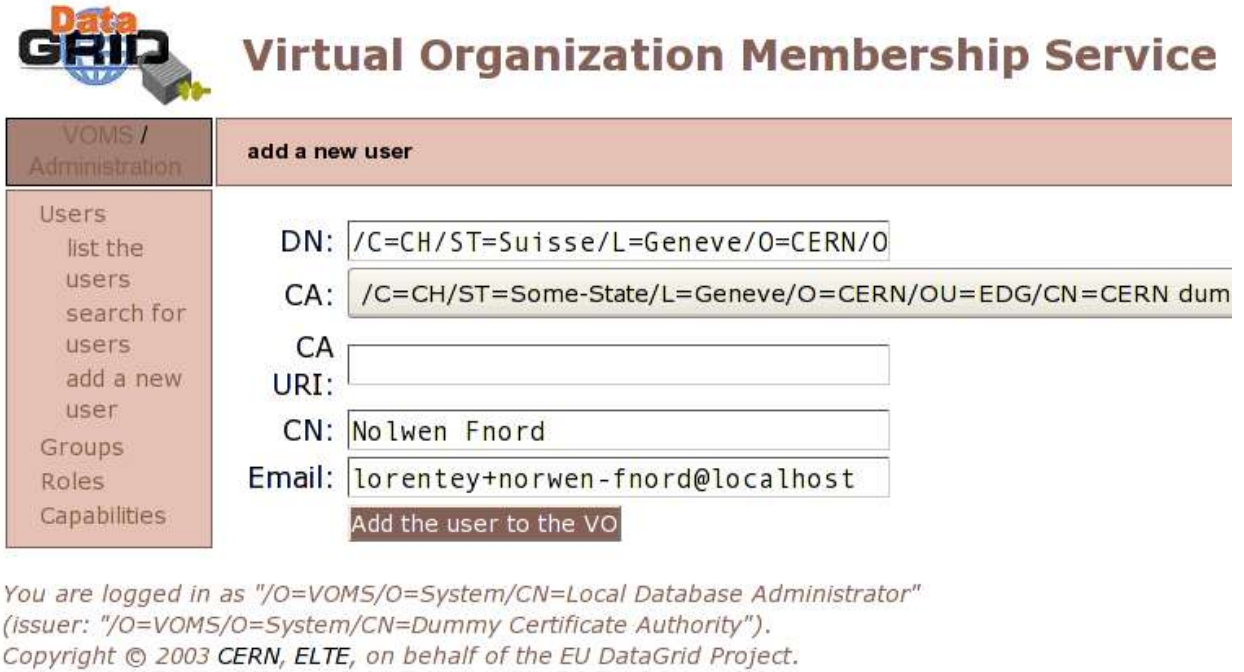
Ákos Frohner
Károly Lőrentey

Registration Flow

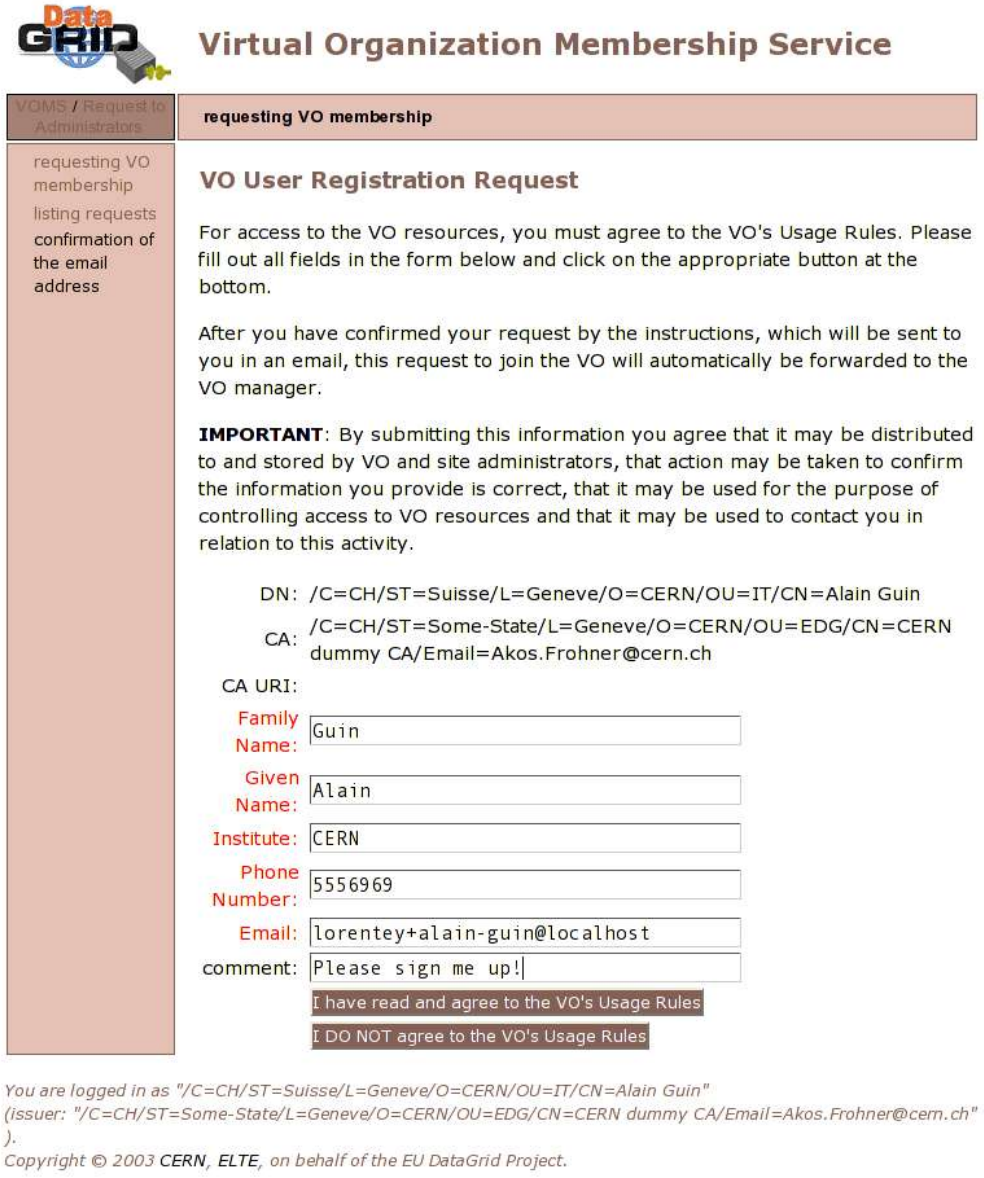


Nolwen Fnord as admin

- Create new user as local admin (local DB maintenance mode)
- Assign Nolwen to the /Fnord/Role=VO-Admin role
- Other admin functions:
  - group, role and capability creation and deletion
  - add/remove users from these containers
  - modify user details



Alain Guin as user



Virtual Organization Membership Service

VOMS / Request to Administrators

requesting VO membership

requesting VO membership
listing requests
confirmation of the email address

VO User Registration Request

For access to the VO resources, you must agree to the VO's Usage Rules. Please fill out all fields in the form below and click on the appropriate button at the bottom.

After you have confirmed your request by the instructions, which will be sent to you in an email, this request to join the VO will automatically be forwarded to the VO manager.

IMPORTANT: By submitting this information you agree that it may be distributed to and stored by VO and site administrators, that action may be taken to confirm the information you provide is correct, that it may be used for the purpose of controlling access to VO resources and that it may be used to contact you in relation to this activity.

DN: /C=CH/ST=Suisse/L=Geneve/O=CERN/OU=IT/CN=Alain Guin

CA: /C=CH/ST=Some-State/L=Geneve/O=CERN/OU=EDG/CN=CERN dummy CA/Email=Akos.Frohner@cern.ch

CA URI:

Family Name: Guin

Given Name: Alain

Institute: CERN

Phone Number: 5556969

Email: lorentey+alain-guin@localhost

comment: Please sign me up!

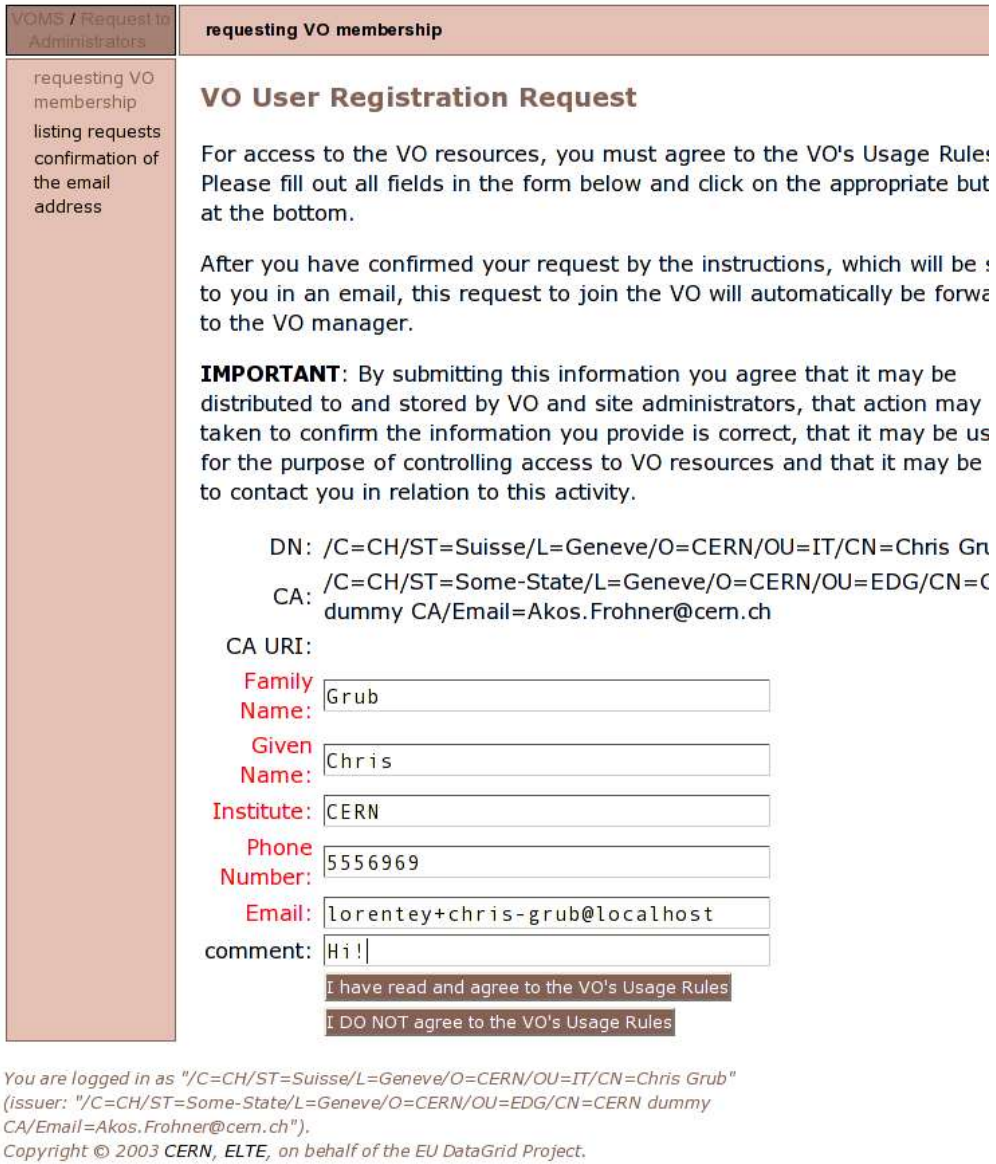
I have read and agree to the VO's Usage Rules

I DO NOT agree to the VO's Usage Rules

You are logged in as "/C=CH/ST=Suisse/L=Geneve/O=CERN/OU=IT/CN=Alain Guin" (issuer: "/C=CH/ST=Some-State/L=Geneve/O=CERN/OU=EDG/CN=CERN dummy CA/Email=Akos.Frohner@cern.ch").

- Request for VO membership
- Receiving the confirmation email
- Confirmation of the address
- ... waiting for Nolwen

Chris Grub as luser



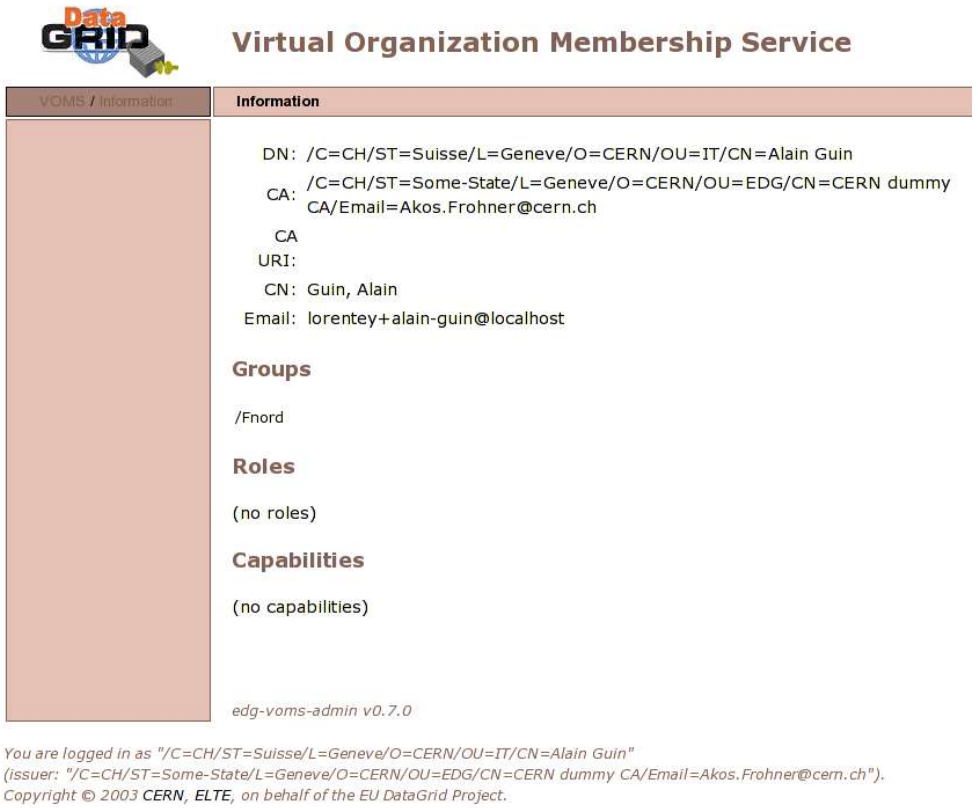
- Request for VO membership
- Receiving the confirmation email
- Confirmation of the address
- ... waiting for Nolwen

Nolwen making decisions

- Pending requests
- Allow/deny decisions at once

- Other functionalities:
  - All requests (“resurrect”)
  - Incomplete requests
  - Details of a request

Happy Alain



Virtual Organization Membership Service

VOMS / Information

Information

DN: /C=CH/ST=Suisse/L=Geneve/O=CERN/OU=IT/CN=Alain Guin

CA: /C=CH/ST=Some-State/L=Geneve/O=CERN/OU=EDG/CN=CERN dummy CA/Email=Akos.Frohner@cern.ch

CA URI:

CN: Guin, Alain

Email: lorentey+alain-guin@localhost

Groups

/Fnord

Roles

(no roles)

Capabilities

(no capabilities)

edg-voms-admin v0.7.0

You are logged in as "/C=CH/ST=Suisse/L=Geneve/O=CERN/OU=IT/CN=Alain Guin" (issuer: "/C=CH/ST=Some-State/L=Geneve/O=CERN/OU=EDG/CN=CERN dummy CA/Email=Akos.Frohner@cern.ch").

- Receives notification
- Looks at the info page
- Starts using the VO resources by using edg-voms-proxy-init every day...
- Tries to add a group, but fails

Sad Chris

- Receives notification
- Tries to use the info page, but fails

Plans for Registration

- Request for group or role membersip
- Request for the creation of a new group or role
- Change of DN old certificate expires and has to be replaced by a new one
- Pre-registration hooks
  - Email address confirmation (done)
  - DN/CA check against the LCG guidelines database
  - Contact sites (to vote against)
- Post-registration hooks
  - Notify sites in addition to the client

VOMS plans

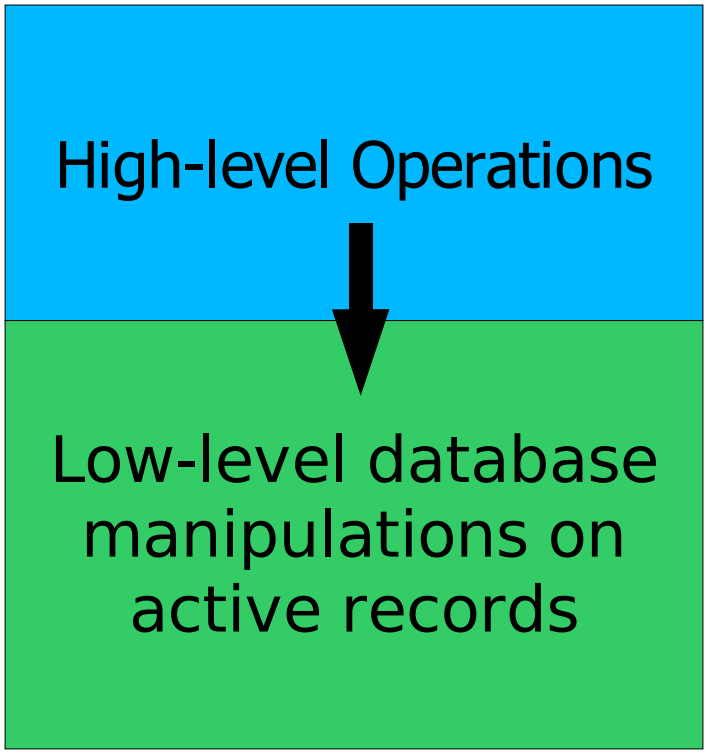
- Attribute Certificates (just arrived)
- Replication of the service
- History interface
- Change of user info:
  - Phone number, institute
  - Email address (confirmation required)
  - ... etc.
- ... improving the documentation! :-)

edg-voms-admin internals

- SOAP interface for all functionality
  currently with Java and Perl bindings, but can be extended to C/C++ clients stubs upon demand
- Command line and web interface (using the SOAP API)
- Complete audit logs for all events (at database level)
- Fine grained access control on all containers (role/group/cap)
- Extensible request flow engine with notifications

Internals 2: Overview

- Layered architecture
- Most SOAP calls are converted into Operation objects
  - Questions and Actions
  - Can be stored for delayed execution (requests)
  - Separate Authorization and execution
- Expressed as a series of manipulations on domain objects
  - Active records
  - No transaction handling



Internals 3: Operation example

- Operation example: create user

```
public class CreateUserAction extends ActionHelper {
    private User user;          // Parameter

    public User getUser () { return user; }

    public void checkPermission() throws VOMSException {
          DBGroup.getVOGroup().checkPermission (Operation.CREATE);
    }

    public void perform() throws VOMSException {
          DBUser u = DBUser.create (user);
          log.info ("User " + u.getDN() + " created");
    }

    public CreateUserAction (User user) {
          this.user = user;
    }
}
```

Internals 4: Execution of Operations

- Transaction handling is separated into a simple façade class
  - Allocates a database connection
  - Starts a new transaction
  - Checks permission
  - Performs the operation
  - Commit/rollback
- Supports transaction restarts
  - Cures database deadlocks
  - Reliability under high load
- Example: (from the SOAP implementation)

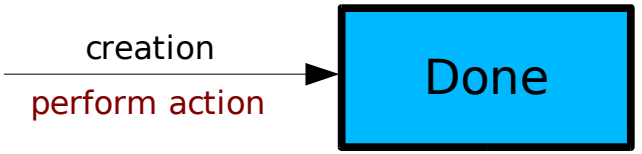
```
public createUser (User user) {
   log.info (“createUser (...)”);
   Database.perform (new CreateUserAction (user));
}
```

Request flow engine

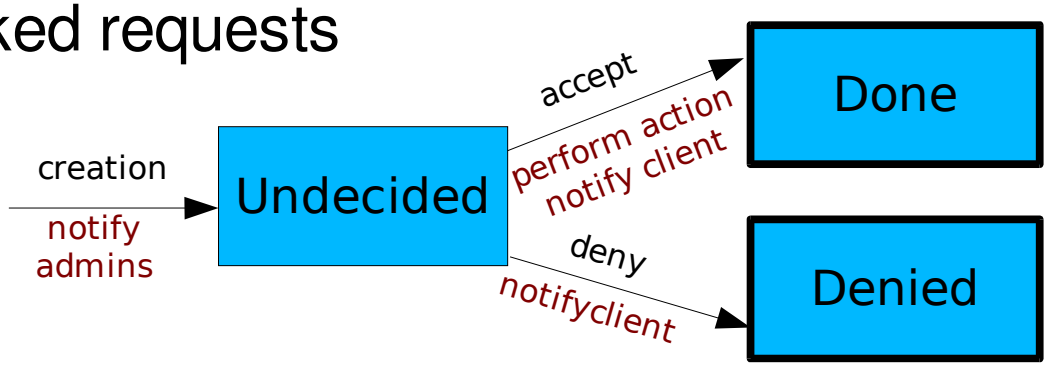
- Requests are entities requesting the execution of an Operation
- Stored in the database as serialized Java objects
  - Flexibility
  - The most important values are indexed for speedy access
- Each request has a request type that defines its workflow
  - State machines
  - Highly reusable components (states, events)
  - The workflow is independent of the operation type
  - Easily extendable
- The SOAP API need not be changed for a new request type
  - Exception: new events need a new API function
- Requests maintain a chronicle of what happened to them

Predefined request types

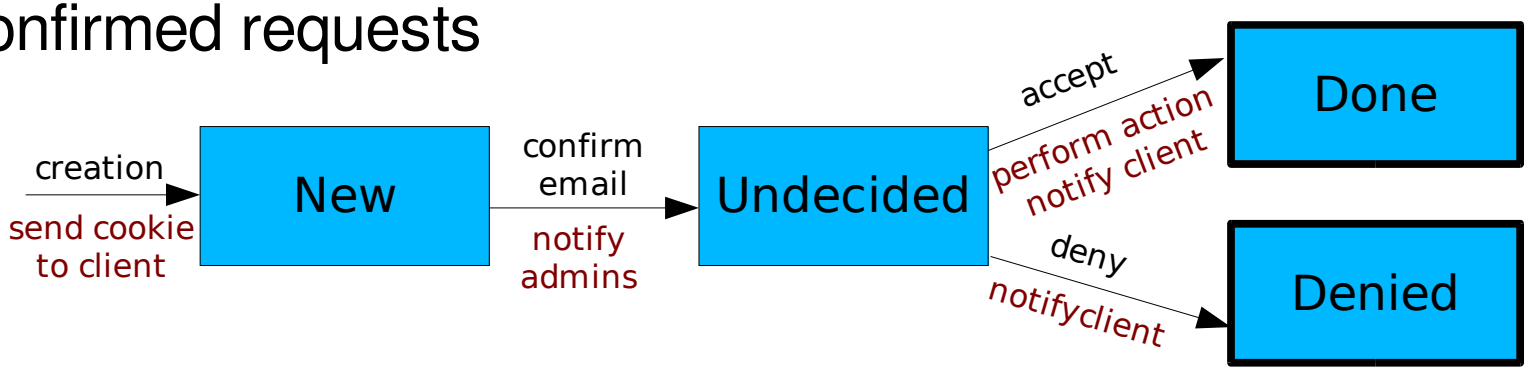
- Automatic requests



- Checked requests



- Confirmed requests



(Timeout and Delete events not shown)

Request examples

- Creating a new request:

```
public long newCreateRequest (final User user, final String comment) {
    log.info ("newCreateRequest (...)");
    Object result = Database.perform (new ActionHelper() {
                // An inline operation:
                public void checkPermission();
                public Object performWithResult() throws VOMSException {
                    Action action = new CreateUserAction (user);
                    Request request = ConfirmedRequest.createRequest
                        (action, comment);
                    return new Long (request.getId());
                }
            });
    return ((Long) result).longValue();
}
```

- Processing an event:

```
public void allowRequest (final long id, final String comment) {
    log.info ("allowRequest (...)");
    Database.perform (new ActionHelper() {
                public void checkPermission() {}  // done by the request type
                public void perform() throws VOMSException {
                    Request r = Request.getInstance (id);
                    r.processEvent (new DenyEvent (comment));
                }
            });
}
```