

CESNET Technical Report 5/2010
Grouper in University Environment,
Implementation at the University of West
Bohemia

JIŘÍ BOŘÍK, FRANTIŠEK DVOŘÁK, JOSEF KRUPIČKA

University of West Bohemia in Pilsen

Received 19.12.2009

Abstract

In this project the Grouper system is utilised as an application provider data source. However, the Grouper system could be optimized for this purpose in university user community. As a part of the final solution this project handles interconnecting between Grouper system and Sun Java System Identity Manager.

Keywords: grouper, Internet2, federation, groups, Identity Management, Sun IdM

1 Identity Management in University Environment

Identity management in university environment is usually quite complicated. University computing environment is used to be heterogeneous and it consists from big amount of end systems. Beyond that there is a high fluctuation, because the most of identities origin from students.

It is necessary to have central system in the advanced computing environment, which provides propagation of identities. Most often there are available one or more authoritative data sources with information, which need to be transferred to end systems. The transfer provides the central identity management, and keeps the data in the end systems up to date.

How grow application demands in computing environment, it is necessary more detailed control of access rights and more detailed distinguish in roles. Integral part of the management is then group and membership management.

2 Searching Solution for Group Management

There are several ways, how to deal with groups management:

- home-made solutions – updating scripts, simple tools, ...
- part of central Identity Management – utilization of the possible features provided by Identity Management; not always full and high-quality support for groups
 - MIT Moira – older system; utilization of email list management
 - Sun Identity Management – without central group management, but there is basic support in individual resource adapters

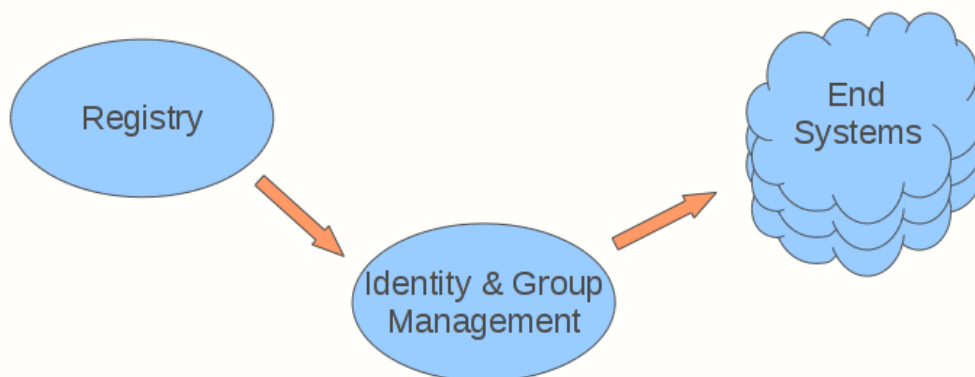


Figure 1. Simplified Insight on Central Identity Management

- standalone specialized SW

We had following requirements, when searching the right solution for group management:

- it is required some user interface for group management
- possibility to delegate the rights to manage the groups – for example support for user groups too
- usability of group arithmetic – defining groups as combinations of other objects (unions, products, ...)
- fast and efficient data import and export

Our final decision was a project from Internet2 consortium – Grouper.

2.1 Grouper

Grouper is a group management toolkit developed by the Internet2 Middleware Initiative.

By consolidating group information in Grouper and creating a single point of management, changes of membership are done once and then provisioned to the applications. Grouper holds information not only about the membership, but also the hierarchy of authority regarding who may create, update, and delete it. It enables the group owner to define membership, create a group structure within their domain, or delegate all of this to someone else.

With Grouper, individuals within a campus manage the memberships of the groups they steward. Grouper keeps the group membership decisions in the hands of the business/group owners, access control in the hands of the application owners, and the technology management in the hands of the technologists.

Grouper separates the management of the groups from the technical system, so a change in technology details does not affect those using it. After integrating Grouper with identity management system, there will be a way to manage the

membership of roles and other functions that individuals have with the institution. Further, automatic change or revocation of service can be accomplished based on group membership changes. Removing IT from the middle of managing groups will help ease your helpdesk headaches as well.

So Grouper fulfills our requirements. In addition it is used in universities across Europe and USA.

We bumped to several drawbacks too, which was needed to deal with:

- relatively slow and complicated user interface
- missing Czech localization
- no incremental imports and export
- needless repeated accesses to LDAP directory
- very long data import and export times

There was needed to create a bind too between Grouper and Sun Identity Manager.

3 Description Implementation of Grouper in Common Environment

3.1 Generic design

At the University of West Bohemia, we keep dynamic groups up-to-date by means of our own import tools, synchronizing data in Grouper directly from source agenda.

The following figure illustrates a generic way of the connections of the Grouper on the surrounding systems.

Our goal on the Sun IdM side was achieving unidirectional transfer, from Grouper to Sun IdM, with Sun IdM taking care of all various target resources at the university. The resulting relationships are:

- 1) *grouper “funnel”* – checking out from source systems and importing to Grouper
- 2) *grouper -> IdM* – read-only Grouper interface accessible by Identity Management
- 3) *IdM -> end systems*

3.2 Grouper Funnel

There are several information systems from which we need to extract members of these groups:

- teachers, staff, students,...
- courses students and teachers
- lessons students and teachers
- departments students, teachers, ...
- ...

Standard GrouperLoader (the importer supplied with Grouper SW) is not fit for our purpose because some systems (for example AFS groups) do not offer a database interface. Besides that, we want a fast tool capable of returning groups

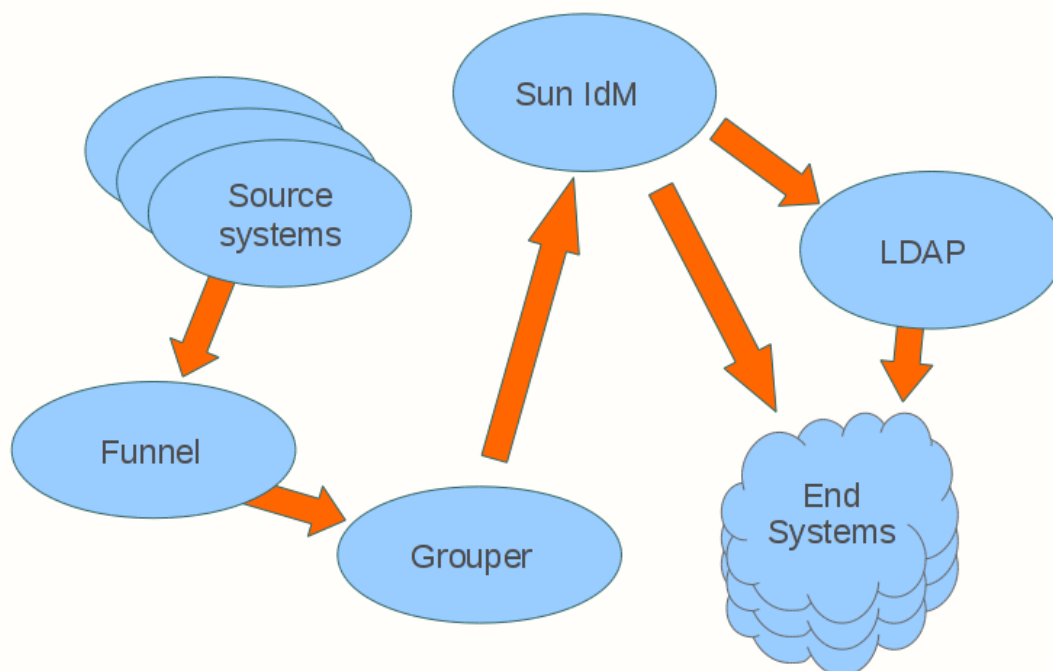


Figure 2. Design of the Connections of the Grouper on the Surrounding Systems

and their members in a matter of seconds, not causing any significant load to source information systems.

That is why we have developed a simple application, which provides a generic database interface and another interface for developing custom classes.

You can download `GrouperFunnel.zip`¹, containing:

- sample config file
- binary distribution with required libraries
- Java sources
- shell script

Every source is fetched in a separate thread. Results are stored in simple csv files. We plan to change it to more verbose XML format with more options.

Currently, we have a configuration file with twelve sources. With optimized sql and custom classes, fetching all data takes several seconds. Our custom provisioning allows us to accomplish near-realtime distribution of changes in central systems to Grouper and consequently to Sun IdM.

In the second stage of the *funnel*, recently gathered data are fed to Grouper. We have implemented a custom importer to perform this function; most of its logic is described in the following chapter.

¹ <https://spaces.internet2.edu/download/attachments/10519561/GrouperFunnel.zip?version=1>

3.2.1 Importer

We wanted a fast way to import large number of groups and group memberships. GSH (a scripting interface application for Grouper) is very slow. Using plain Grouper API with some logic for incremental changes is better, but still not good enough. So we developed simple application which makes changes in Grouper database with plain JDBC and uses batch updates.

Our importer reads input files. The name of input file corresponds with the name of a *stem* (=hierarchical namespace for groups, colon separated path) to which groups and theirs members will be imported. If the *stem* doesn't exist, application will create it.

At start it will fetch all subjects (user identities) from registered sources, so it doesn't need to query for every imported subject every time. It is incremental importer, so it fetches current status of groups and their members in *stem*, and compares it with data from input file. This "diff" runs in several threads.

Source files:

- *grouper-importer.zip*² – contains binary and source code distribution of our custom provisioning tool
- *trg_compute_insert_membership.sql*³ – trigger to compute effective and composite memberships after membership insert
- *trg_compute_delete_membership.sql*⁴ – trigger to compute effective and composite memberships after membership delete
- *grouper_memberships_seq.sql*⁵ – sequence which is used to generate effective and composite memberships id

3.2.2 Simple Benchmark

We did simple benchmark where we compared our custom provisioning with provisioning based on Grouper API. It was tested on Intel Xeon 5160 (4 cores, 3 GHz) with 8 GB RAM and two SAS disks in raid.

- *OS*: customized Debian based on Lenny
- *RDBMS*: Postgres 8.3

Provisioning application was started on the same server.

Testing data contained 200 groups with 183000 group membership. Size of the groups ranged from 1 to 20000.

Results:

- *console application (import from XML file)*: 10 hours
- *script interface GSH*: 6 hours

² <https://spaces.internet2.edu/download/attachments/10519622/grouper-importer.zip?version=3>

³ https://spaces.internet2.edu/download/attachments/10519622/trg_compute_insert_membership.sql?version=1

⁴ https://spaces.internet2.edu/download/attachments/10519622/trg_compute_delete_membership.sql?version=1

⁵ https://spaces.internet2.edu/download/attachments/10519622/grouper_memberships_seq.sql?version=1

- *grouper API based multithread application*: 1 hour
- *database triggers based multithread application*: 759 seconds

So initial import this data using our new solution took 759 seconds. Subsequent incremental update durations depends on the amount of the data. In the wild it is matter of seconds.

3.3 Grouper Interface

We have decided to allow Sun IdM access Grouper's database, with specific views of the database used by IdM as an interface.

We need some way of logging changes in groups and their membership. It is important for the integration of our Sun IdM installation with Grouper. Grouper 1.4 does not offer this feature, which is why we have developed a set of triggers and views providing this functionality. Hooks may do the job, but using Grouper Java API is quite slow.

We are using Oracle and Postgres, but the final solution is based on Postgres. There is no fancy logic so it should be possible to convert these triggers and use them in other RDBMS (Oracle, MySQL, ...).

Information concerning changes in groups and group membership is stored in table `grouper_changes` with the following columns:

- *operation* – specify what kind of operation was made: insert, delete
- *item_type* – specify type of entity: group or membership
- *item_id* – entity id: `group_id` or `member_id`
- *item_name* – group or member name
- *parent_item_id* – parent group or stem id
- *parent_item_name* – parent group or stem name
- *timestamp*

SQL scripts:

- `grouper_changes.sql`⁶ – sql script for table `grouper_changes`
- `record_attribute_changes.sql`⁷ – function which monitors changes in table `grouper_attributes` and logs info about changes in groups
- `record_membership_changes.sql`⁸ – function which monitors changes in table `grouper_memberships` and logs info about changes in memberships
- `triggers.sql`⁹ – sql script for triggers

There are two views for the Grouper IdM resource adapter:

- `grouper_idm_groups.sql`¹⁰ – list of groups

⁶ https://spaces.internet2.edu/download/attachments/10519577/grouper_changes.sql?version=1

⁷ https://spaces.internet2.edu/download/attachments/10519577/record_attribute_changes.sql?version=1

⁸ https://spaces.internet2.edu/download/attachments/10519577/record_membership_changes.sql?version=1

⁹ <https://spaces.internet2.edu/download/attachments/10519577/triggers.sql?version=1>

¹⁰ https://spaces.internet2.edu/download/attachments/10519577/grouper_idm_groups.sql?version=1

- grouper_idm_memberships.sql¹¹ – list of members

There are only groups with assigned 'IDM' access right in those views. Grouper could be used this way for different systems management independently on identity management.

4 Implementation at University of West Bohemia

4.1 Environment at University of West Bohemia

The distributed computing environment at the university has central administration. Sun Java System Identity Manager (Sun IdM) is deployed as an identity manager. Groups and group membership are managed by Grouper from Internet2 consortium. Passwords are stored in Kerberos, authorization to web applications (WebAuth, SSO) relies on LDAP groups or static lists.

Systems using group management at University of West Bohemia are:

- LDAP “rfc2307” – unix accounts and groups, used for authorization, too
- LDAP “portal” – identities for portal, fewer of groups, different schema
- AFS – pts groups and users, updated only several selected groups
- Windows Active Directory – domain control for Windows world

4.1.1 LDAP Directory

Served by two duplicated and load-balanced OpenLDAP servers. There are two branches – “rfc2307” and “portal”. There are identities with reverse information about membership, too, in both (group list for each account).

In addition we have two group types in the main LDAP directory branch:

- *posixGroup* – with gid number
- *pleiadesGroup* – without gid number

Sun IdM resource adapter was extended to support our additional *pleiadesGroup* schema. Custom workflow can change the schema when needed by recreating the group.

4.1.2 Java System Sun Identity Manager (Sun IdM)

Certain options and conventions in Sun IdM need to be introduced first.

There are several synchronization methods in Sun IdM. The two most important are Reconciliation and Active Sync.

Reconciliation is used for elementary comparison of data in identity management and the target resource. Synchronization goes through all identities and it is typically used just to look up missing identities and perform basic operations (responses, e.g. create, delete, unlink...). Checking for attribute changes in target systems is possible, but quite resource-intensive.

¹¹ https://spaces.internet2.edu/download/attachments/10519577/grouper_idm_memberships.sql?version=1

Active Sync is important for on-line updates. It is intended to update only identities that have been modified in the source resource, making it a preferred method of transferring modified attributes. This method requires support at the source – ability to select all identities that have changed since the last synchronization.

IdM offers elementary support for managing groups on several types of the target resources. For example, group members in directory based (LDAP, Active Directory) and unix-like resources are manageable through multivalued attributes assigned to user accounts. Group membership at that resource is managed by updating this attribute.

Groups themselves as well as their metadata are represented as a special type of resource objects on target resource adapters in Sun IdM. Automatic transfer of groups between resources is not supported directly in Sun IdM by standard synchronization mechanisms, but it may be done by custom workflows. More simplistic approach could treat groups in the same way as user identities, using two separate resources – one for users and another one for groups. It would require resource types, where groups are represented by identities, like in case of users (for example *LDAP* is good, but *passwd* resource couldn't be used). We have chosen the more difficult and more interesting way – using custom workflows and treating groups as special objects (see Sun IdM Deployment chapter).

4.2 Synchronizing Groups between Grouper and Sun IdM

We use Active Sync as the preferred method of synchronizing groups and group membership in Sun IdM. Incremental changes should be quick, and we may use complicated custom workflows when necessary.

As far as groups removal is concerned, there are few quite complicated scenarios. Managing users AND groups may lead to race conditions. Sun IdM requires us to avoid assigning users to groups that do not yet exist on target resources. The solution we chose involved an adjustment of the order of updates:

- 1) process all changes in existing groups and newly created groups
- 2) process all users with changes in memberships
- 3) process all groups marked for deletion

Database views have been created in Grouper's database to provide an interface accessible by Sun IdM. One view shows groups and another one shows group membership. Both views are being polled periodically by Active Sync, relying on timestamps in the views to recognize recently modified identities.

The Sun IdM side uses a single source resource adapter to retrieve both users and groups. Used attributes:

users

- *accountId*: login name
- *objectClass*: “user”
- *grpgroups*: membership group names (multivalued)
- *grpenabled*: membership group states (multivalued)

- *grpchanged*: time of the membership group changes (multivalued)

groups

- *accountId*: group IdM identity internal identifier ('group.'STEM:GROUP_NAME)
- *objectClass*: "group"
- *grpgroups*: group name
- *grpenabled*: group state
- *grpchanged*: time of the group metadata change
- *grpuid*: group unix id number
- *grpstem*: group stem (hierarchy) in Grouper

As seen from the lists: groups being removed are not deleted from the views, just marked as disabled, setting the proper timestamp in *grpchanged*. This way Sun IdM can detect group or membership removal.

4.3 Sun IdM Deployment

Different groups should be propagated to the different systems. In simpler cases it's possible to classify groups to stems and use regular expressions to control which group should be propagated to which target system.

There are two regular expressions per target system to specify desired groups: selective (hits) and suppressive (exceptions). See also *ZCU Rules:Group Regex*.

4.3.1 Configuration Files

- *grouper.xml*¹² – configuration of the resource adapter
- *adapter-actions.zip*¹³ – Grouper resource actions in Java (using JDBC API)
- *zcu-sample-workflow.zip*¹⁴ – custom workflow of group synchronization, helper functions and other objects...

5 Summary

The fundamental concept is:

- *Grouper* – "funnel" – regular checkout from source systems such as STAG (study agenda) or CRO (person registry), list of logins is in the LDAP directory
- *Sun IdM* – updating end systems (LDAPs, Active Directory, AFS, ...)

¹² <https://spaces.internet2.edu/download/attachments/10519572/grouper.xml?version=1>

¹³ <https://spaces.internet2.edu/download/attachments/10519572/adapter-actions.zip?version=1>

¹⁴ <https://spaces.internet2.edu/download/attachments/10519572/zcu-sample-workflow.zip?version=1>

5.1 Original Grouper

We have already been using Grouper at the university before this project. Loading data was very inefficient, taking hours to complete (“study”, “affiliation” binds in the figure). End systems were being updated by complicated scripts launched by cron (the red full lines binds in the figure), and export to other systems was not implemented (the dotted lines binds in the figure).

Important relationships shown in Figure 3:

- *import from IS/STAG, CRO (affiliation, study), LDAP (users)* – conceptually OK, but very demanding and slow updates
- *export to LDAP, LDAP portal* – need to be removed and replaced by Sun IdM, Sun IdM is already connected to these to update user accounts
- *export to AD, AFS* – planned, but never implemented or not finished

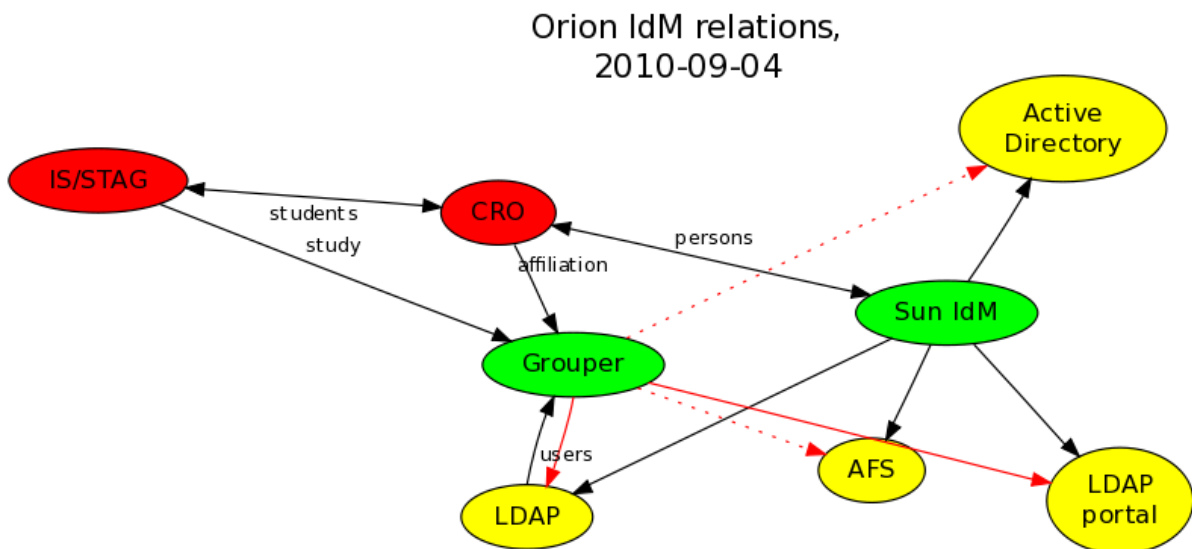


Figure 3. Relationship of All Systems before New Grouper

5.2 New Optimized Grouper

This project resulted in more efficient, faster updates. Grouper *funnel* was drastically sped up, with data transfer taking less than a minute. Information goes from Grouper through a single channel to Identity Management (see single red line in the figure), and Identity Manager keeps end systems up-to-date in a unified way (see double-line red-black bindings in the figure). It is easier to add new end systems with groups (plans to extend exports to AFS and Active Directory).

Important relationships in the new environment:

- *import from IS/STAG, CRO (affiliation, study), LDAP (users)* – optimized Grouper “funnel”
- *export to LDAP, LDAP portal* – relationships existing previously, groups and memberships are updated through those

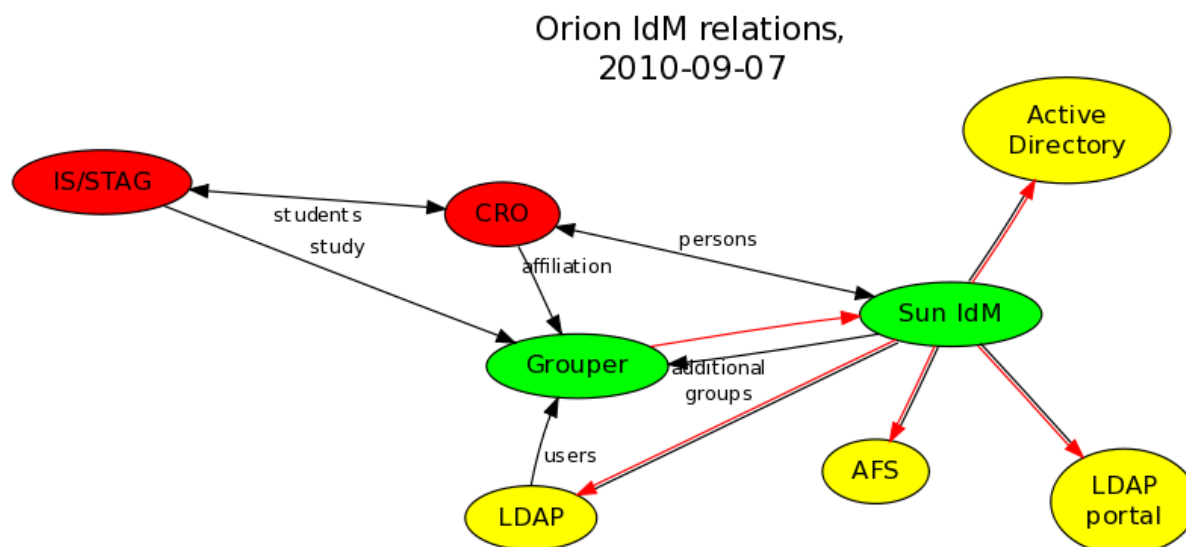


Figure 4. Relationship of All Systems with New Grouper

— *export to AD, AFS* – both implemented within Sun IdM

For groups and membership propagation is taken the advantage of already existing binds on end systems.

On the figure above is presented that groups updates has been unified and simplified. The real environment is a little more complicated though:

5.3 Practical Problems

5.3.1 Initial Set-Up

There were certain practical problems during initial synchronization. Joining all identities with Grouper inside Sun IdM took some time, and identities that have not been joined inside IdM had no groups assigned and could not have been propagated to target systems.

Other updates that were being carried out by Sun IdM at the same time were deleting group membership information systematically. We identified the problem quite early and solved it without having to interrupt the time consuming initial synchronization.

5.3.2 Big Transaction Logs

There are several large groups (in case of the University, they are staff, students and users). In OpenLDAP, changing one membership in a group results in replacing the whole multivalued attribute with the new one, which is quite a big transaction. Sun IdM is configured to change membership records by updating identities one-by-one, which lead to many changes and to a very big transaction log on the LDAP server, filling up the whole disk.

Problem has been solved one-shot by manual change of these big groups. If the Sun IdM find the member in the given groups (or do not find when deleting

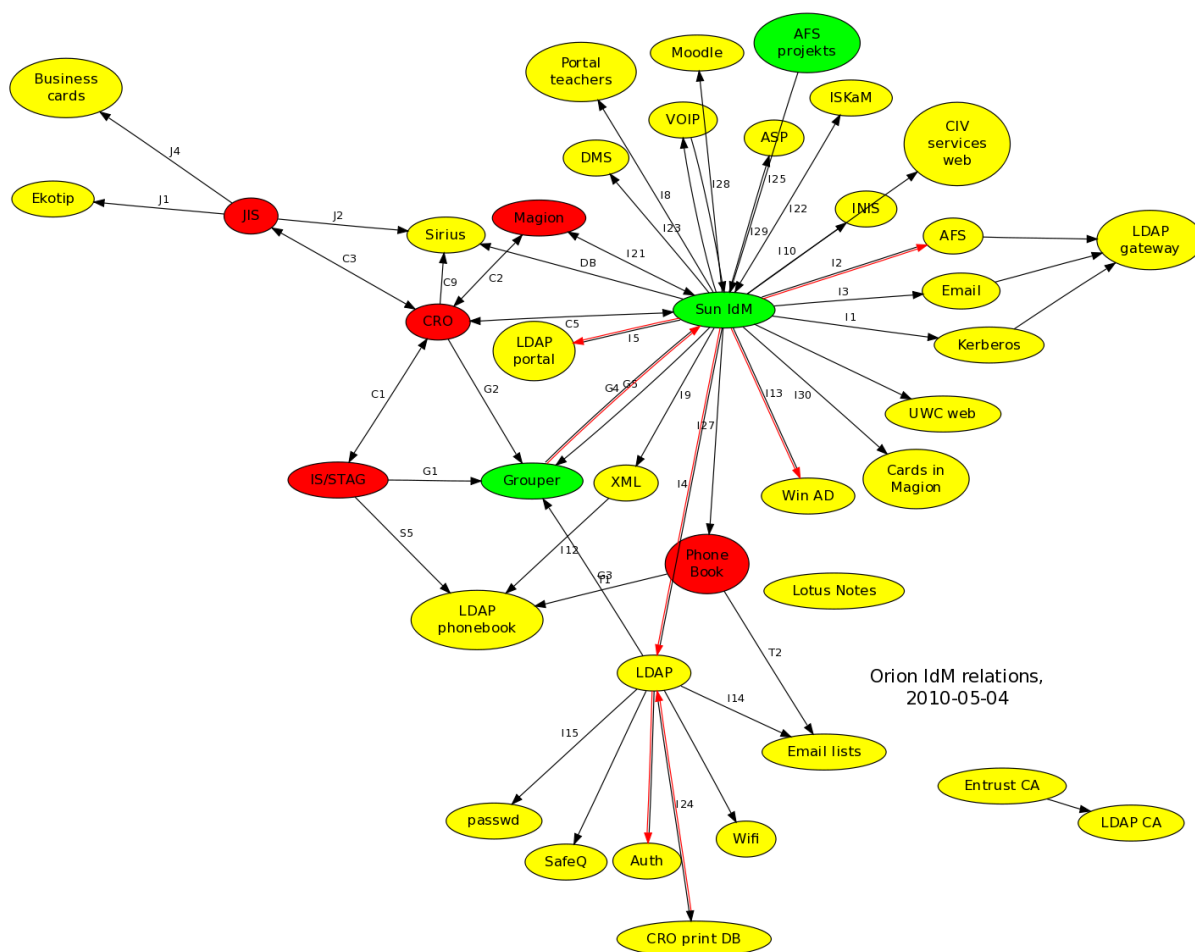


Figure 5. Relationship of All Systems at UWB

membership), the big group record will not be updated anymore.

Further updates are small, so improvements have not been needed anymore. There is a way how to fix it, though. It is possible to keep members in groups (memberUid and uniqueMember attributes) independently from group lists in identities (memberof attribute). The ActiveSync synchronization process could be modified to update big multivalued member attributes in groups just once, in advance, before every update cycle.

6 Conclusion

Practical result of this project is optimized Grouper in production environment at University of West Bohemia. Importing of groups and group membership decreased from hours to seconds, which is more than sufficient for us.

Synchronization is handled by central Identity Manager. Modifying end systems is done by unified way by Sun IdM, compared to using additional scripts for groups and group membership before. We've developed new resource adapter for Sun IdM to gather all necessary data from Grouper on-line. In addition to accounts and group membership, it is automatized group creating, removing, and modifying their metainfo too.

All this is done with awareness of Grouper developers, as see at Internet2 contributor page¹⁵. There is possibility to include the changes in future Grouper releases.

Grouper WWW interface was localized to Czech. New simple user interface for managing groups has been developed too, since casual user does not need interface so complex as the standard WWW grouper interface.

¹⁵ <https://spaces.internet2.edu/display/GrouperWG/University+of+West+Bohemia>