dCache.org

gPlazma theory

Paul Millar

Zeuthen, 2012

Fermilab NDGF Nordic DataGrid Facility EUROPEAN MIDDLEWARE INITIATIVE DESY HELMHOLTZ ASSOCIATION

Overview

- Background: who are you?
- gPlazma (the old)
- gPlazma (the new)
- Future directions

Who are you?

‘Who are you ?’ said the Caterpillar.

This was not an encouraging opening for a conversation. Alice replied, rather shyly, ‘I— I hardly know, sir, just at present — at least I know who I was when I got up this morning, but I think I must have been changed several times since then.’

Proving who you are: in the “real” world

- You are who you say you are
  - usually too weak when money is involved
- You are who others say you are
  - Typically your parents (shortly after birth),
  - “The state” (or agents thereof) after:
- How do you prove you are this person?
  - You have to look like your passport photo
  - Have recent bills
  - ...

Credential vs Principal

Credentials

Principals

Name: Wile E. Coyote

ACME customer ID: 11493

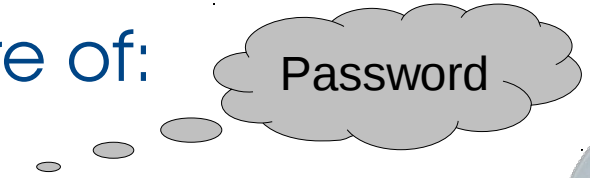
Passport number: 0008103314

Bank account number: 001213921
Banks with: United ACME Bank

Member-of: Antagonists Anonymous

Proving who you are: in computer world

- Based on one or more of:
  - knowing a secret
  - Holding something difficult to fake
  - Some biometric identifier
- Examples:
  - Username + password
  - X509 certificate (+ private key)
  - Kerberos
  - Federated identity:



Shibboleth OpenID OAUTH BrowserID

X509: a quick primer

- Public/private key-pair.
- Certificate, signed by a trustworthy organisation.
  - Chain-of-trust
  - Certificate Revocation List (CRL)
- Proxy certificate
  - short-lived certificate, created by the user.
- VOMS proxy certificate
  - a proxy certificate with group membership embedded.

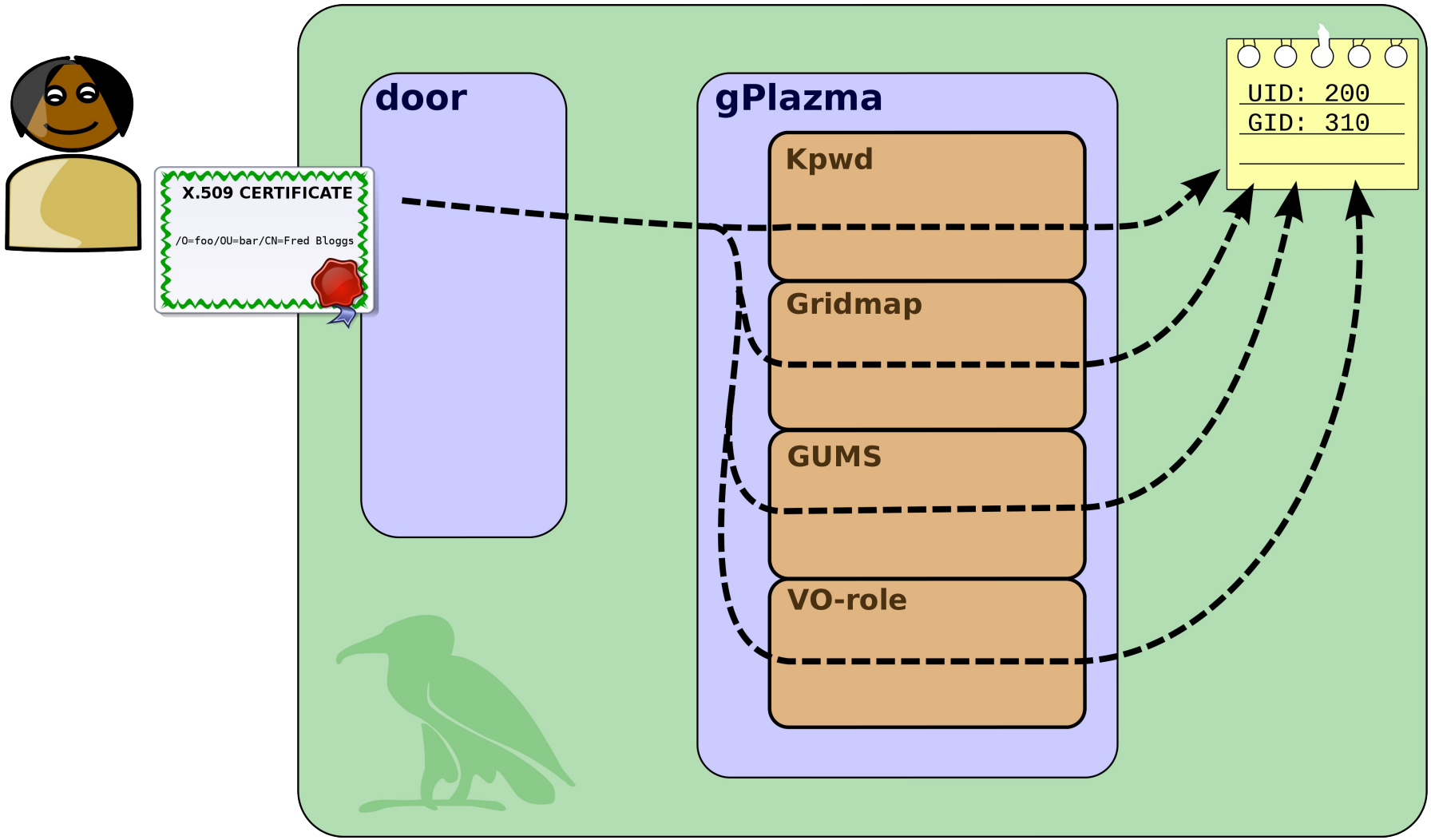
Kerberos: a quick primer

- Central 'KDC' issues users with 'tickets'
  - Ticket for login session, another for each service you use
- Trusted hosts on an untrusted network
  - Mutual authentication
- Trust relationship between services and KDC.
- Login is part of desktop login or with `kinit`

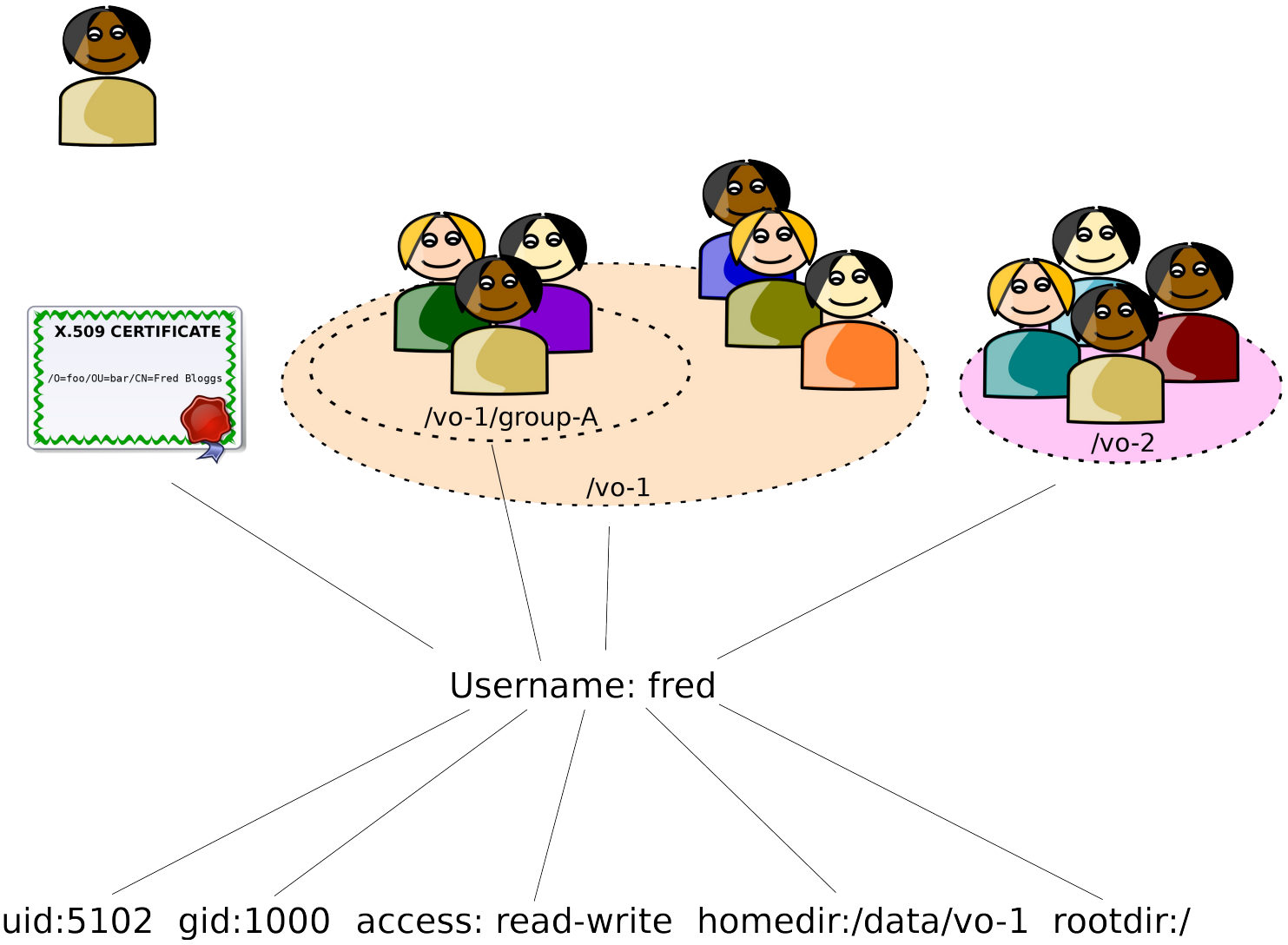
gPlazma v1

An awkward predicament –
W. Heath Robinson

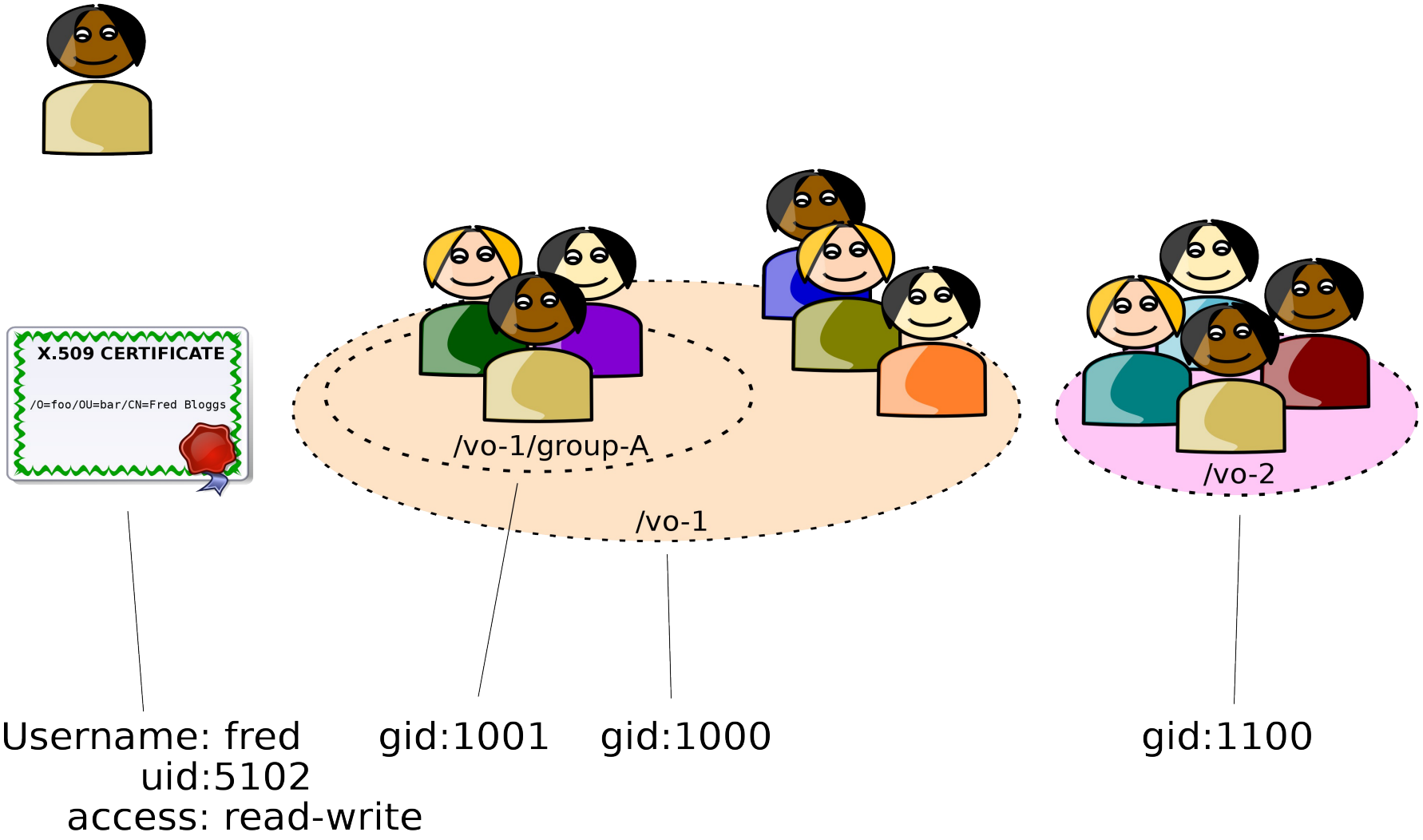
gPlazma v1 structure



gplazmalite-vorole-mapping



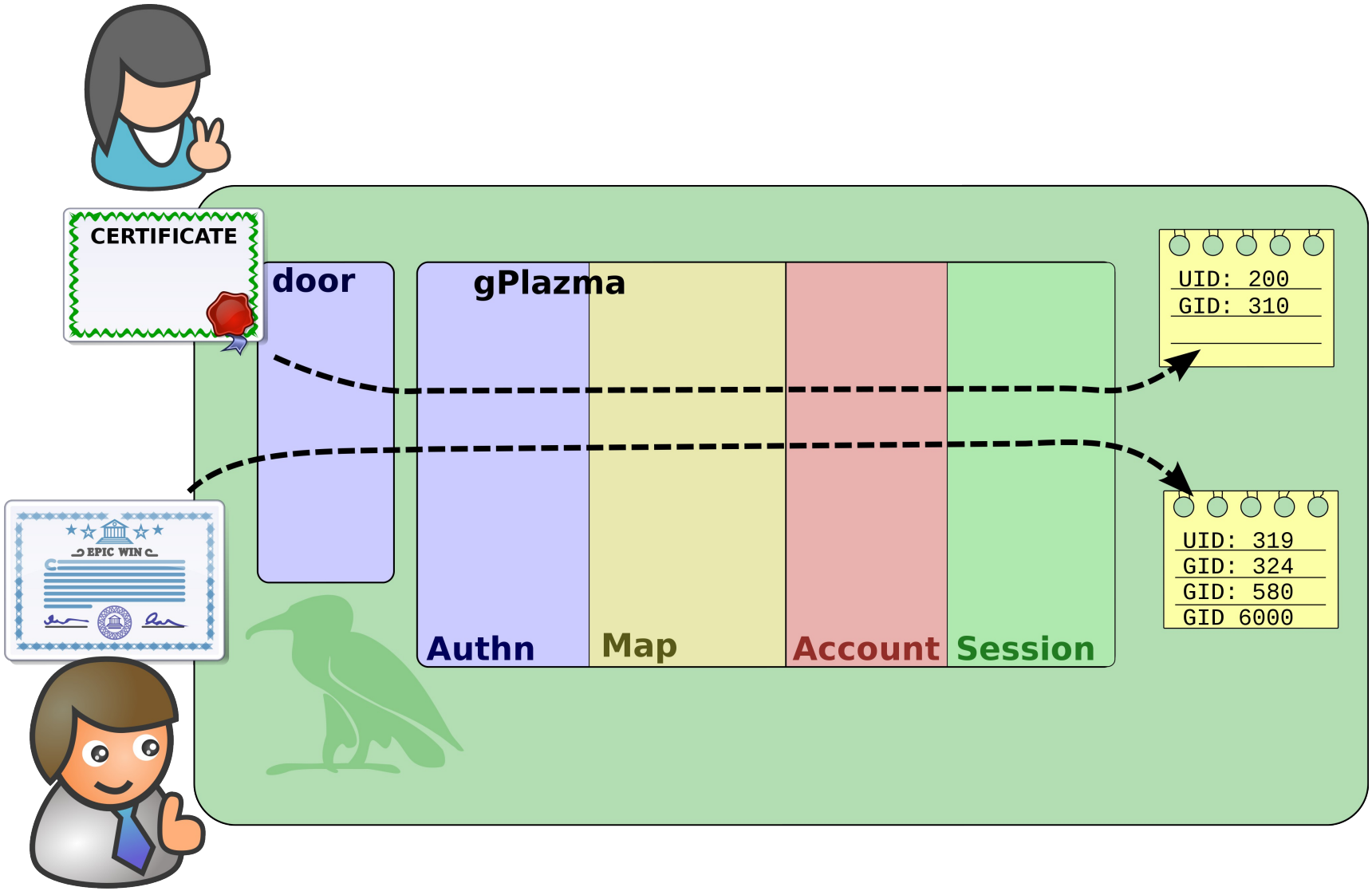
What we would like



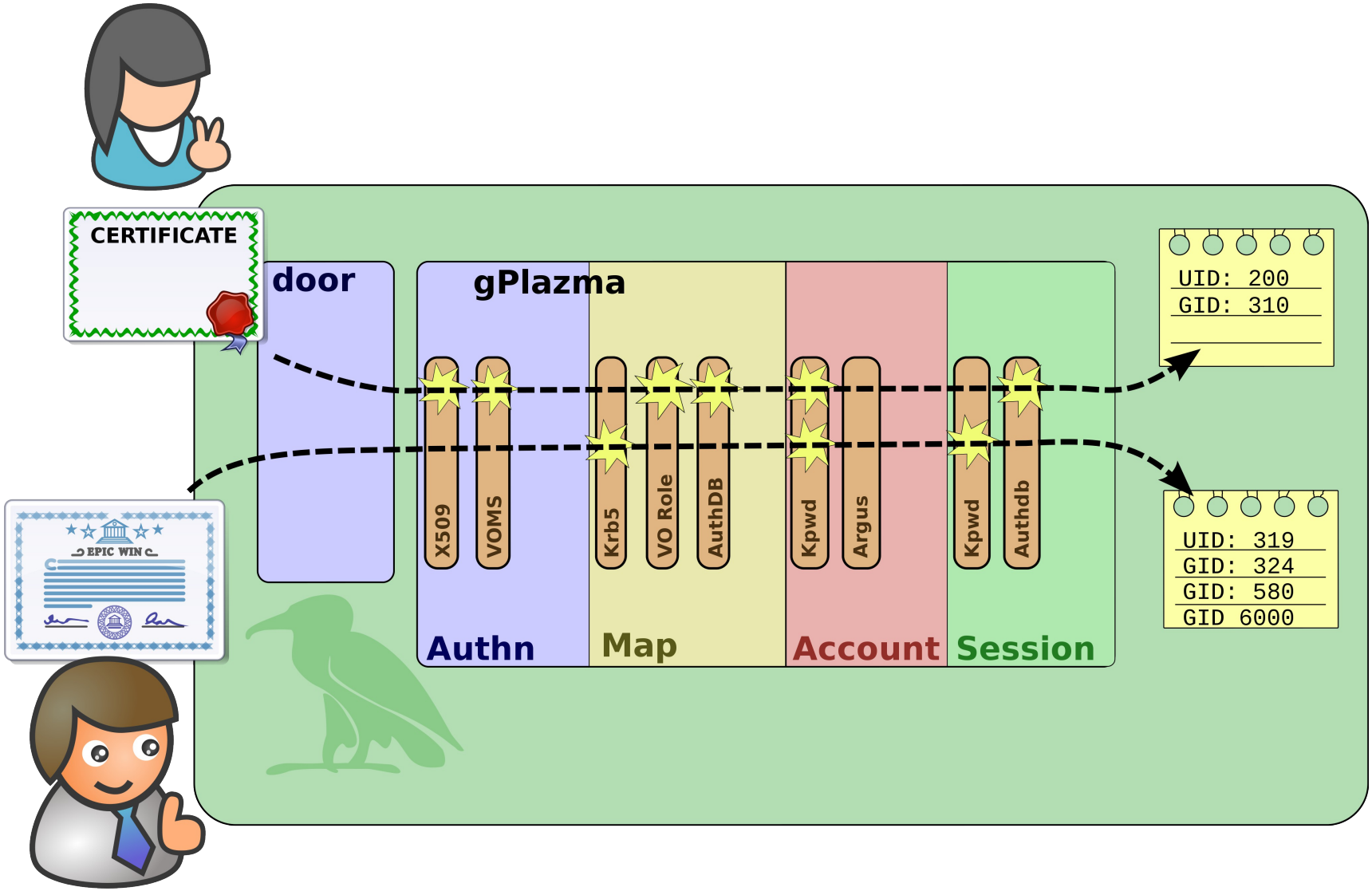
gPlazma v2

"The only way you can predict the future is to build it." -Alan Kay

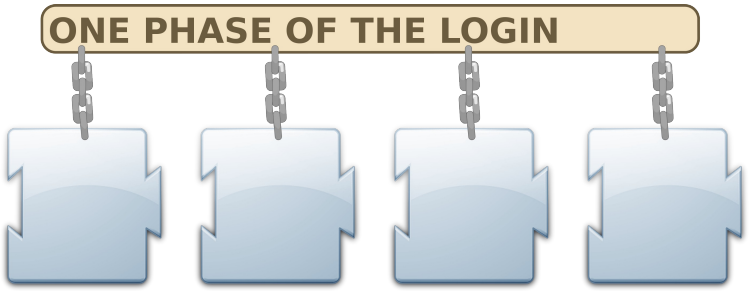
Logging in: the four phases



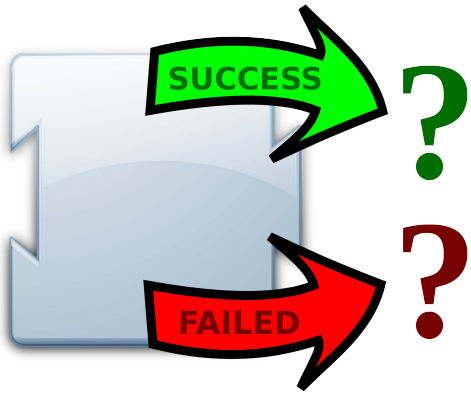
Logging in: plugins



Wiring plugins together



- Each phase is comprised of plugins
  - Result of a phase depends on the result of running these plugins
- When a plugin runs:
  - Running a plugin is either **success** or **failure**.
  - Plugins that fail sometimes is expected
- Four options describe what to do next:



Name	Description
`optional`	Success or failure of the plugin doesn't matter; always move onto next one in the phase
`sufficient`	Successful plugin finishes the phase with success
`requisite`	Failing plugin finishes the phase with failure
`required`	Failing plugin fails the phase but remaining plugins are still run

gPlazma configuration

```
auth    optional    x509
auth    optional    voms
auth    optional    kpwd        gplazma.kpwd.file=/etc/dcache.kpwd

map     optional    krb5
map     optional    vorolemap
map     sufficient  authzdb
map     requisite   kpwd

account requisite   argus

session optional    authzdb
session optional    kpwd
```

gPlazma configuration

First column is required and identifies in which phase the plugin is being configured

```
auth     optional    x509
auth     optional    voms
auth     optional    kpwd        gplazma.kpwd.file=/etc/dcache.kpwd

map      optional    krb5
map      optional    vorolemap
map      sufficient  authzdb
map      requisite   kpwd

account  requisite   argus

session  optional    authzdb
session  optional    kpwd
```

gPlazma configuration

Second column is required and identifies the wiring: what to do next

```
auth    optional   x509
auth    optional   voms
auth    optional   kpwd        gplazma.kpwd.file=/etc/dcache.kpwd

map     optional   krb5
map     optional   vorolemap
map     sufficient authzdb
map     requisite  kpwd

account requisite  argus

session optional   authzdb
session optional   kpwd
```

gPlazma configuration

Third column is required and identifies which plugin is being configured

```
auth    optional   x509
auth    optional   voms
auth    optional   kpwd      gplazma.kpwd.file=/etc/dcache.kpwd

map     optional   krb5
map     optional   vorolemap
map     sufficient authzdb
map     requisite  kpwd

account requisite  argus

session optional   authzdb
session optional   kpwd
```

gPlazma configuration

Final column is optional and provide local configuration. Configuration is normally achieved in dcache.conf and layout file

```
auth    optional    x509
auth    optional    voms
auth    optional    kpwd

map     optional    krb5
map     optional    vorolemap
map     sufficient  authzdb
map     requisite   kpwd

account requisite   argus

session optional    authzdb
session optional    kpwd
```

```
gplazma.kpwd.file=/etc/dcache.kpwd
```

gPlazma configuration

The plugin configuration order is the order in which they are run

x509, voms, kpwd
krb5, vorolemap, authzdb, kpwd
argus
authzdb, kpwd

```
auth     optional    x509\
auth     optional    voms
auth     optional    kpwd        gplazma.kpwd.file=/etc/dcache.kpwd

map      optional    krb5
map      optional    vorolemap
map      sufficient  authzdb
map      requisite   kpwd

account  requisite   argus

session  optional    authzdb
session  optional    kpwd
```

gPlazma configuration

The same plugin may appear multiple times, usually in different phases

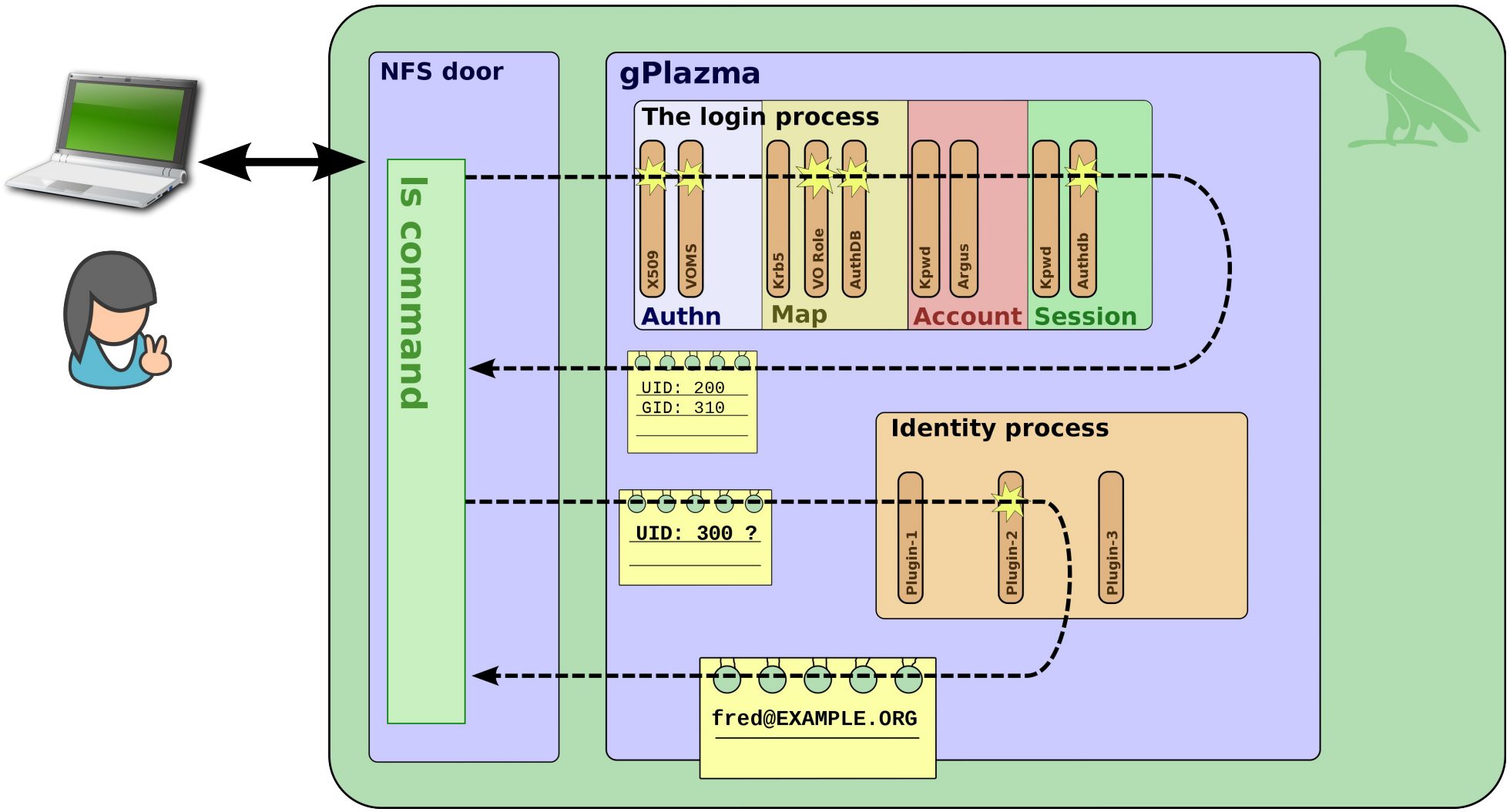
```
auth    optional    x509
auth    optional    voms
auth    optional    kpwd          gplazma.kpwd.file=/etc/dcache.kpwd

map     optional    krb5
map     optional    vorolemap
map     sufficient  authzdb
map     requisite   kpwd

account requisite   argus

session optional    authzdb
session optional    kpwd
```

Something new: identity mapping



gPlazma1 in a gPlazma2 world

- In 1.9.12, there's a switch:

  gPlazma instance is either v1 or v2, but never both.

- In 2.2, there is only gPlazma2:

  Don't panic:

  - gPlazma1 available as a gPlazma2 plugin

    (this is the default configuration)

  - gPlazma2 can be configured to give same behaviour as a configured gPlazma1

    (see next talk for details)

- After 2.2 we will remove v1 completely

Future directions

To infinity ... and beyond!

Future direction

- **Autogenerate uid/gid** for previously unknown principals:

  X509 DNs → uid, FQAN → gid.

- **Discover FQANs** if user didn't provide them

  Not always possible for user to provide FQANs

  Likely only be used for reading.

- Add support for **federated identity** systems.

One final thing..

Life's not always that easy

- Sometimes system is misconfigured
- User is unable to use dCache
  - User is told ***only*** that login failed

    for good security reasons
- Simple logging might not provide enough information, or information in the wrong place at wrong time:
  - gPlazma failures may be difficult to understand
- Need an overview

Introducing gPlazma result printer

Simple example

```
LOGIN FAIL
 |    in: host/zitpcx6184.desy.de@DESY.DE
 |
 +--AUTH OK
 |   |
 |   +--x509 OPTIONAL:FAIL (no X509 certificate chain) => OK
 |   |
 |   +--voms OPTIONAL:FAIL (no X509 certificate chain) => OK
 |   |
 |   +--kpwd OPTIONAL:FAIL (no username and password) => OK
 |
 +--MAP FAIL
 |   |  removed: host/zitpcx6184.desy.de@DESY.DE
 |   |
 |   +--krb5 OPTIONAL:OK => OK
 |   |      added: UserNamePrincipal[host/zitpcx6184.desy.de]
 |   |
 |   +--vorolemap OPTIONAL:FAIL (no record) => OK
 |   |
 |   +--authzdb SUFFICIENT:FAIL (no mapping exists for host/zitpcx6184.desy.de) => OK
 |   |
 |   +--kpwd REQUISITE:FAIL (no login name) => FAIL (ends the phase)
 |
 +--(ACCOUNT) skipped
 |
 +--(SESSION) skipped
 |
 +--(VALIDATION) skipped
```

Simple example

Easy to spot why login failed

```
LOGIN FAIL
 |    in: host/zitpcx6184.desy.de@DESY.DE
 |
 +--AUTH OK
 |    |
 |    +--x509 OPTIONAL:FAIL (no X509 certificate chain) => OK
 |    |
 |    +--voms OPTIONAL:FAIL (no X509 certificate chain) => OK
 |    |
 |    +--kpwd OPTIONAL:FAIL (no username and password) => OK
 |
 +--MAP FAIL
 |    |  removed: host/zitpcx6184.desy.de@DESY.DE
 |    |
 |    +--krb5 OPTIONAL:OK => OK
 |    |       added: UserNamePrincipal[host/zitpcx6184.desy.de]
 |    |
 |    +--vorolemap OPTIONAL:FAIL (no record) => OK
 |    |
 |    +--authzdb SUFFICIENT:FAIL (no mapping exists for host/zitpcx6184.desy.de) => OK
 |    |
 |    +--kpwd REQUISITE:FAIL (no login name) => FAIL (ends the phase)
 |
 +--(ACCOUNT) skipped
 |
 +--(SESSION) skipped
 |
 +--(VALIDATION) skipped
```

Example with simple X509

```
LOGIN FAIL
 |    in: S:/131.169.252.82
 |        X509 Certificate chain:
 |          |
 |          +--CN=Alexander Paul Millar,OU=DESY,O=GermanGrid,C=DE [16724]
 |               |
 |               +--Issuer: CN=GridKa-CA,O=GermanGrid,C=DE
 |               +--Validity: OK for 366 days, 20 hours, 30 minutes and 13.0 seconds
 |               +--Algorithm: SHA-1 with RSA
 |               +--Subject alternative names: paul.millar@desy.de
 |               +--Key usage: digital signature, key encipherment, data encipherment
 |
 |   out: GidPrincipal[1000,primary]
 |        UidPrincipal[1000]
 |        UserNamePrincipal[paul]
 |        /C=DE/O=GermanGrid/OU=DESY/CN=Alexander Paul Millar
 |        KpwdPrincipal[paul]
 |
 +--AUTH OK
 |   |    added: /C=DE/O=GermanGrid/OU=DESY/CN=Alexander Paul Millar
 |   |
 |   +--x509 OPTIONAL:OK => OK
 |   |      added: /C=DE/O=GermanGrid/OU=DESY/CN=Alexander Paul Millar
 |   |
 |   +--voms OPTIONAL:FAIL (no FQANs) => OK
 |   |
 |   +--kpwd OPTIONAL:FAIL (no username and password) => OK
 |
```

Example with voms proxy

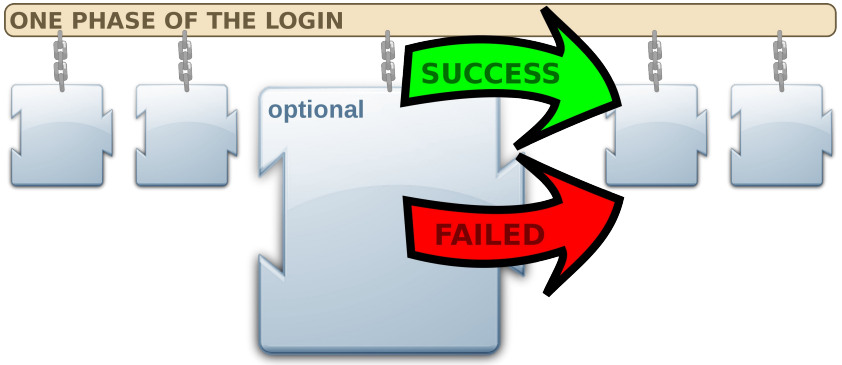
```
LOGIN FAIL
 |    in: S:/131.169.137.140
 |        /C=DE/ST=Hamburg/O=dCache.ORG/CN=Kermit the frog
 |        X509 Certificate chain:
 |          |
 |          +--CN=proxy,CN=Kermit the frog,O=dCache.ORG,ST=Hamburg,C=DE [11549466642107437257]
 |          |    |
 |          |    +--Issuer: CN=Kermit the frog,O=dCache.ORG,ST=Hamburg,C=DE
 |          |    +--Validity: OK for 11 hours, 59 minutes and 42.0 seconds
 |          |    +--Algorithm: SHA-1 with RSA
 |          |    +--Attribute certificates:
 |          |    |    |
 |          |    |    +--C=DE,O=GermanGrid,OU=DESY,CN=host/grid-voms.desy.de
 |          |    |         +--Validity: OK for 11 hours, 59 minutes and 42.0 seconds
 |          |    |         +--Algorithm: SHA-1 with RSA
 |          |    |         +--FQANs: /desy, /desy/workshop
 |          |    +--Key usage: digital signature, key encipherment, data encipherment, key agreement
 |          |
 |          +--CN=Kermit the frog,O=dCache.ORG,ST=Hamburg,C=DE [11549466642107437257]
 |          |    |
 |          |    +--Issuer: CN=dCache.ORG CA,O=dCache.ORG,ST=Hamburg,C=DE
 |          |    +--Validity: OK for 357 days, 10 hours, 23 minutes and 52.0 seconds
 |          |    +--Algorithm: SHA-1 with RSA
 |          |    +--Subject alternative names: kermit.the.frog@dcache.org
 |          |    +--Key usage: digital signature, key encipherment, data encipherment, key agreement
 |          |
 |          +--CN=dCache.ORG CA,O=dCache.ORG,ST=Hamburg,C=DE [11549466642107437183] (self-signed)
 |               |
 |               +--Validity: OK for 866 days, 12 hours, 13 minutes and 49.0 seconds
 |               +--Algorithm: SHA-1 with RSA
 |               +--Key usage: key certificate signing, CRL signing
 |
```

Thank you and good night!

For readers at home, there is more information in following slides ...

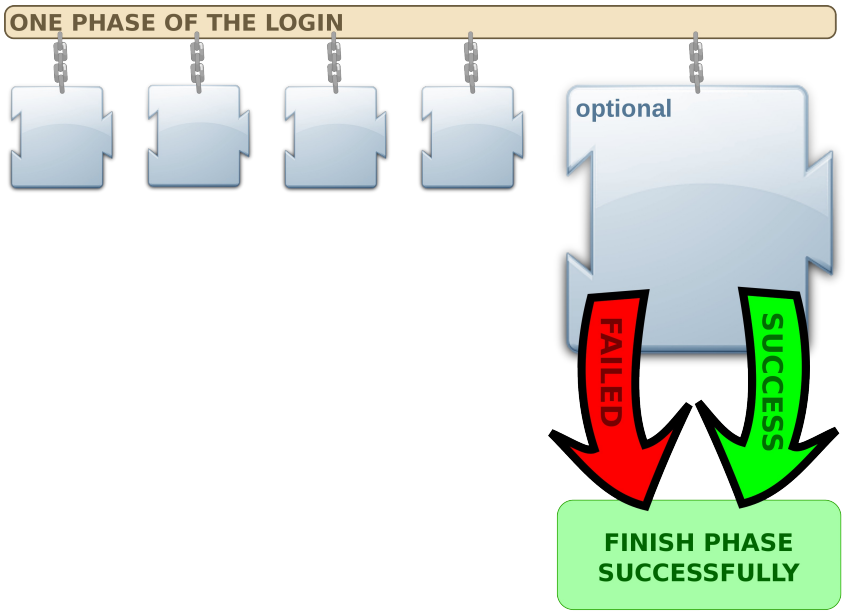
gPlazma wiring in detail

Wiring plugins together: optional



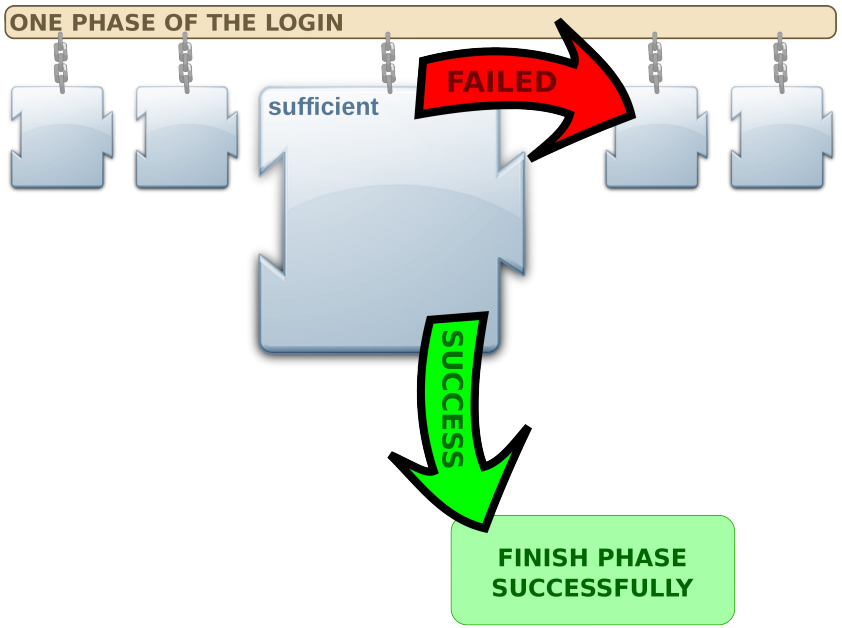
- If a plugin is `optional` then always move on to the next plugin

Wiring plugins together: optional



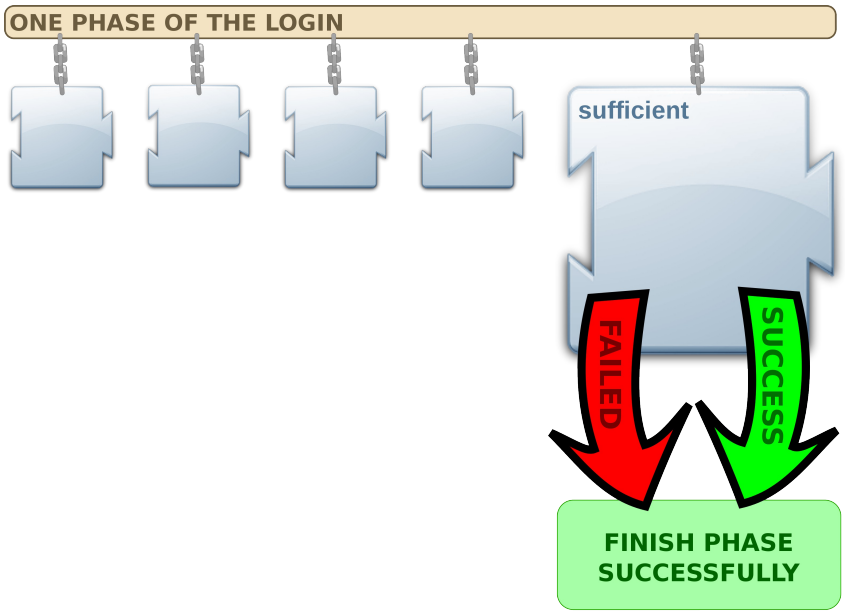
- If the last plugin is `optional`, the phase always succeeds

Wiring plugins together: sufficient



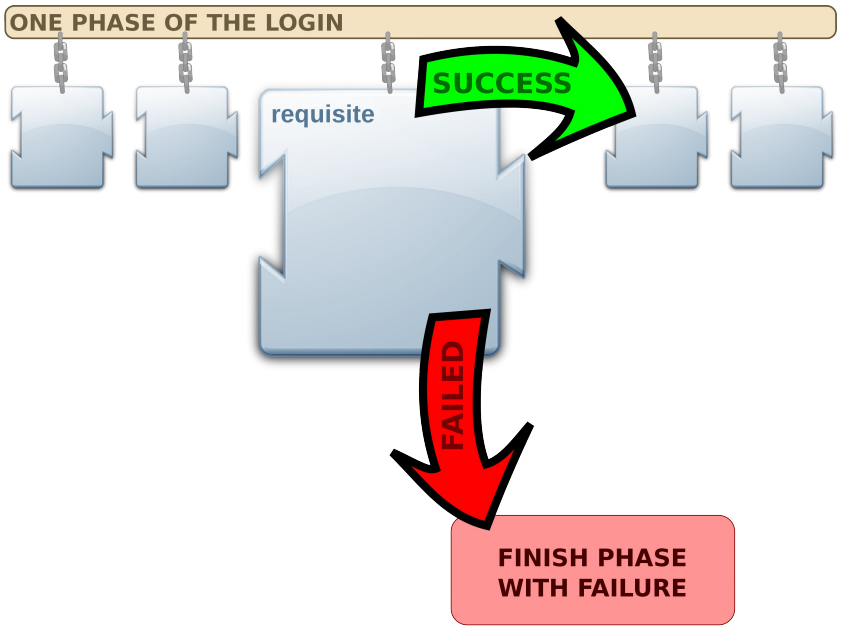
- If a `sufficient` plugin is successful, then end the phase immediately
- If a `sufficient` plugin fails, move on to the next plugin

Wiring plugins together: sufficient



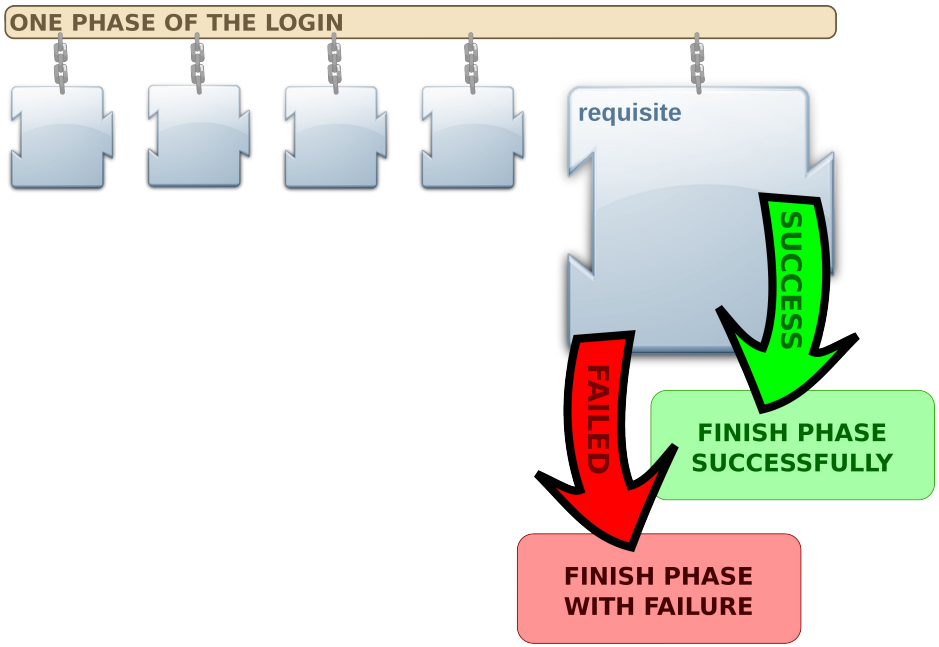
- If the last plugin in the phase is `sufficient` then the phase will always be successful.

Wiring plugins together: requisite



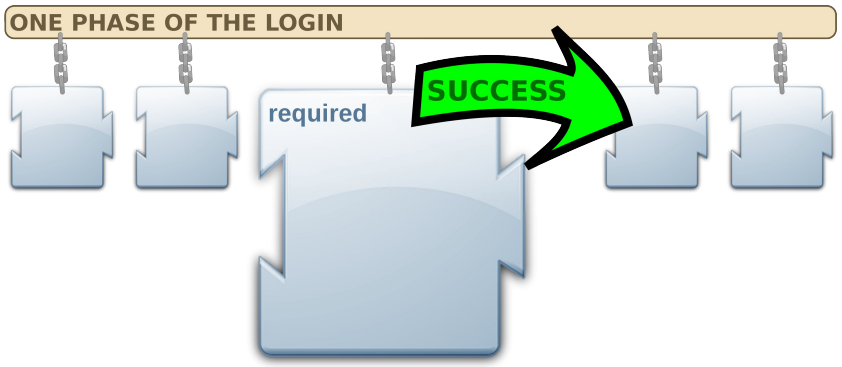
- If a `requisite` plugin fails then fail the phase immediately.
- If a `requisite` plugin is successful, continue to the next plugin

Wiring plugins together: requisite



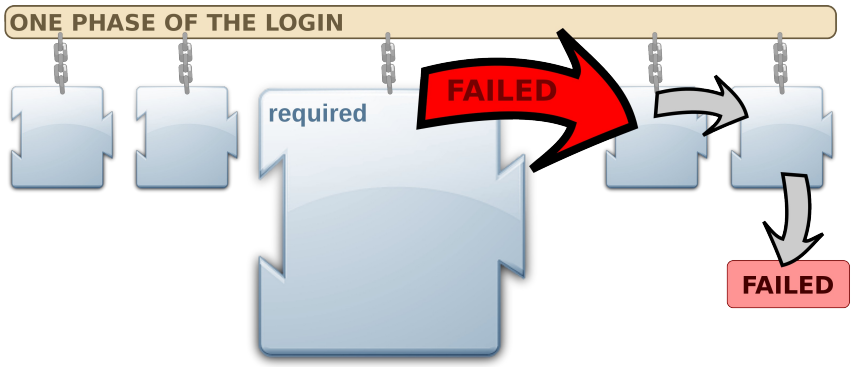
- If the last plugin is `requisite` then the success of the phase depends on the success of the last plugin

Wiring plugins together: required



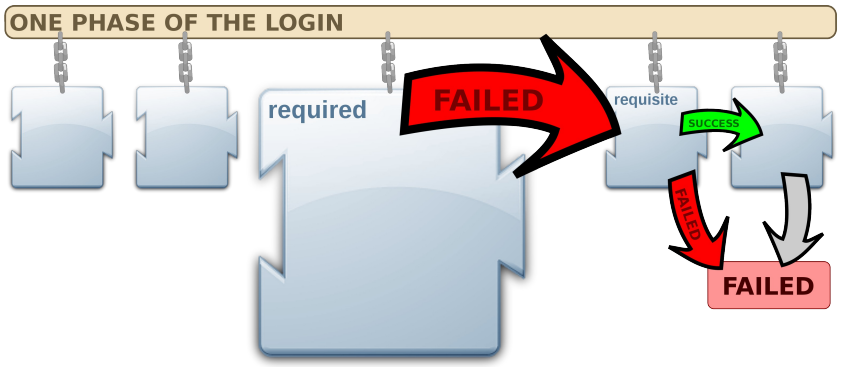
- If a `required` plugin succeeds then move onto the next plugin

Wiring plugins together: required



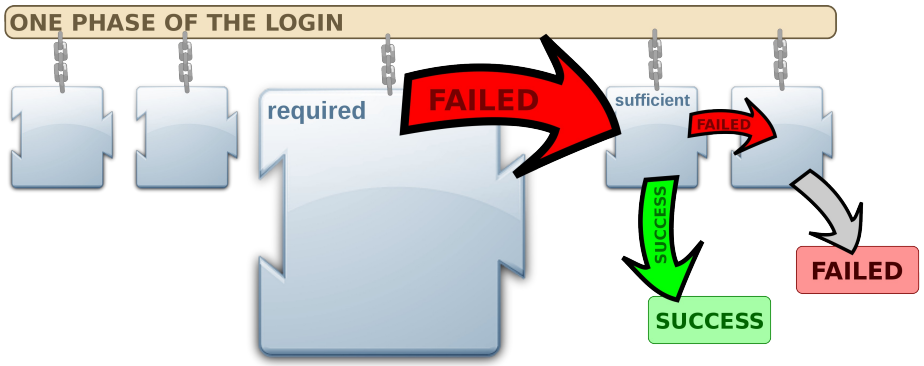
- If a `required` plugin fails then continue onto the next plugin, but ultimately fail with an error from this plugin
- unless ...

Wiring plugins together: required



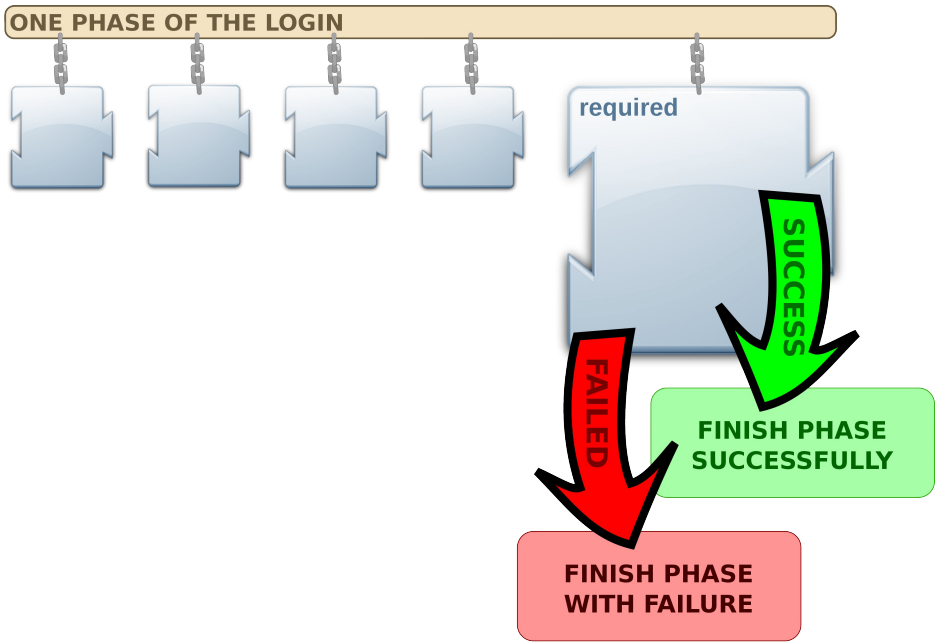
- If one of the plugins after the failing `required` plugin is `requisite` and also fails then immediately fail the phase with the error from the failing `required` plugin.

Wiring plugins together: required



- If one of the plugins after the failing `required` plugin is `sufficient` and succeeds then end the phase successfully.

Wiring plugins together: required



- If the last plugin is `required` then the success of the phase depends on the success of the final plugin

Backup slides

Kerberos in more detail