The following contains some example texts from the URZadmin documentation. The chapters were chosen randomly.

Roles

DB tables	Description
fct	List of known roles (functions), like Rektor, Vorsteher or CV
mfct	Assigns roles to persons

The concept of a role, as it is used here, is that of a function a person takes in the organization. It should be distinguished from a "technical role" like the administrator within URZadmin or the moderator of a mailing list which are regarded, at least in our context, as access rights to an application, i.e. authorizations. Of course, a role may imply access rights to some applications, but the two concepts should be clearly separated.

An important function of a role is the display in perssearch: It is a requirement to indicate that a person is the rector of the university, head of a department, etc. To make this display correctly, we distinguish between the male and female (and neutral) forms of the role's name.

Every role has a priority assigned to it. It is represented by a number with lower numerical values meaning a higher "organizational importance". When displaying multiple roles of a person, the list will be ordered "high importance first", i.e. by increasing numerical values of the priority. In perssearch, the publicly accessible person search of the university, roles are displayed along with person's entries when the role priority is not above of 200.

Often, a person holds a role for a certain organization. A head of department is head of a well defined department: he is the head of department of mathematics and not the head of department of economics. On the other hand, the role of an assistant may not be bound to an organization, even though an assistant normally works for a well defined institute. With every role, it can be specified whether it must also be mapped to an organization in addition the role holder.

In order to prevent having several rectors for the university or multiple heads of a department, with every role a maximal number of occupants per organization is specified. This protects e.g. against forgetting to remove a previous holder of a role when adding the next successor.

Sometimes, a role is exercised by its holder in close connection with another person. An example is a secretary working in the legal department for a well specified professor. This professor's relationship is called "reference person". Of course, other uses for this reference person are possible.

A short note may carry additional information like the relation to a research group that has not an organization entry by itself. This short note has a purely informational character and is not evaluated anywhere.

`fct` *table*

This table lists all roles and their properties.

Column	Data Type	Description
fID	UINT	unique record ID
fns	CHAR(32)	role display name (short form)
fnm	CHAR(32)	role display name (for men)
fnf	CHAR(32)	role display name (for women)
fni	CHAR(32)	role display name (if sex unknown)
fnl	CHAR(64)	role name (for mail addresses)
fom	CHAR(1)	is role mapped to an organization? (yes/no)
fx5	CHAR(1)	include role in an LDAP (X.500) export? (yes/no)
fpr	UINT	priority if description of person allows only few roles
fon	UINT	number of role occupants per organization
for	UINT	number of role occupants per organization and all sub-organizations
fxc	CHAR(16)	time of entry creation
fxm	CHAR(16)	time of last entry modification

`mfct` *table*

This table assigns roles to persons.

Column	Data Type	Description
hID	UINT	unique record ID
hfi	UINT	ID of role (maps to fID)
how	UINT	ID of role occupant (owner)
hoi	UINT	ID of org
hrp	UINT	reference person (like a professor for secretaries etc)
hrn	CHAR(32)	reference number (like Dozenten-Nummer, Vorsteher-Nummer etc)
hds	TEXT(32)	short note
hed	CHAR(16)	expiration date
hxc	CHAR(16)	time of entry creation
hxs	UINT	entry status

Provisioned Systems

The accserv daemon

The accserv daemon is the main entry point to every provisioned system. It must be installed as a permanent background daemon process. It serves as a general gatekeeper (access control) and dispatcher for all requests from the URZadmin client.

File	Description
`etc/accserv.cfg`	The accserv configuration file. It defines clients (sources), services and permissions (which source has access to what service).A description is given below.
`bin/accserv.pl`	The main executable. Must be started at system startup time.
`accserv/<component>`	accserv components.
`logs/accserv.<date>.log`	Log files

Clients are identified by their IP address or a CIDR range of IP addresses (e.g. 10.157.10.240/28). On daemon start-up, a list of all configured client IP addresses is built. Any TCP connection sourced by an IP address not on this list will be immediately dropped.

The services define what kind of managed objects are accommodated at this server. They correspond directly to the account domains or mail domains within URZadmin. Additionally, the userv service implements the user controlled services like mail auto-reply. The login and config services are used by publicly accessible computers.

Every service (except userv, login and config) has a name, e.g. IGOR or AuthN. It is implemented by an accserv component: The service IGOR is implemented by the UNIX accserv component (or is type UNIX) and the AuthN service is implemented by the LDAP accserv component (or is type LDAP). Available accserv components are:

Component	Description
UNIX	Manages accounts and groups on classical (True 64) UNIX machines. Accesses /etc/passwd, /etc/groups and other central files directly. Cannot use the shadow file.
Linux	Manages accounts and groups on Linux machines. It uses the useradd/usermod/userdel utilities and system calls like getpwent and getpwnam.
LDAP	Manages accounts, groups, organizations and other objects in an LDAP directory.
RADIUS	Manages accounts in a RADIUS users file.
userv	Provide user services like mail auto-forward or SPAM-filter.
login	The login service is used when authenticating users of publicly accessible computers.
config	The config service is used by publicly accessible computers to obtain information on nearby printers etc.

It is conceivable that several services are provided on the same server. As an example, a single server could offer RADIUS accounts, an LDAP directory and local Linux accounts at the same time.

The configuration file etc/accserv.cfg

This identifies our accserv. The default settings are commented out.

```
Host:          MASER
#Port:         12012
#Bind-Address: 10.157.1.5
```

Define the directories for the accserv components and configuration files.

```
Root-Dir:       /usr/local/urzadm
Lib_Dir:        ${Root-Dir}/lib
Dialog_Dir:     ${Root-Dir}/accserv
conf_dir:       ${Root-Dir}/etc
```

Logging configuration: Maximal logging level and location of the log file.

```
#
# Log levels:   0  fatal error
#               1  permanent error
#               2  temporary error
#               3  warning
#               4  information
#               5  protocol
#               6  trace
#               7  debugging
#               8  more debugging
#               9  logorhoe
#
Log-Level:      6
Log-Dir:        ${Root-Dir}/logs
Log-Owner:      0  0
Log-Mode:       0640
```

Define the default read timeout (in seconds) that is used when waiting for client input. This value is used only when the source does no specify an own timeout. The timeout may be changed by clients, but cannot be raised above of max_timeout.

```
timeout:        60
max_timeout:    600


#
# Default access for all services
#
default_access: RO
```

The `service` table lists the available services. Included is information on how to invoke the service: the accserv component library module and the name of the entry subroutine. The library modules must be located in the directory specified by the `dialog_dir` configuration.

```
#
# Services
#   list of available services (account domains)
#
#  If no Password or "*", then set DefaultAccess to NO!
#
#Service:    Service-Name    File            EntryPoint      DefaultAccess   DefaultPassword
#
Service:     MASER           unix.pl.lib     xs_dialog       RO              **NeverGuess
Service:     PPP             radius.pl.lib   xs_dialog       RO              **NeverGuess
Service:     mail            mail.pl.lib     xs_dialog       RO              **NeverGuess
Service:     userv           userv.pl.lib    xs_dialog       RO              Never**Guess
Service:     RAP             unix.pl.lib     xs_rap_dialog   RO              Never**Guess
Service:     sts             status.pl.lib   xs_dialog       RO              LetMeIn
Service:     login           login.pl.lib    xs_dialog       NO              Ravenport!!
Service:     config          config.pl.lib   xs_dialog       NO              AnyTime
```

Here is the list of the clients that may use this server. They are identified by their IP address. The source-name is used in the permission mapping.

```
#
# Sources: only these hosts may connect to the account server
#
#        IP-Address/Mask      Source-Name    Timeout   Description
#
Source:  10.157.1.20        webmail            90   webmail - IMP access
Source:  10.157.2.32/32     betty              10   Test Workstation
Source:  10.157.1.5         maser              90   URZADMIN Server
Source:  10.157.2.154       petr-test          60   Petr Test fuer Unilogon
Source:  10.157.6.0/24      urz-public         90   URZ DHCP area
Source:  10.157.10.96/27    bioz-public        90   Biozentrum Oeffentliche Rechner
Source:  10.157.10.240/28   bioz-bib           90   Biozentrum Oeffentliche Rechner
Source:  10.157.24/24       pz-public          90   PharmaZentrum Oeffentliche Rechner
Source:  10.157.23.0/27     pz-kiwi-public     90   PharmaZentrum Oeffentliche Rechner
Source:  10.157.23.64/27    pz-guava-public    90   PharmaZentrum Oeffentliche Rechner
Source:  10.157.168.64/27   nursing-public     60   Fuersorgeamt Oeffentliche Rechner
Source:  10.157.209.0/24    ub-public          60   UniBib Public Rechner (UBP sub 209)
Source:  10.157.212.0/24    ub-intern          60   Unibib intern   (UB Sub 212)
```

The permission table defines what source may use what service.

```
#
# Permissions: what source may do what on services
#
#   Access: read only RO, read/write RW, no access NO
#           login: basic access to authentication server LOGIN
#
#Permission:     Source-Name           Service-Name           Access  Password
#
Permission:      betty                 MASER                  RO      NoDisclosure??
Permission:      betty                 MASER                  RW      NoDisclosure!!
Permission:      maser                 MASER                  RW      --Delavare
Permission:      maser                 login                  RW      *
Permission:      maser                 *                      RW      *
Permission:      petr-test             login                  login   No..Question
Permission:      pz-public             login                  login   Auto::didakt
Permission:      pz-kiwi-public        login                  login   Auto::didakt
Permission:      pz-guava-public       login                  login   Auto::didakt
Permission:      bioz-public           login                  login   Auto::didakt
Permission:      ub-public             login                  login   Autodidakt::
Permission:      ub-intern             login                  login   Autodidakt::
Permission:      urz-public            login                  login   Autodidakt::
Permission:      nursing-public        login                  login   Autodidakt::
Permission:      bioz-bib              login                  login   Autodidakt::
Permission:      webmail               login                  RO      Increase++
```

The accserv protocol

The protocol employed between account client and accserv (XCS: Exchange between Client and Server) is a simple text-based, line oriented protocol. A typical conversation between client and server contains the steps:

1: Client connects to a particular TCP port on the server.
2: Server checks whether this client may connect.
3: Client selects an accserv component.
4: Server dispatches the requested component and sends a confirmation.
5: Client sends a command.
6: Server replies with (optional) command output data and an indication of command success or error.
7: The command/reply cycle may be repeated as often as needed (steps 5 and 6).
8: Client and server drop the TCP connection.

Client Commands

The client may send the following commands:

Command	Description
`con`	The con (connect) command selects an accserv component to work with and sends a password to authenticate the access to the component. This is checked against the password in the etc/accserv.cfg file. The con command must be issued immediately after the TCP connection was established. Any other command will be refused at this stage and the connection will be closed. Once an accserv component was selected, it cannot be switched to another one within the same TCP connection.
`help`	Print a list of valid commands (Warning: not implemented on some accserv components).
`debug`	Server prints its internal data structures. This is deep magic. (Warning: not implemented on some accserv components).
`verbosity`	Set verbosity level for progress and debug messages.
`timeout`	Set a read timeout in seconds.
`mode`	The mode command controls two aspects of the connection: • The **flow mode** can assume the values "`checkpoint`" and "`stream`". It becomes effective when the client receives data for several objects. In checkpointing mode, the server appends a `J checkpoint` line after each data image for an object, and waits until the client replies with a `J continue` line. Only then the next object's image is sent. The reason for checkpointing is not to overrun the client with a large volume of data, e.g. when listing the entire content of an LDAP directory. In streaming mode, all the data is sent as fast as the server gets it and it does not wait for the client. Checkpointing may conserve the client's resources, but it slows down the client/server communication to a high degree. It is better to use spool files on the client side if a high amount of data is expected. • The **data mode** has the legal values "`sequential`", "`indexed`" and "`keyed`". It is used to select the format that the server uses to sends its reply data back to client. See the **`D`**, **`I`** and **`K`** server reply codes below. The default mode setting is "`mode stream sequential`".
`data`	The data command is used to send data images to the server. Images contain object (account, person, organization, etc.) information like login name, a new password or a peson's organizational status. After the data command, the server expects any number of image lines, until terminated by a line containing a single dot "`.`" (white space may follow the dot). Normally, after a data command, an operation like `modacc` or `addgrp` is issued to process the image data. Several data commands in sequence accumulate the data into one image.

`modacc`
`addgrp`
`lstacc do`
	Operations like these do the real work on the server. They are accserv component specific. Some need image data to operate on (e.g. when creating an account), others operate without data (e.g. listing all accounts on a server). The neutral `do` operation processes the image data and takes the real operation from each image's `action` field.

`quit`	The communication between client and server is terminated.

Other commands may be added for individual accserv components if needed.

After each client command, the server replies with optional command output (e.g. when a list of objects was requested or as a confirmation of the account changes made) and a command success status indication. The accserv's reply may be interspersed with debugging information, depending on the verbosity level.

Server Replies

The server reply is a sequence of lines. Every line starts with a one character code, followed immediately by a numeric severity and a colon "`:`". Then, data or a text message may follow.

```
<code><severity>: <message>
```

The code classifies each line as carrying reply data, informational message or an error/success status. The legal values of line code are

Code	Description
P	Progress message. In the web interface, the message may be displayed to the user (depending on current verbosity settings)
C	Debugging message. In the web interface, this message is included in the HTML code as a HTML comment (`<!-- message -->`) only.
D	Sequential data message. All D line messages for a command are collected in an array data structure (`@srvdata`) in the order as they are received without any transformations. D line format is requested by the "`sequential`" data mode.
I	Indexed data message. The message part has the form `key: value`. These paires are collected in an associative array `%srvhash` (`$srvhash{key} = value`). I line format is requested by the "`indexed`" data mode.
K	Keyed data message. The message part has the form `ident key: value`, where the `ident` piece identifies different objects. The data is collected in a nested hash `%srvkey` of the form `$srvkey{ident}{key} = value`. This format is useful when the replies for multiple objects are to be collected in one data structure. K line format is requested by the "`keyed`" data mode.
J	Checkpoint message. Used with the "`mode checkpoint`" connection setting. See the description of the flow mode above.
S	Command success indication. This line tells the client that the server finished processing the command and that no severe errors were encountered.
E	Error indication. A severe error was encountered. The connection will be closed immediately.

The severity levels are:

Severity	Meaning	Description
0	fatal error	Error on server, manual intervention necessary
1	permanent error	Next try only when database entry changed
2	temporary error	Next try any time (E.g. a common file was locked, may be free now)
3	warning	Operation on entry done, something was not perfect, but still acceptable.
4	user information	All is fine, no complains.
5	protocol	Talk between client and server
6	trace	Trace program flow
7	debug	Debugging
8	more debug	More debugging
9	logorhoe	Extreme debugging

Normally messages with severity level of 4 or lower should be displayed to the user. Higher level messages are internal or debugging stuff.

The protocol is strictly line oriented. Any lines that do not fit the <code><severity>: <message> scheme are interpreted as protocol error and terminate the client/server connection immediately.

As an example, the full client/server exchange for changing an account's password is displayed below:

Client:	attempts TCP connection to host "ldap1.urz.unibas.ch", port "12012"
Server:	accepts TCP connection
Client:	con LDAP password
Server:	S6: Hello, maser, what's up (60)
Client:	timeout 300
Server:	S5: Timeout is "300" seconds
Client:	verbosity 90 90
Server:	S5: Verbosity level is "90" (msg), "90" (dbg)
Client:	mode indexed
Server:	S5: Current mode "stream" "indexed".
Client:	data
Server:	S5: Erwarte Daten gefolgt von PUNKT-Zeile...
Client:	!transferid: 1178470275-29461-1 regid: a.18.16586 ref: a.AuthN.zimakn login: zimakn hash: ki6lear ownertype: pers action: modacc .
Server:	S5: "519" Zeichen erhalten.
Client:	modacc
Server:	P7: xs_ldap_entry_locate: account "a.AuthN.zimakn": entry found at … "uid=zimakn,ou=people,dc=unibas,dc=ch" P7: xs_ldap_entry_modify: account "a.AuthN.zimakn": replacing attribute "userpassword" P7: xs_ldap_entry_modify: account "a.AuthN.zimakn": replacing attribute "sambalmpassword" P7: xs_ldap_entry_modify: account "a.AuthN.zimakn": replacing attribute "sambantpassword" P7: xs_ldap_entry_modify: account "a.AuthN.zimakn": replacing attribute … "unibasChModifyTimestamp" (20070506185116) P5: update for account "a.AuthN.zimakn" suceeded P7: xs_process_worklist: severity "5", action status "modify for account … "a.AuthN.zimakn" suceeded I5: !transferid: 1178470275-29461-1 I5: dn: uid=zimakn,ou=people,dc=unibas,dc=ch I5: enchash: {crypt}qsFWuDl9FUa2. I5: gidnumber: 1170 I5: login: zimakn I5: gecos: Nina Zimak I5: homedir: /users/staff/urz/zimakn I5: uidnumber: 17292 I5: mailbox: /users/staff/urz/zimakn/Maildir/ I5: shell: /bin/bash I5: severity: 5 I5: actionstatus: modify for account "a.AuthN.zimakn" suceeded I5: ref: a.AuthN.zimakn P7: xs_process_worklist: total severity "5", total success "1" S5: - 1 objects
Client:	quit
Server:	S5: Das war's.
Client:	closes TCP connection
Server:	closes TCP connection