Penn
Groups

PennGroups

Central Authorization System
January 2009

Penn Profile

- **Private research university founded in 1740**
- **259 buildings, 283 acres located in West Philadelphia**
- **10,345 undergraduates; 12,103 graduate and professional students (as of Fall 2007) enrolled into twelve graduate/professional schools**
- **Over 20,000 employees, including 14,000+ in University Health System**
- **University (including health system) operating budget of four billion dollars**
- **Central IT in a decentralized environment**
  - **Twelve schools and multiple administrative centers operate with autonomy**
  - **Most schools and centers have their own IT department**
  - **Central IT provides university-wide applications and infrastructure**

Identity Management at Penn

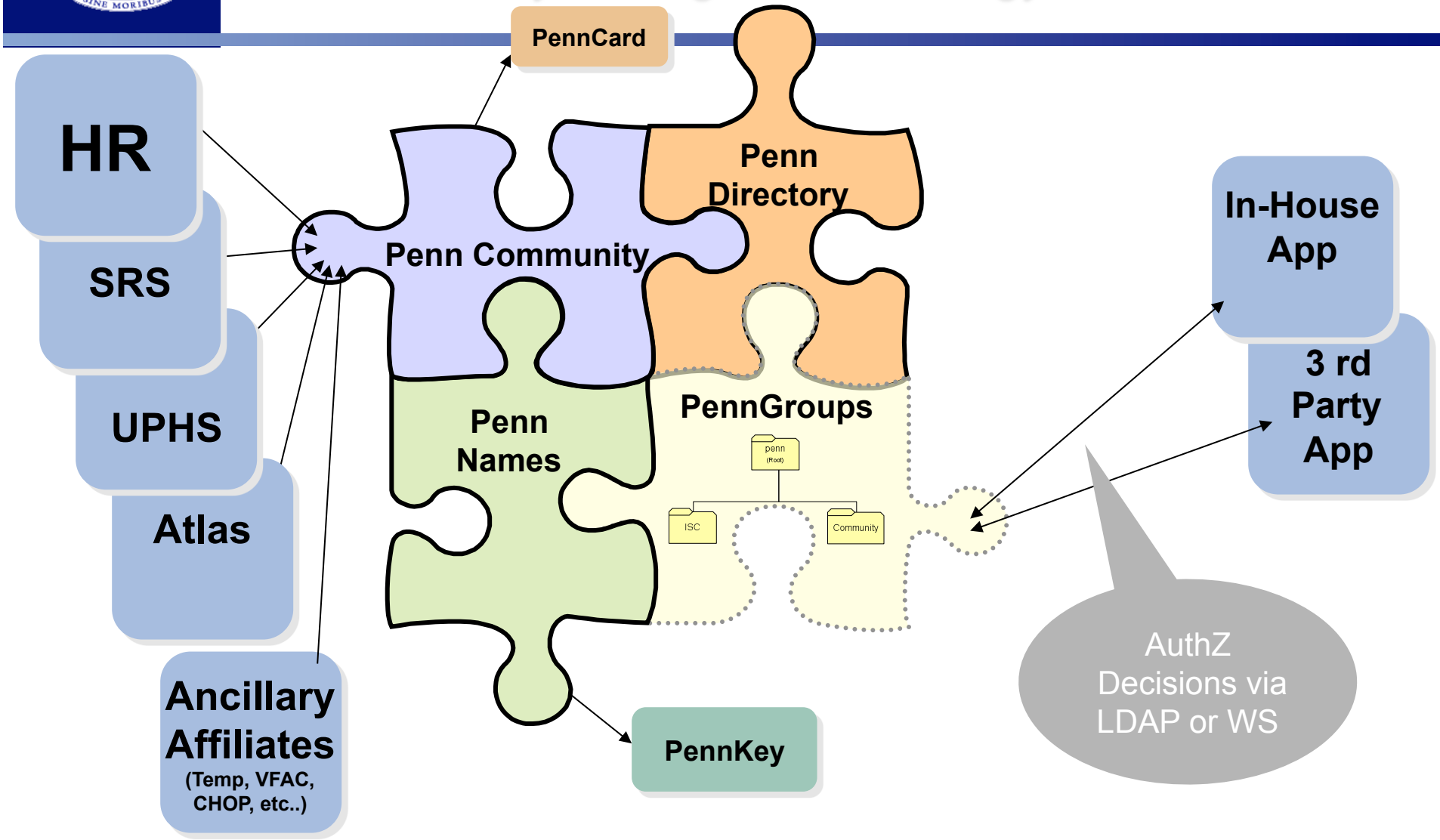
- Goal: To increase protection of the confidential and sensitive information at Penn by:
  - Uniquely identifying entities associated with Penn
  - Providing access to appropriate facilities, services, and systems
  - Preventing unauthorized access to facilities, services, and systems

Elements of Identity Management

Components of identity management

- Penn Community – central repository for a person's bio/demo data as fed by core business systems (SRS, HR/Payroll, Atlas, UPHS) and entered directly for ancillary affiliates
- Penn Directory – system that holds the preferred name and contact info for all Penn affiliates
- Penn Card – system used to generate the physical ID card that is used for building access and commercial transactions across the university
- PennNames - system used to associate a unique username to each individual at Penn, providing a common and consistent University namespace for online services
- PennKey – unique identifier for Penn's central authentication system; with associated password, provides an electronic means to authenticate an individual and provide access to systems across the university
- PennGroups – system for creating and managing groups to facilitate authorization decisions by applications with hooks to LDAP or web services

Penn's Identity Management Strategy



What Is PennGroups

- PennGroups is derived from the Internet2 open source Grouper initiative
- Has been adopted and deployed at many other universities (Brown, Cornell, Yale)
- Penn has worked with the Grouper team to enhance the baseline product (UI, web services, SQL loaded groups)
  - Better meets the needs of Penn
  - Provides additional useful functionality to other grouper users
  - Allows Penn to benefit from future grouper enhancements without maintaining a separate source code instance

Benefits

- Facilitates consistent application of University business rules
  - Managed through a common UI and web services
- Streamlines maintenance of authorization data
  - Brings scattered redundant groups together for re-use
  - Allows useful actions on these groups -- group math, group nesting, exclusion criteria
- Leverages Penn Community data for accurate, up to date authorization decisions
  - Can leverage existing attribute information
- Distributed/delegated model of control
  - Supports the creation of new groups by schools and centers

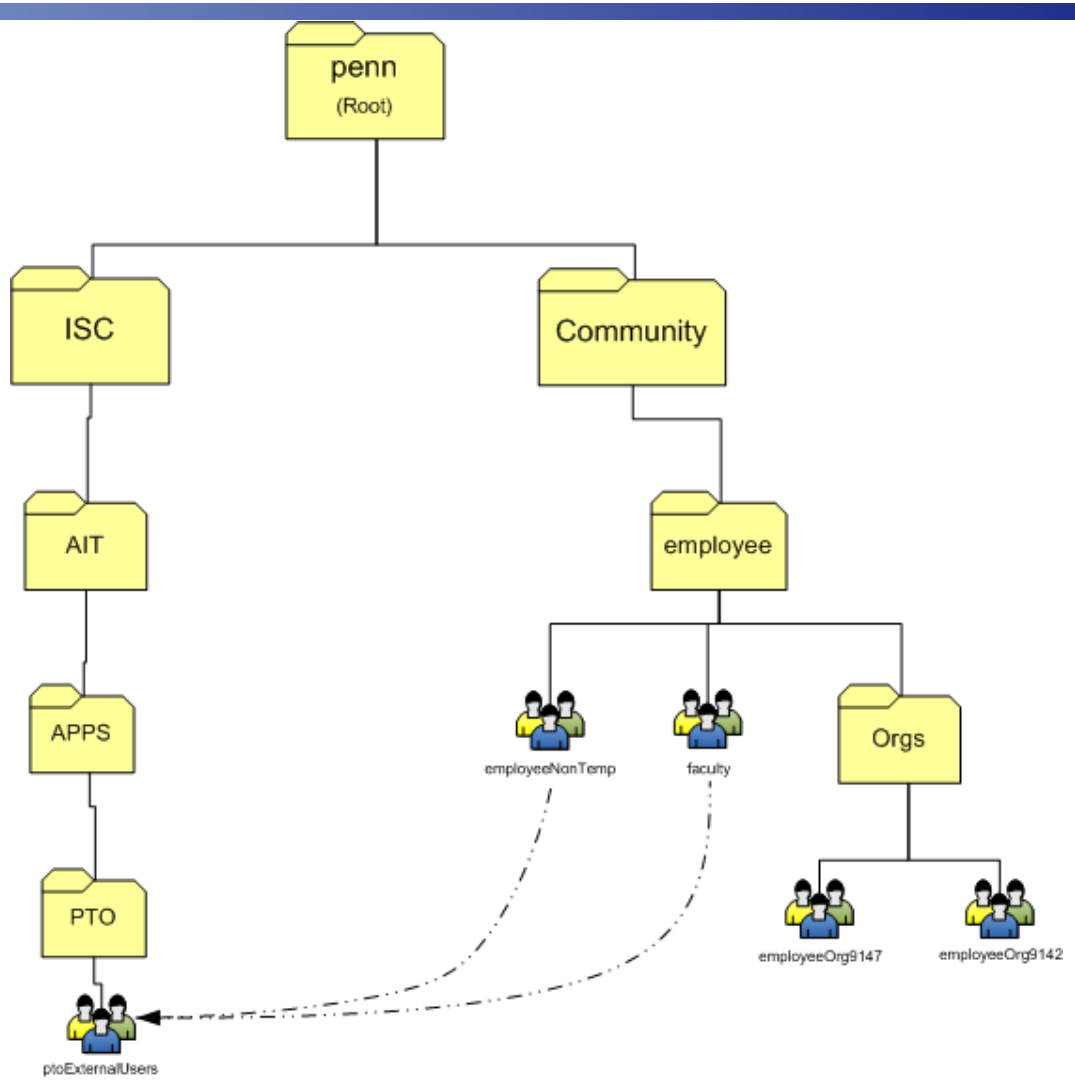
How It Works

- Authorization by application
- After authentication the application can interrogate PennGroups for access to group membership data
  - Web services
  - LDAP
- Changes to group membership are reflected automatically and propagate to the application dynamically

Managing PennGroups

- Two modes for creating and managing groups
  - Automated
    - Web services - build and run a query from your data store and send group membership information to PennGroups via the web service API
    - SQL loaded groups– Configure a SQL query within the PennGroups UI to run on a scheduled basis to modify group membership
  - Manual
    - UI – log onto the PennGroups UI to manually manage your group membership
      - You cannot manually add members to or remove members from a group that is managed in an automated fashion
      - You can simulate this with include/exclude composite groups

PennGroups Hierarchy



PennGroups in a Decentralized Environment

- When School/Center is purchasing or developing a new system
  - LSP (local support provider)/ application developer contacts Central IT
  - LSP/developer and Central IT collaborate to:
    - Establish authorization use cases for the specific application
    - Determine access method (LDAP or Web Services)
    - Determine best approach for group creation and maintenance
  - School/Center fills out access forms
  - Central IT consults with LSP/developer on group hierarchy structure

Use Cases

- PTO – Paid Time Off
  - Self service system used to request/track vacation/sick time
  - Penn Groups provides the flexibility so that the user selects their approver for time off.
  - Time off can be routed and approved by other than a direct supervisor
- Warehouse Apps
  - Penn groups provides a feed for org based security based on active status
- Abramson's Cancer Center
  - Builds custom research related applications and needs a means to confirm that users who log in currently have an active status
- School of Engineering and Applied Science
  - Affiliate level groups - faculty members, staff members, students, undergrads, grads, PhD students
  - Class level groups - everyone enrolled in every SEAS course, and several ad-hoc groups.
  - Kept up to date via a SEAS data store and propagated to PennGroups via the SQL loader
  - Group hierarchy (groups such as freshman, sophomore, etc are members in the group uGrad).
  - Ad hoc groups generated and maintained via specific applications and business rules.
  - Use of groups to determine access to various resources such as SSH (with different groups allowed to access different machines), IMAP, POP, SMTP, etc.

Penn
UNIVERSITY of PENNSYLVANIA

Penn Groups

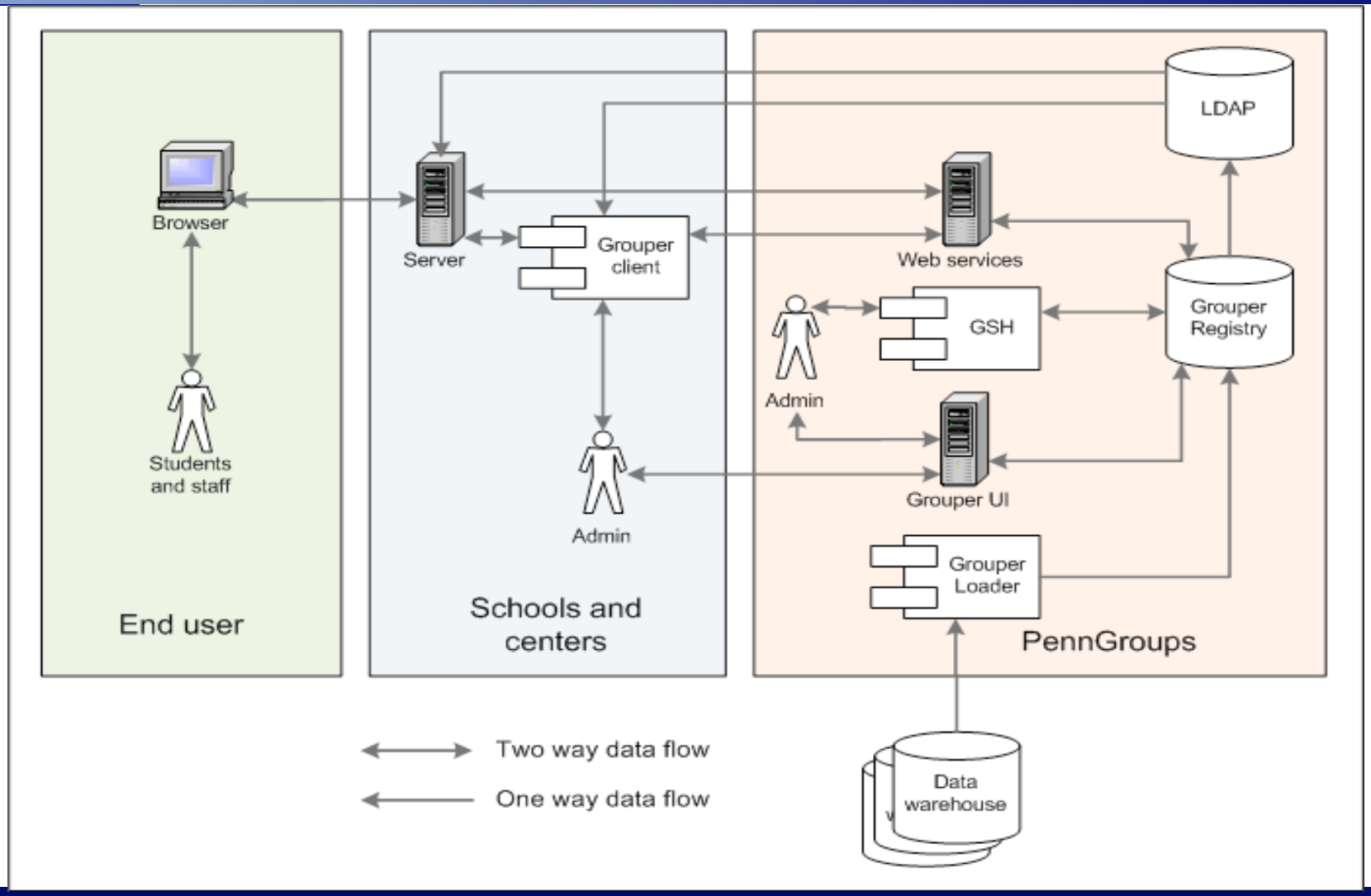
PennGroups

Technical Discussion

Agenda (note: additional information in slide notes)

- PennGroups architecture
- User interfaces
- Web services
- LDAP
- Grouper client
- Grouper loader
- What's new with Grouper in 1.4
  - Configuration checking
  - Daily report

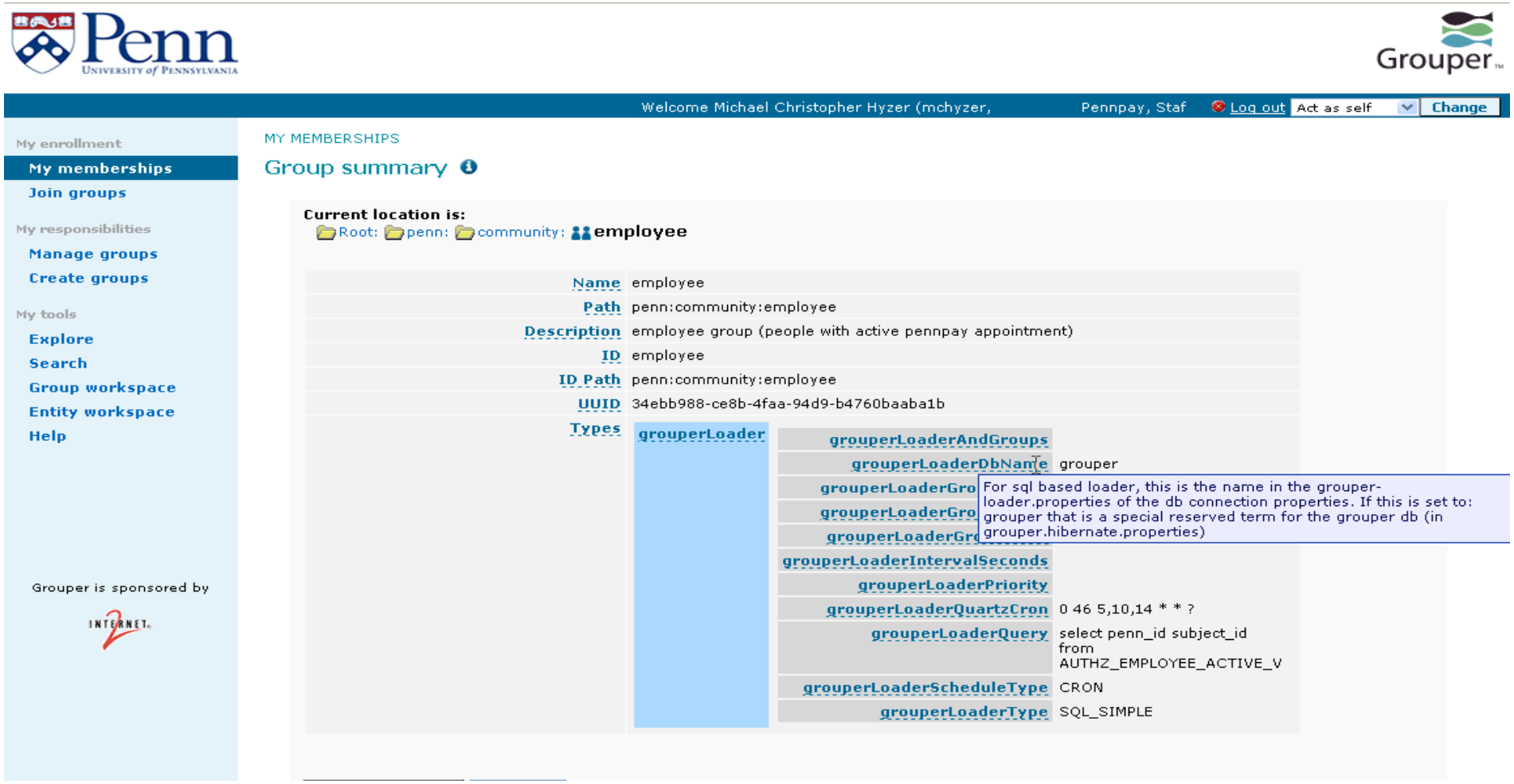
PennGroups architecture



Grouper user interface

- Grouper has a built in user interface
- Penn generally uses the default UI, though:
  - We customized the authentication to use Penn's single signon
  - We added custom code to require users to be in a grouper group to be able to log in (not everyone allowed)
- Penn did a facelift for the Grouper 1.3 release in Spring 2008, improving the usability and help documentation
- For Grouper 1.4 in January 2009, we added the ability to have tooltips on types and attributes

Grouper user interface (continued)



Grouper user interface (continued)

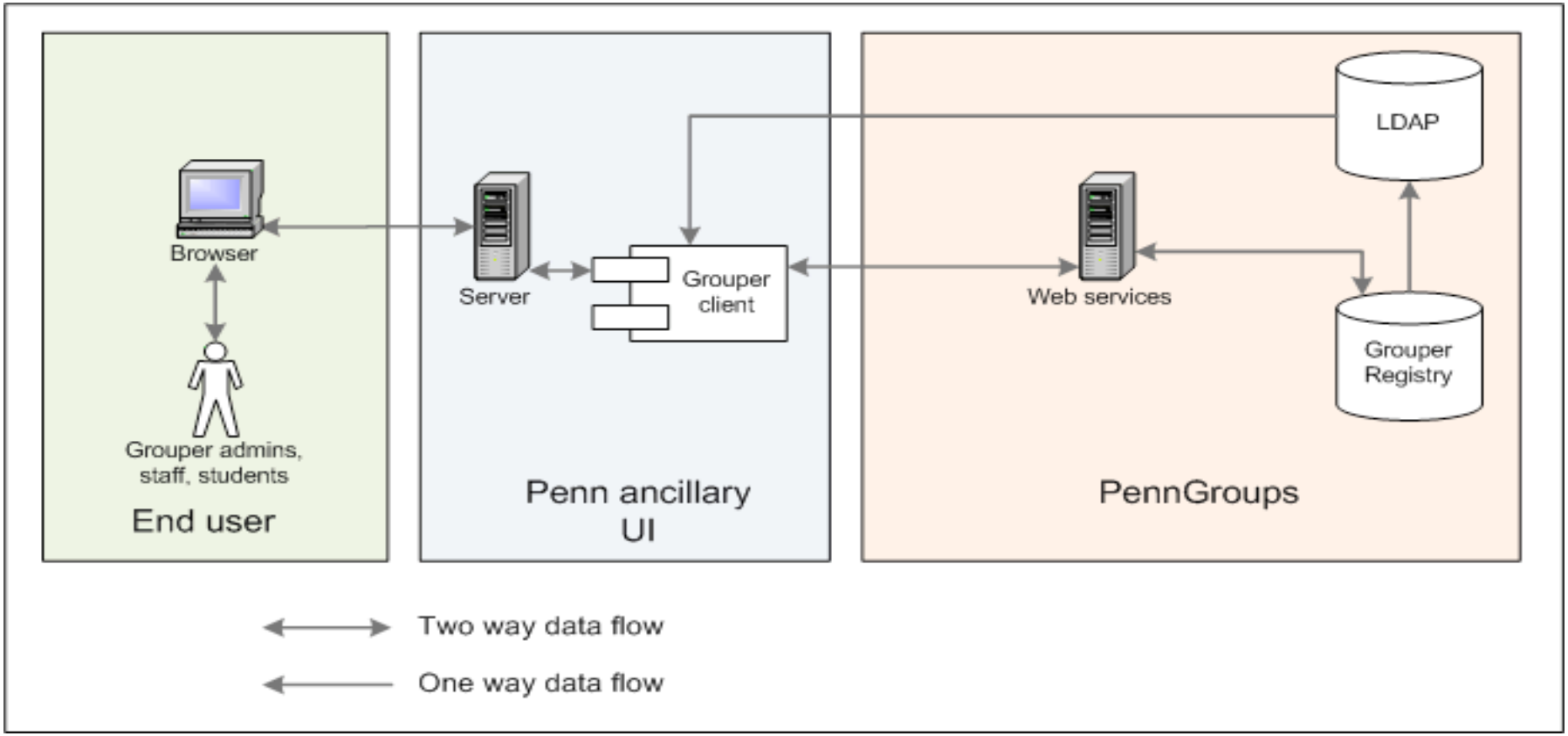
- Tooltips configured in nav.properties

nav.properties

```
groups.summary.types=Types

#prefixes for messages
message.Message=Note:
message.ErrorMessage=Error:
message.WarningMessage=Warning:

tooltipTargetted.groupTypes.grouperLoader=Group membership automatically manage
tooltipTargetted.groupFields.grouperLoaderDbName=For sql based loader, this is
tooltipTargetted.groupFields.grouperLoaderIntervalSeconds=If a START_TO_START_I
tooltipTargetted.groupFields.grouperLoaderPriority=The loader has a max number
tooltipTargetted.groupFields.grouperLoaderQuartzCron=Quartz cron-like string (:
tooltipTargetted.groupFields.grouperLoaderQuery=This is the query to run in the
tooltipTargetted.groupFields.grouperLoaderScheduleType=CRON: This is a cron-lik
tooltipTargetted.groupFields.grouperLoaderType=SQL_SIMPLE: a group whose member
tooltipTargetted.groupFields.grouperLoaderAndGroups=If you want to restrict mer
tooltipTargetted.groupFields.grouperLoaderGroupTypes=If you want types assigned
tooltipTargetted.groupFields.grouperLoaderGroupsLike=If you want the group (if
```

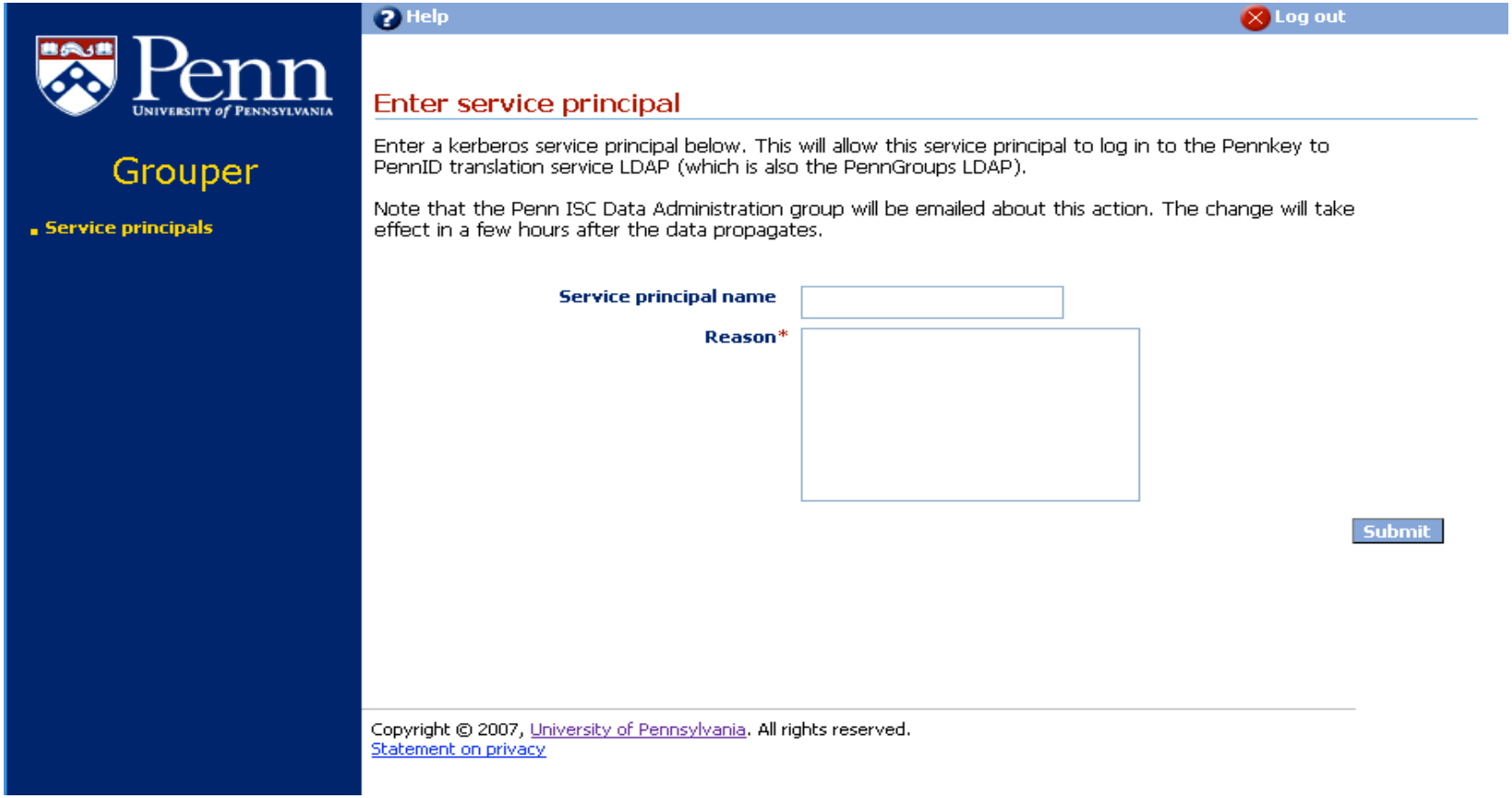
Penn's ancillary Grouper user interface

- For PennGroups tasks not included in Grouper, we have an ancillary UI for Grouper



Penn's ancillary Grouper user interface (continued)

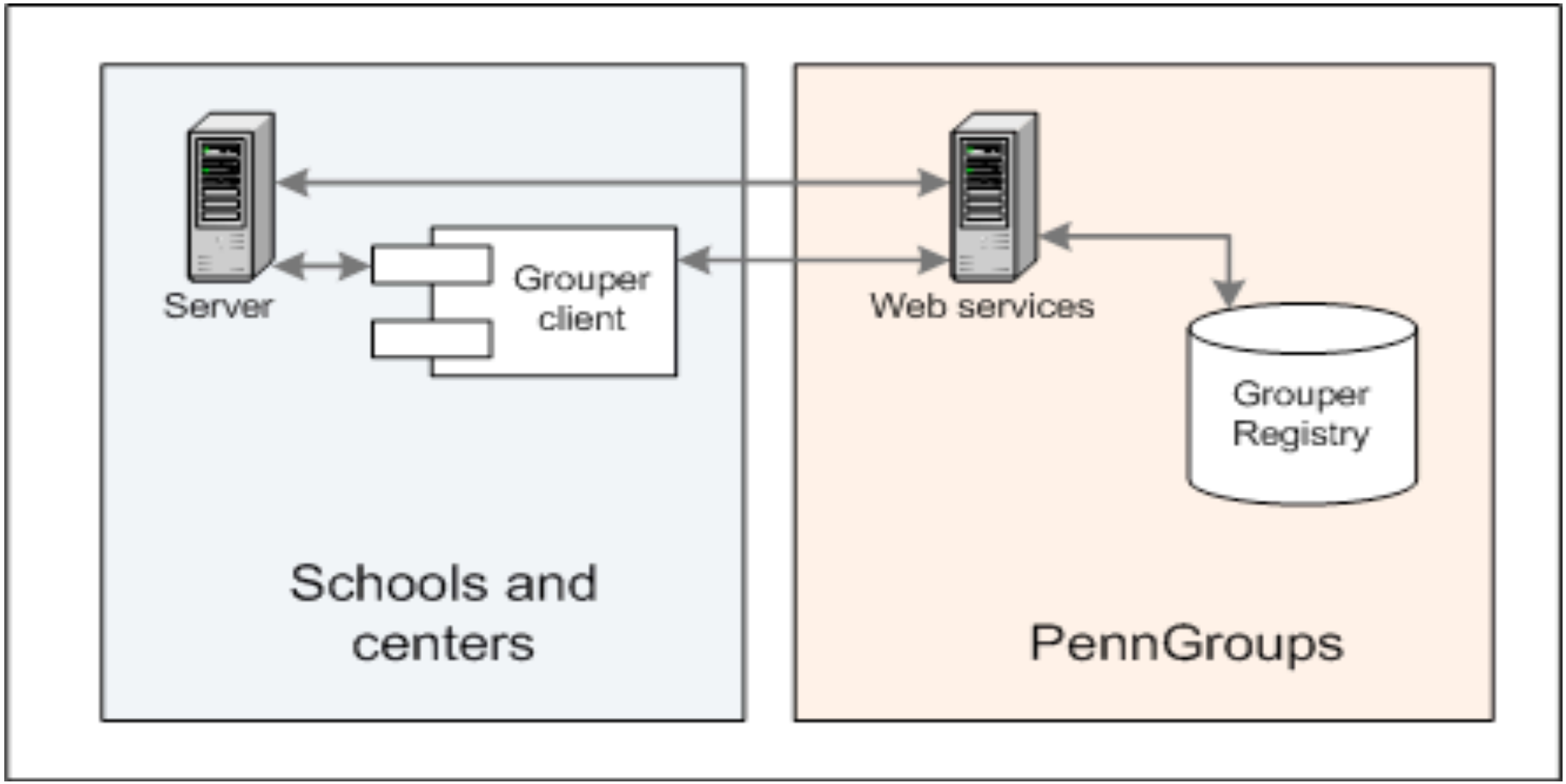
- Currently we only have one task, registering an LDAP login



Grouper web services

- Penn/Internet2 spent a lot of effort in winter/spring 2008 to help create the Grouper web services
- They can be REST or SOAP
- They can be simple “Lite” calls, or batched
- REST accepts formats: XML, XHTML, JSON, HTTP params
- There are a dozen operations exposed, including managing:
  - Groups
  - Memberships
  - Permissions
  - Folders
- Penn uses HTTP credentials sent to kerberos and penn:etc:webServiceUsers group required for authorization

Grouper web services (continued)



Grouper web services (continued)

- There are hundreds of samples to manage
- Custom sample generator is a harness which runs all samples, and stores them in CVS:
  - Listens on TCP port, forwards to web service
  - Makes web service request to the listener
  - Captures request and response
    - Indents the XML or JSON
    - Masks sensitive data (e.g. authentication credentials)
  - Captures stdout and stderr
  - Collates everything including source of sample, saves file in CVS
  - Runs each sample for all different formats, web service types, etc.
  - 163 total sample files

Grouper web services (continued)

[I2MI] / grouper-ws / grouper-ws / doc / samples / addMember Repository: I2MI

Index of /grouper-ws/grouper-ws/doc/samples/addMember

Files shown: **13** (Show 17 dead files)

Sticky Tag: Set

File	Rev.	Age	Author	L
Parent Directory				
WsSampleAddMemberLite_soap.txt	1.10	5 days	mchyzer	1
WsSampleAddMemberRestLite2_withInput_http_xhtml.txt	1.8	5 days	mchyzer	1
WsSampleAddMemberRestLite2_withInput_json.txt	1.8	5 days	mchyzer	1
WsSampleAddMemberRestLite2_withInput_xhtml.txt	1.8	5 days	mchyzer	1
WsSampleAddMemberRestLite2_withInput_xml.txt	1.8	5 days	mchyzer	1
WsSampleAddMemberRestLite_http_xhtml.txt	1.9	5 days	mchyzer	1
WsSampleAddMemberRestLite_json.txt	1.10	5 days	mchyzer	1
WsSampleAddMemberRestLite_xhtml.txt	1.10	5 days	mchyzer	1
WsSampleAddMemberRestLite_xml.txt	1.9	5 days	mchyzer	1
WsSampleAddMemberRest_json.txt	1.10	5 days	mchyzer	1
WsSampleAddMemberRest_xhtml.txt	1.10	5 days	mchyzer	1
WsSampleAddMemberRest_xml.txt	1.10	5 days	mchyzer	1
WsSampleAddMember_soap.txt	1.10	5 days	mchyzer	1

Grouper web services (continued)

Parent Directory | Revision Log

Revision **1.10** - (**download**) (**annotate**)
Mon Dec 15 09:07:26 2008 UTC (5 days, 21 hours ago) by *mchyzer*
Branch: **MAIN**
CVS Tags: **GROUPER_WS_1_4_0, HEAD**
Changes since **1.9: +34 -24 lines**

```
1.4

Grouper web service sample of service: addMember, WsSampleAddMemberLite, code generated classes,


##########################################
##
## HTTP request sample (could be formatted for view by
## indenting or changing dates or other data)
##
##########################################


POST /grouperWs/services/GrouperService HTTP/1.1
Content-Type: application/soap+xml; charset=UTF-8; action="urn:addMemberLite"
User-Agent: Axis2
Authorization: Basic xxxxxxxxxxxxxxxx==
Host: localhost:8092
Transfer-Encoding: chunked

3f2
<?xml version='1.0' encoding='UTF-8'?>
<soapenv:Envelope xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope">
  <soapenv:Body>
    <ns1:addMemberLite xmlns:ns1="http://soap.ws.grouper.middleware.internet2.edu/xsd">
      <ns1:clientVersion>v1_4_000</ns1:clientVersion>
      <ns1:groupName>aStem:aGroup</ns1:groupName>
      <ns1:groupUuid></ns1:groupUuid>
      <ns1:subjectId></ns1:subjectId>
```

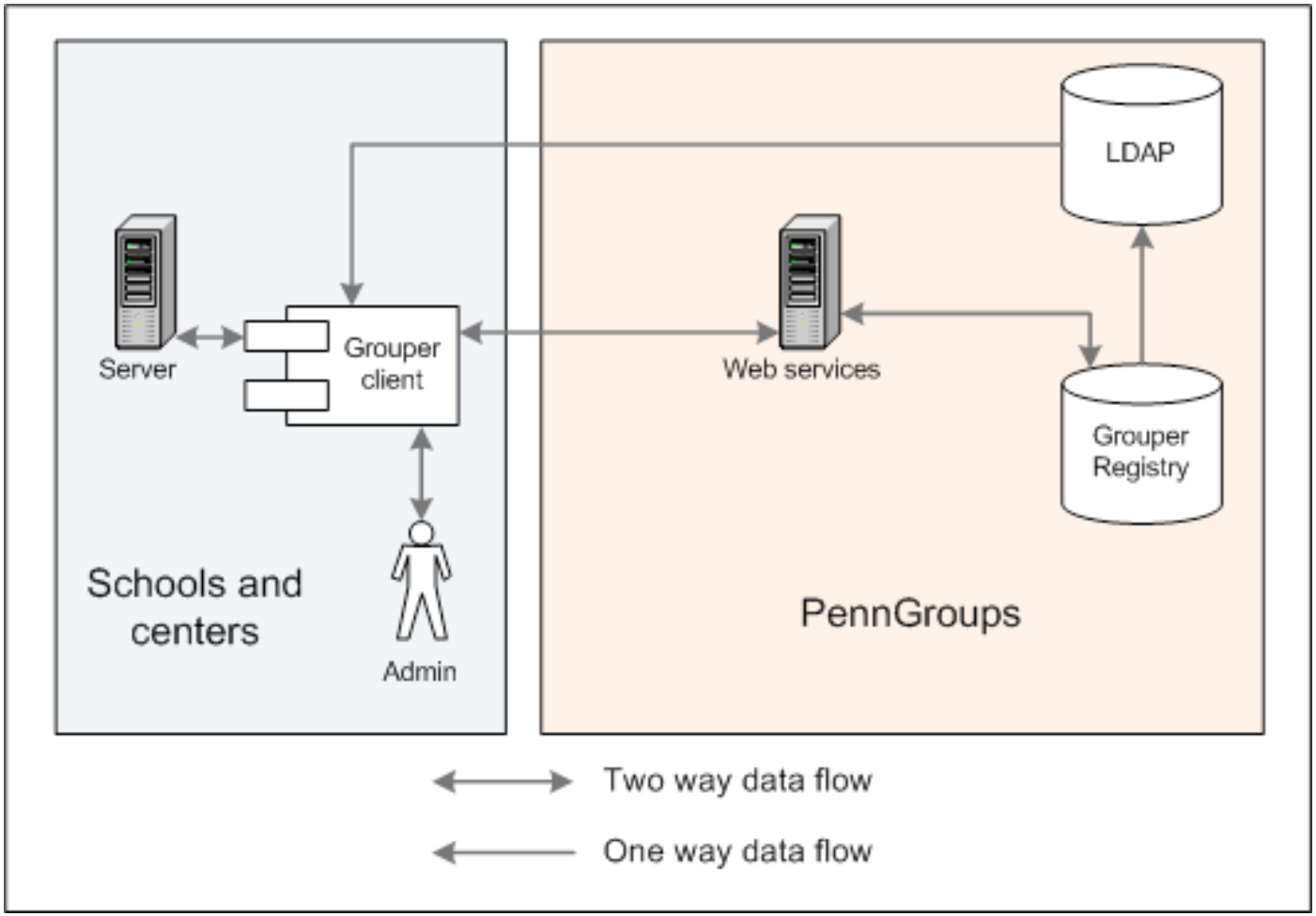
PennGroups LDAP

- There is a Grouper LDAP provisioning connector called LDAPPC, though Penn does not use this
- We have some simple triggers in Oracle which add records to a change log
- Then a process pulls records off of that table to sends diffs to openLDAP (runs every 10 minutes)
- Daily all records are refreshed
- Only users in penn:etc:ldapUsers can login to ldap
- Users can only read group membership lists they have privileges to read in Grouper

Grouper client

- LDAP and web services are low level
- Grouper client exposes Grouper LDAP and web services to a command line API or a Java library
- It can also be used to generate custom web service samples (can log requests and responses)
- Institutions can customize the client before distributing so the LDAP config is done (e.g. Penn allows ID lookups)
- Callers aren't tied to output, they can tell the client the output format that is expected

Grouper client (continued)



Grouper client (continued)

Sample LDAP config:

```
ldapSearchAttribute.operationName.2 = hasMemberLdap
ldapSearchAttribute.ldapName.2 = ou=groups
ldapSearchAttribute.matchingAttributes.2 = cn, hasMember
ldapSearchAttribute.matchingAttributeLabels.2 = groupName,
pennnameToCheck
ldapSearchAttribute.returningAttributes.2 = cn
ldapSearchAttribute.outputTemplate.2 = hasMember: ${resultBoolean}
ldapSearchAttribute.resultType.2 = BOOLEAN
```

- ## Sample LDAP command line call:

```
c:\grouper>  java -jar grouperClient.jar --operation=hasMemberLdap
--groupName=penn:myfolder:mygroup --pennnameToCheck=jsmith

hasMember: true

c:\grouper>
```

Grouper client (continued)

- Sample command line web service call:

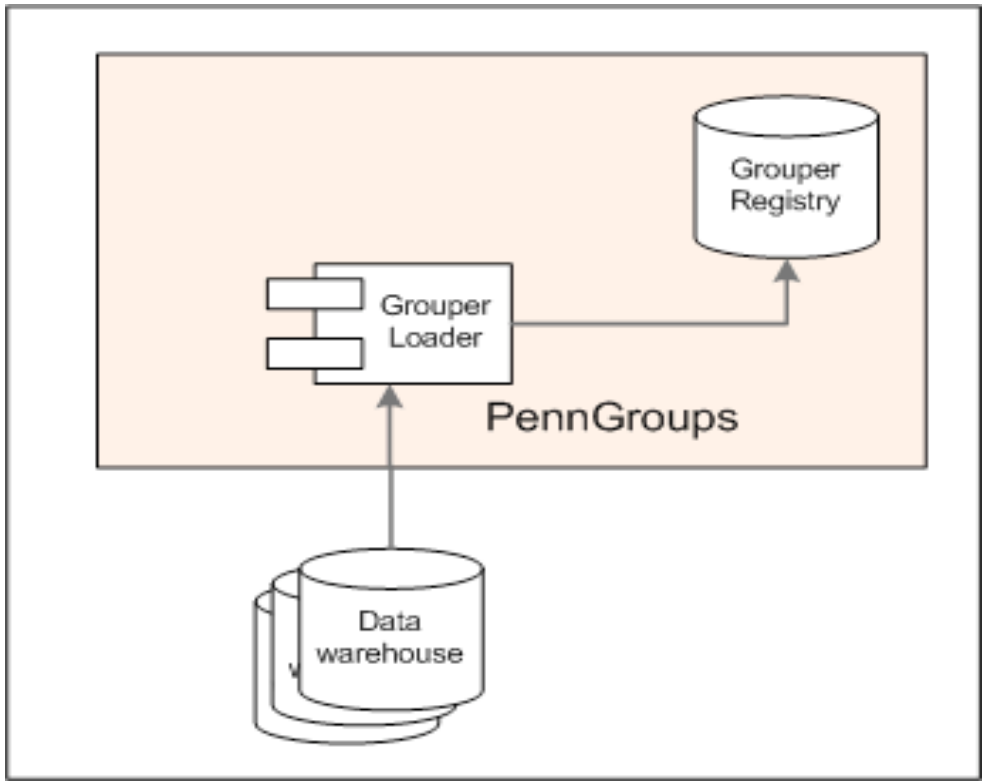
```
c:\grouper> java -jar grouperClient.jar --operation=getMembersWs
--groupNames=aStem:aGroup --outputTemplate=${index}: ${subject.id}

0: 12345
1: 23456

c:\grouper>
```

- Sample Java web service call:

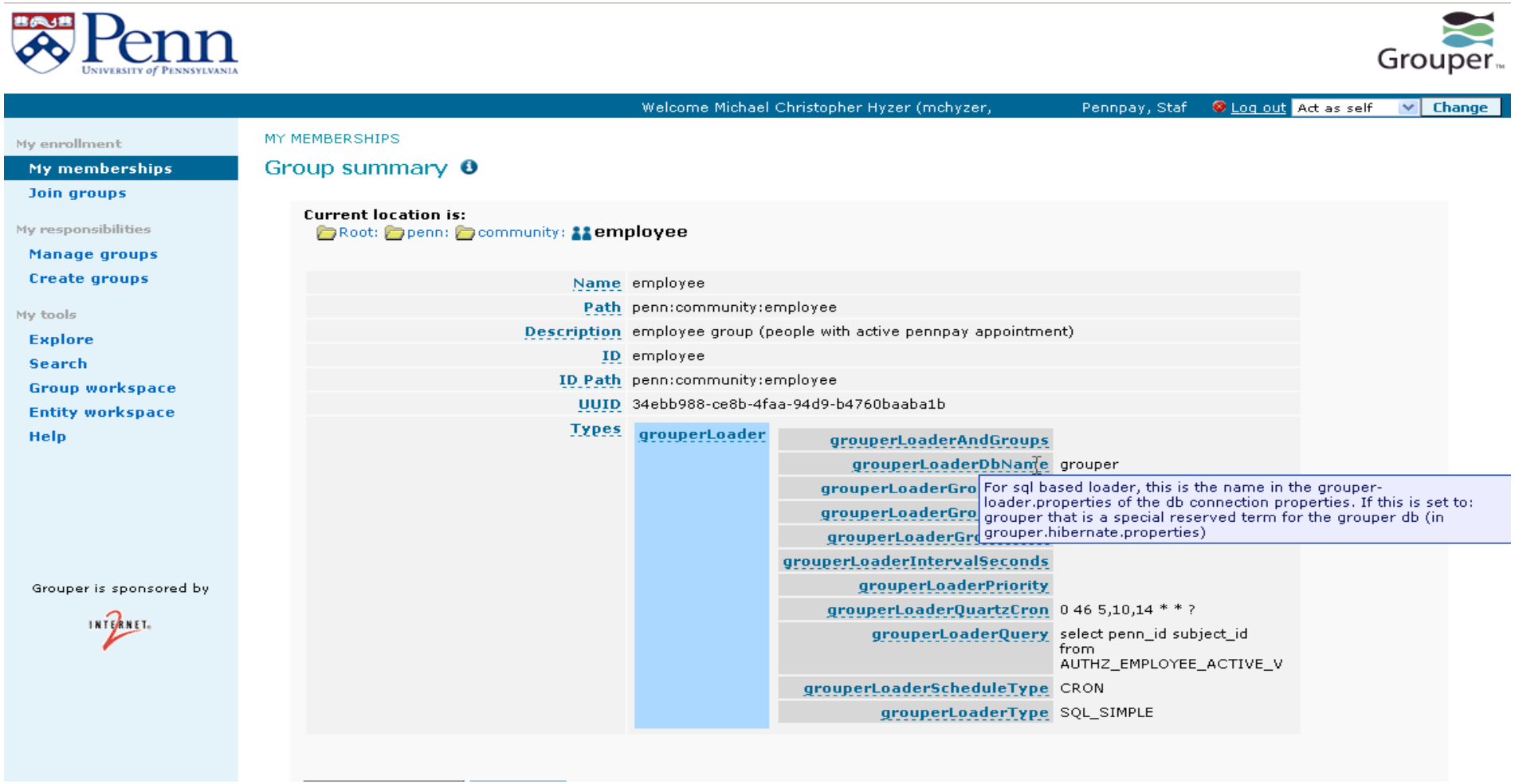
```
WsAddMemberResults wsAddMemberResults =
  new GcAddMember().assignGroupName("aStem:aGroup")
  .addSubjectId("12345").execute();
```

Grouper loader

- Penn contributed the “Grouper loader” in spring 2008
- This keeps groups in sync with results of sql queries



Grouper loader (continued)



Grouper loader (continued)

SQL> select * from authz_employee_active_v where rownum < 10

PENN_ID	PENN_NAME
12345	jsmith
12346	asmith
12347	bsmith
12348	rjohnson
12349	sjohnson
12350	tjohnson
12351	ajones
12352	bjones
12353	cjones

Grouper loader (continued)

Current location is:
Root: penn: community: employee: **orgGroups**

Name	orgGroups
Path	penn:community:employee:orgGroups
Description	dynamic group with configs for org groups (dont add members to this)
ID	orgGroups
ID Path	penn:community:employee:orgGroups
UUID	6f75f636-ad7c-4d50-a02f-c779b94c4aa8
Types	grouperLoader

grouperLoaderAndGroups	
grouperLoaderDbName	warehouse
grouperLoaderGroupQuery	
grouperLoaderGroupTypes	
grouperLoaderGroupsLike	
grouperLoaderIntervalSeconds	
grouperLoaderPriority	
grouperLoaderQuartzCron	0 8 9 * * ?
grouperLoaderQuery	select subject_id, group_name from EMPLOYEE_ORG_GROUPS_V
grouperLoaderScheduleType	CRON
grouperLoaderType	SQL_GROUP_LIST

Grouper loader (continued)

```
SQL> select * from employee_org_groups_v where rownum < 10
```

SUBJECT_ID	GROUP_NAME
12345	penn:community:employee:orgs:employeeOrg123
12346	penn:community:employee:orgs:employeeOrg123
12347	penn:community:employee:orgs:employeeOrg123
12348	penn:community:employee:orgs:employeeOrg124
12349	penn:community:employee:orgs:employeeOrg124
12350	penn:community:employee:orgs:employeeOrg124
12351	penn:community:employee:orgs:employeeOrg128
12352	penn:community:employee:orgs:employeeOrg128
12353	penn:community:employee:orgs:employeeOrg128

Grouper configuration checking

- If grouper is not configured correctly, it sometimes did not give descriptive errors
- With 1.4, on startup, it will verify its configuration and give descriptive errors
- It checks:
  - All DBs connectivity
  - Config file validity (including data types)
  - Subject API queries
  - System groups exist (auto-create)

Grouper configuration checking (continued)

- Print out useful grouper info on startup

```
Grouper starting up: version: 1.4.0, build date: 11/2/2008, env: DEV
grouper.properties read from: C:\grouper\build\grouper.properties
Grouper current directory is: C:\grouper
log4j.properties read from:   C:\grouper\build\log4j.properties
Grouper is logging to file:   console, at min level WARN for package:
   edu.internet2.middleware.grouper, based on log4j.properties
grouper.hibernate.properties: C:\grouper\grouper.hibernate.properties
grouper.hibernate.properties: jdbc:mysql://localhost:3306/grouper
sources.xml read from:        C:\grouper\build\sources.xml
sources.xml jdbc source id:   pennperson: GrouperJdbcConnectionProvider
sources.xml groupersource id: g:gsa
sources.xml jdbc source id:   jdbc: GrouperJdbcConnectionProvider
```

Grouper daily report

- With Grouper 1.4 there is a daily report
- This is emailed out every morning to grouper admins
- Includes a state of the registry:
  - E.g. number of new / total groups and memberships
- Loader job reports
  - Number of successes and failures, inserts/updates/deletes
- Registry health
  - Unresolvable subjects, bad memberships
- Stores history of reports on file system

Grouper daily report (continued)

```
Subject: Grouper report

OVERALL:

environment:            PROD

memberships:            135,280

groups:                 20

members:                56,207

folders:                17

unresolvable subjects: 1,197

bad memberships:        0

---------------

WITHIN LAST DAY:

new memberships:        66

new groups:             0

updated groups:         0

new folders:            0
```

Grouper binary distribution

- Grouper used to be distributed as source that needed to be built with ant and a java compiler
- Now with grouper 1.4 there is a binary build which is the java libraries
- All that is required is a java runtime
- An HSQL database is included, you can unzip, init the db, and run grouper shell (GSH)
- There is also a grouper client binary distribution

Grouper binary distribution (continued)

```
[mchyzer@ellis temp]$ tar xzf grouper.binary.1.4.0.tar.gz

[mchyzer@ellis bin]$ ./gsh.sh -registry -runscript

Grouper starting up: version: 1.4.0...

Are you sure you want to schemaexport db user 'sa', db url
   'jdbc:hsqldb:/temp/.../grouper;create=true'? (y|n):

y

Continuing...

Script was executed successfully

[mchyzer@ellis bin]$ ./gsh.sh

Grouper starting up: version: 1.4.0...

Type help() for instructions

gsh 1% addRootStem("myschool", "myschool");

stem: name='myschool' displayName='myschool' uuid='abcde'

gsh 2% addGroup("myschool", "agroup", "agroup");

group: name='myschool:agroup' displayName='myschool:agroup'
   uuid='abcdef'
```

Grouper encrypted passwords

- Grouper database passwords can now be encrypted and stored in external files to the normal config files
  - Grouper / loader DB's
  - Subject API
  - Grouper client LDAP and web service
- There is a stand-alone Internet2 library: morphString.jar (can easily be reused in other projects)
- Facilitates:
  - Non-cleartext passwords
  - Sanitized config files (for email or source control)
  - Separation between development and deployment

Grouper hooks

- Grouper 1.4 has 100 hook points built in to the data layer API
- You can get the data to do something (notification), add more queries to the transaction (audit), or veto the transaction
- Currently Grouper ships with some default implementations of hooks:
  - Group name and attribute validator regex (e.g. alphanumeric)
  - Group type edit security (e.g. only let admins edit grouper loader attributes)
  - Include/exclude auto-create
  - Require groups auto-create

More Information

- For technical documentation see the Internet2 Grouper wiki at:
  - Grouper product
    - https://wiki.internet2.edu/confluence/display/GrouperWG/Grouper+Project
  - Grouper project
    - https://wiki.internet2.edu/confluence/display/GrouperWG/Grouper+Project
  - Web services info
    - https://wiki.internet2.edu/confluence/display/GrouperWG/Grouper+-+Web+Services

Grouper DDL management

- Grouper used to use Hibernate schemaexport
- Switched to a custom method built on Jakarta ddlutils
- Supports hsql, oracle, mysql, and postgres (and probably other untested db's)
- Supports tables, views, comments, indices, foreign keys, data massaging
- Knows when the database is out of sync (keeps state in DB table), and logs to ERROR that update needed
- If you drop a column of a table, and run "deep" ddl registry check, it will generate DDL to recreate it

Grouper DDL management (continued)

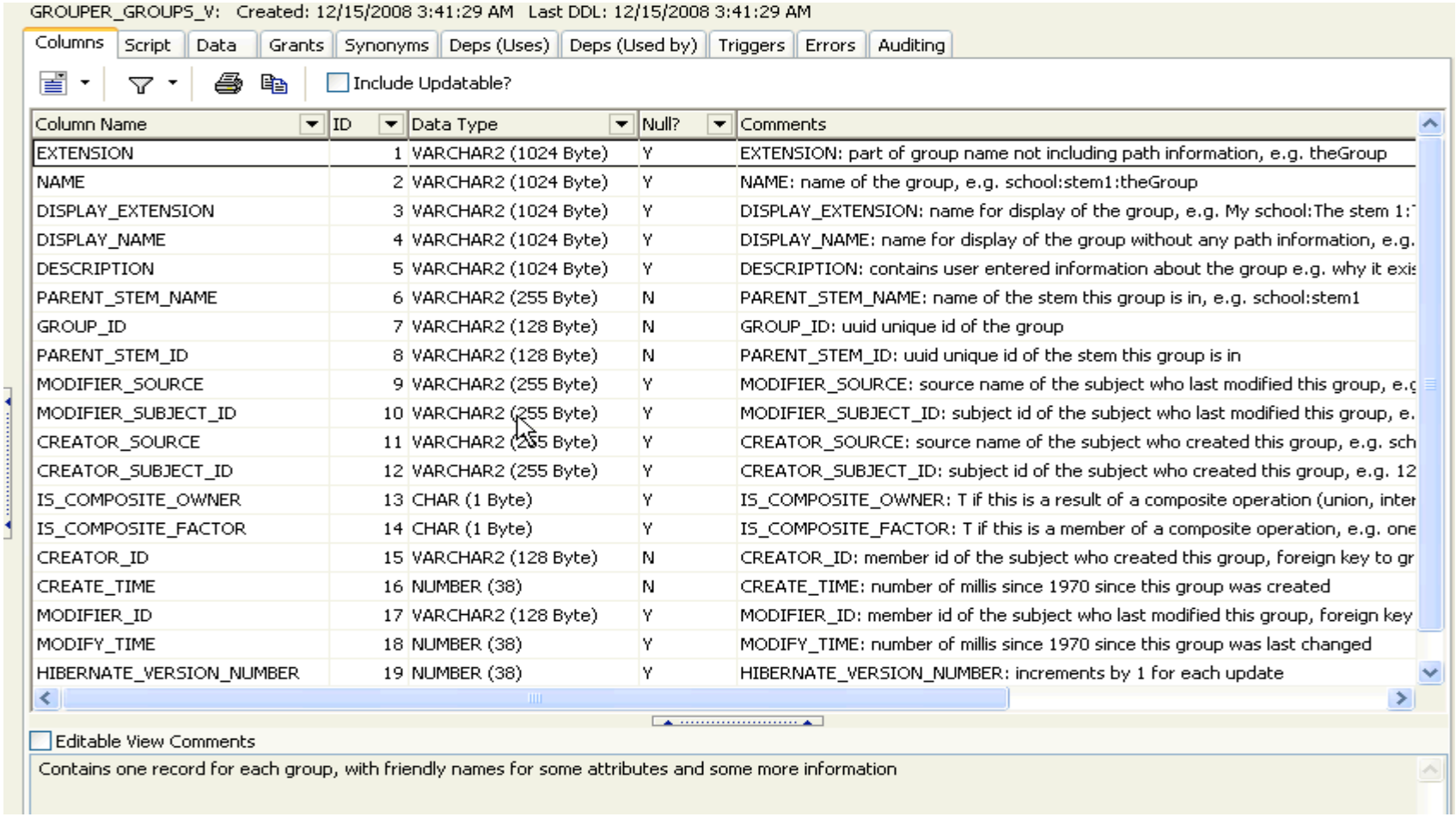
GROUPER_GROUPS: Created: 12/15/2008 3:41:18 AM Last DDL: 12/15/2008 3:41:22 AM

Columns | Indexes | Constraints | Triggers | Data | Scripts | Grants | Synonyms | Partitions | Subpartitions | Stats/Size | Referential | U

Sort by Primary Key
Read Only

ID	PARENT_STEM	CREATOR_ID
711a6173-2b45-4c5d-a814-e45669c13793	42891ef7-74e6-4cbd-987e-451620b25988	540f6f10-784c-4396-a0d
644e26b0-0295-4b54-b7e4-6e09b1e7716e	42891ef7-74e6-4cbd-987e-451620b25988	540f6f10-784c-4396-a0d
54e35f33-ab76-4894-9372-f807c1972a29	42891ef7-74e6-4cbd-987e-451620b25988	540f6f10-784c-4396-a0d
937a8933-e547-4039-8033-cb90eeb3cc2d	42891ef7-74e6-4cbd-987e-451620b25988	540f6f10-784c-4396-a0d
7fe9678c-e92b-48a8-91ab-82d0222ad38e	42891ef7-74e6-4cbd-987e-451620b25988	540f6f10-784c-4396-a0d
7f2f0d0f-eeb4-4[illegible]ba-a44e-c5503a6b5263	42891ef7-74e6-4cbd-987e-451620b25988	540f6f10-784c-4396-a0d
4813a3b3-7647-4956-a81d-8fa03cdf5a3d	42891ef7-74e6-4cbd-987e-451620b25988	540f6f10-784c-4396-a0d
53fe910a-057e-4291-9384-44a450a238ce	42891ef7-74e6-4cbd-987e-451620b25988	540f6f10-784c-4396-a0d
6a0954ac-feb1-4bb0-bfe9-eefdbeb09947	42891ef7-74e6-4cbd-987e-451620b25988	540f6f10-784c-4396-a0d
740f2098-0215-4756-9f8f-4f2563d7f20f	42891ef7-74e6-4cbd-987e-451620b25988	540f6f10-784c-4396-a0d
04c22387-e5a1-4858-9765-d7edefc256cf	42891ef7-74e6-4cbd-987e-451620b25988	540f6f10-784c-4396-a0d
ada0dc1f-7a3c-4e9c-a1eb-bf1780002d6b	42891ef7-74e6-4cbd-987e-451620b25988	540f6f10-784c-4396-a0d
913f96cb-0ceb-4bce-a1a6-15428810cc66	42891ef7-74e6-4cbd-987e-451620b25988	540f6f10-784c-4396-a0d
e81d3421-0f5d-4824-b229-16bff6113990	42891ef7-74e6-4cbd-987e-451620b25988	540f6f10-784c-4396-a0d
093b398e-2cd4-46b0-89e5-0e78062d238c	42891ef7-74e6-4cbd-987e-451620b25988	540f6f10-784c-4396-a0d

Grouper DDL management (continued)



GROUPER_GROUPS_V: Created: 12/15/2008 3:41:29 AM Last DDL: 12/15/2008 3:41:29 AM

Columns | Script | Data | Grants | Synonyms | Deps (Uses) | Deps (Used by) | Triggers | Errors | Auditing

Include Updatable?

Column Name	ID	Data Type	Null?	Comments
EXTENSION	1	VARCHAR2 (1024 Byte)	Y	EXTENSION: part of group name not including path information, e.g. theGroup
NAME	2	VARCHAR2 (1024 Byte)	Y	NAME: name of the group, e.g. school:stem1:theGroup
DISPLAY_EXTENSION	3	VARCHAR2 (1024 Byte)	Y	DISPLAY_EXTENSION: name for display of the group, e.g. My school:The stem 1:
DISPLAY_NAME	4	VARCHAR2 (1024 Byte)	Y	DISPLAY_NAME: name for display of the group without any path information, e.g.
DESCRIPTION	5	VARCHAR2 (1024 Byte)	Y	DESCRIPTION: contains user entered information about the group e.g. why it exis
PARENT_STEM_NAME	6	VARCHAR2 (255 Byte)	N	PARENT_STEM_NAME: name of the stem this group is in, e.g. school:stem1
GROUP_ID	7	VARCHAR2 (128 Byte)	N	GROUP_ID: uuid unique id of the group
PARENT_STEM_ID	8	VARCHAR2 (128 Byte)	N	PARENT_STEM_ID: uuid unique id of the stem this group is in
MODIFIER_SOURCE	9	VARCHAR2 (255 Byte)	Y	MODIFIER_SOURCE: source name of the subject who last modified this group, e.g
MODIFIER_SUBJECT_ID	10	VARCHAR2 (255 Byte)	Y	MODIFIER_SUBJECT_ID: subject id of the subject who last modified this group, e.
CREATOR_SOURCE	11	VARCHAR2 (255 Byte)	Y	CREATOR_SOURCE: source name of the subject who created this group, e.g. sch
CREATOR_SUBJECT_ID	12	VARCHAR2 (255 Byte)	Y	CREATOR_SUBJECT_ID: subject id of the subject who created this group, e.g. 12
IS_COMPOSITE_OWNER	13	CHAR (1 Byte)	Y	IS_COMPOSITE_OWNER: T if this is a result of a composite operation (union, inter
IS_COMPOSITE_FACTOR	14	CHAR (1 Byte)	Y	IS_COMPOSITE_FACTOR: T if this is a member of a composite operation, e.g. one
CREATOR_ID	15	VARCHAR2 (128 Byte)	N	CREATOR_ID: member id of the subject who created this group, foreign key to gr
CREATE_TIME	16	NUMBER (38)	N	CREATE_TIME: number of millis since 1970 since this group was created
MODIFIER_ID	17	VARCHAR2 (128 Byte)	Y	MODIFIER_ID: member id of the subject who last modified this group, foreign key
MODIFY_TIME	18	NUMBER (38)	Y	MODIFY_TIME: number of millis since 1970 since this group was last changed
HIBERNATE_VERSION_NUMBER	19	NUMBER (38)	Y	HIBERNATE_VERSION_NUMBER: increments by 1 for each update

Editable View Comments

Contains one record for each group, with friendly names for some attributes and some more information

Grouper DDL management (continued)

- For the upgrade to Grouper 1.4, we removed some duplicate UUID's and normalized some tables
- Backups for columns are kept
- Columns are dropped
- SQL to update other cols
- All generated in a DB independent way
- Though can also grouper-export and import in new registry

Grouper DDL management (continued)

- Some versions of mysql cannot accept indices on cols longer than 1000 bytes
- Grouper can accommodate this (even though Jakarta ddlutils cannot)

```
// see if we have a custom script here, do this since some versions of mysql
// cant handle indexes on columns that large
String scriptOverride = ddlVersionBean.isMysql() ?
    "\nCREATE INDEX attribute_value_idx " +
    "ON grouper_attributes (value(333));\n" : null;

GrouperDdlUtils.ddlutilsFindOrCreateIndex(database, ddlVersionBean,
    attributeTable.getName(), "attribute_value_idx", scriptOverride, false,
    "value");
```