SecurityLib 1.0

by

MarVic

Administrator manual

Table of contents:

1. Introduction

This document is meant for people, who will be managing privileges of all other users of the program. They will be creating hierarchy of users, watching over right leaks, widening possibilities of others. For usual users they will be masters.

The main idea of the system is quite standard: each user of program has his account with password, and each user has its rights and parameters, which are stored in database. You can assign user to a group, which has common rights, but some of users can have privileges or constraints inside a group. You can create hierarchy of groups, which enables you to control which users can change other users rights.

In my opinion the greatest advantage of this system is that you can give your users wide autonomy in adding new users, removing them, setting theirs rights and parameters, and still, they can do only things you let them to. You can precisely tell which users, rights and parameters user can change, and then he can't gain any benefits from making any operations.

Above makes the system very scalable, if your company has 10 employees now managing them seems to be easy. You remember theirs names, know what are theirs duties. But in year or two, your company can grow up to 50 or 100 of workers and you will not be able to control all of them. But with SecurityLib you can give the boss of part of your company right to manage his workers in frame set up by you.

This implies that you don't have to worry about changing your program and security rules in the future.

Two models of security

SecurityLib can work in two models. In the simple one, there is one privileged "god" group, whose users can manipulate everyone else's rights and parameters. All, who are not the god, cannot change any of parameters and rights of somebody else.

In the advanced one, for each group there is maintained a list of groups that are placed lower in the hierarchy, that is, groups, whose users' rights and parameters can be changed by users of group that is higher.

I will write it again: if group A is higher in the hierarchy than group B, then users of group A can manage users of group B. We can say that users of group A are gods to users of group B or that group A is god for group B.

This relation is not $transcendental . That means that if group A is god to group B, and group B is god for group C, that DO NOT implies that group A is god for group C. If you want to put group A higher than group C, you have to do it explicitly.

There are not many differences between these two modes, and, in fact, you don't have to change any settings to use one or another mode. The difference is only in way you use settings of group. This is described in section "Setting up group hierarchy".

Getting Started - Security Administrator Window

In your program documentation there must be information how to invoke the main window, called *Security Administrator*. It centralizes all functions you need to manage security simply and effectively.

The window has five tabs: Users, Groups, Rights, Parameters and Information. The Users and Groups pages are for managing, adding, removing, changing rights and parameters of users or groups. The Rights and Parameters pages are mostly for developers and programmers of your application. As an administrator, you can add new rights and parameters, but it will not be used by your application unless programmer writes it to your application. All you have to do at these pages is to set path for custom parameter editors. See chapter "Custom parameters editors".

At the bottom of window there are three buttons - OK saves all changes and closes the window, Cancel closes without saving any change and Apply writes changes to database. Depending on your application, Apply button may be all the time disabled - then your changes are immediately wrote to database; OK and Cancel buttons do the same thing - closing the window. If Apply button is enabled then you have made some changes that were not saved to the database - when you press Cancel they will be discarded.

Also at the bottom the note can be displayed saying that some operations may need to write to database all modifications you have made prior to invoking operation. Of course, you can't discard

changes you verified with Apply button using Cancel button. Note will not be displayed if all modifications all saved to disk in the moment you leave current record.

2.Administrating users

Getting started

You administrate users on the Users tab of the *Security Administrator* Window. In top left corner there is a grid populated with all users in the system. You can only manipulate these users which are marked with green color. Those green ones are members of groups that are lower than your group in hierarchy (so you are god to and only to the green users).

Below that grid there is a list of available groups; group of currently selected user will be highlighted.

On the right there are two grids displaying right and parameter settings of current user.

User accounts

Names

Each user has its unique name in SecurityLib database, and he is identified basing on it. Name can have maximum 16 chars, case sensitive. User is supposed to enter it in the login window along with password.

Each user also has full name, which is also unique to user, but can have up to 50 chars. It is thought for describing user more precisely, it may be his first and last name or title.

Passwords

Passwords can have up to 16 chars. Passwords are case sensitive. Password may be empty - has the length of zero. Passwords are stored in security database in encrypted form (double DES), so you don't have to worry about having it hacked.

Password options

Each user has five additional options or rights connected to passwords. All of them have three states: on (right granted or forbidden), off (right not granted or allowance of something), and group default (on gray background; if there is a tick inside then the option is on) - this means that when corresponding values of group are modified, values of this user will reflect those changes. The options are:

Name	**Meaning**
Password Control	Allows user to change others' passwords and options
Change password at login	Forces user to change password at next login if the user can change password (which is denoted by the Allow Change Password checkbox).
Disallow null password	User can't have empty password
Allow changing password	Some of users, for example guests, can't change password
Account disabled	Administrator may temporarily disable account of user to prevent him from logging in.

Guest accounts

You may want to create guest account, for example, to enable customers to view the prices of your product. In this section you will find precise directions.

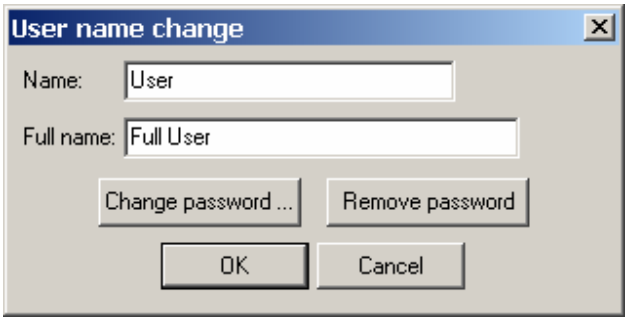
After creating new user set his password options to the following:

- Password control to false
- Password can't be empty to false
- Password can't be changed to true
- Change password on next login to false
- Account disabled to false

You may want to create separate group for guest accounts.

Changing user name and password.

In order to change user name or password, select desired user in the grid called *Users* on *Users* tab, and double-click it, or press *Modify* button. Window similar to this should be displayed:



You can enter new user name and full name in text boxes, click the *Change password* button to modify password or click *Remove password* button to set the password to empty.

The two latter buttons will be enabled only if you have *Password Control* right.

User rights and parameters

You can have four basic rights to every sector of application: right to view data of that sector, add new data, modify existing data, and delete data. Some of them may be irrelevant in specific context, for example if there is a screen only with prices of some items, you can only view and modify that data, because adding and deleting from list of items can be done other form. Each right composed from four basic rights has it's own unique name, and can be different for each user.

Use of each basic right is application specific - that is, your application programmer decides, what will be disallowed when you don't have modified right to some part of app. For more information about meaning of basic rights, see your program documentation.

In contrast, parameters are not divided into basic ones, but into types:

- Default - which may store 16 char string
- Integer - which can store numbers from about -2,000,000,000 to about 2,000,000,000
- Boolean - which stores only two values - true or false,

- User defined parameter types - which are in fact little dialog windows presenting information stored in standard parameters in their own special way.

Each parameter has it's own unique name, and each user can have it's own value of each parameter.

For example, parameter can be used to store your name and address to print them in your faxes or bills. Also, some rights can be stored here, for example right to print reports - storing it in rights section would be waste of disk space, memory and time, you need only one right, not all basic four.

Right and parameter values are shown in the right panel of *Security Administrator*.

When parameter or right name is written in blue or cyan the value of it is taken from group setting. When it's black or red, the value is set only for current user and changing this parameter or right for group has no effect on you.

If the user which is logged in has no right to modify data using the *Security Administrator*, but he has the view right, he can modify parameter values belonging to him, but only if the parameter is not red or cyan. It is used when you want your users to supply some information about themselves. You don't have to enter them as they do that. And they can't change any other thing.

Color	Default	Weak user modify
Black	-	+
Blue	+	+
Cyan	+	-
Red	-	-

Protecting rights

Even if the user has the right to modify data in *Security Administrator*, the systems disallows him to give somebody wider privileges that he has. However, he can exclude all of rights, but only of the users that he is god to. This permits you to grant modify right to most of users and don't worry about users taking over the system, if you have properly set the hierarchy of groups (in simple model, user is not able to modify any of the rights).

Protecting parameters

The system also disallows users from setting certain values of the chosen parameters. The general rule is that user can't set the value that gives more privileges than his own one. But some of the parameters have only information aim, so there are free to change. Those, which values are controlled by the system, are called protected parameters. You can learn which parameters are protected by showing the *Parameters* page, and checking the *Protected* column of the grid.

Because the system can't know what privileges each value of parameter gives, it makes decision if cancel the change basing on the type of parameter. In the following table you have specifications for all built-in types.

Default	The value other than user's one is considered giving wider privileges.
Integer	The higher integer value the more rights it gives. If the parameter is not a valid integer, it is treated as the smallest value.
Boolean	System disallows setting true value of this parameter if the user's value is false.
Custom	All subparameters are treated as separate parameters. The custom parameter itself is considered to be of default type.

Changing user rights and parameters

To change user parameter or rights, first select user from the list. Now, just click on appropriate checkbox to change basic right or click in the *Values* column of parameter list next to chosen parameter. Now you can enter new parameter value. Depending on parameter type, at the right side of field there may be a button for unfolding the list of possible values (currently only for boolean parameter type with values True and False) or button with three dots - for invoking the custom property editor.

Restoring group value

Sometimes you may want to restore the original value of parameter or value, that of group. You can just enter that value in appropriate field, but then the value of field will not change when you modify the value of group parameter. If you want parameter to maintain the group value all the time, you have to press the button to the right of the title of grids. This will change the color of row to blue or cyan, indicating the value is group dependent.

The system disallows to restore to value, which would violate the rules of protecting.

NOTE: In case of custom parameter type sometimes even if the color of parameter indicates that it's default some of sub parameters may not be default. If you want to be sure all subparameters are default, press the *Default Parameter* button as described in previous paragraph.

SecurityLib specific rights

The is a number of rights that has special meaning to *Security Administrator* dialog. They are used to restrict access of users to the dialog. SecurityLib recognizes them by it's name (not full name, which is shown in the grids), but theirs full names should be same as described above (it depends on the Security database).

The *Security Administrator* right enables you to assign basic rights for the part of *Security Administrator* dialog responsible for viewing, adding, modifying and deleting of users and groups. So, for example, granting the modify basic right enables specified user to modify user and group names, changing other users rights and parameters, but only within the range of theirs privileges and only those groups, which are below that user. The delete basic right besides removing users and groups allows to restore the group value of parameter or right. If user has no View basic right, he will not be able to switch to Users or Groups pages.

The *Security Projecting* right has similar meaning, but it refers to managing parameters and rights. This right should only get administrators and developers of parent application. You should always be extremely careful when granting the right, as users having that right can get almost any right in the system (by deleting an adding the right or parameter). Most users shouldn't even have the View basic right.

User's group

Each user has to be a member of exactly one group. You can check what group user belongs to by selecting a user from user list at the *Users* tab. User's group is selected in the groups list below users list.

Groups are provided for your convenience. You can create groups to make managing users quicker and simpler. Now you don't have to remember each user and his rights. You set his group to one of defined earlier and his rights default to group rights. Of course, you can change single user rights at any time. And still, those unchanged rights will be the same like the group's one, even if you change group's rights. The same thing is with parameters.

For example, the predefined Administrators (Admins) group has all rights you can imagine. So its members should be only those users, that are meant to have such rights. And you don't have to set the rights for each Admins member separately.

Setting user group

It is described later in part Adding a user.

Changing user group

Select the user from the list. Then find appropriate group at the group list at the bottom of window.

Click desired group to select it. Small window will pop up with confirm query. Click *OK* button. If the group you selected is god to group your group is not to, then message will be shown. In such case, you are not allowed to change the group, because you or the user, whose group you are changing, would become god to group, that he was not to before.

If you didn't get the message about not being able to change the group, the system will change the group. Then it will check for any right inconsistency. The user, whose group you are changing, cannot have wider rights than the logical or of your rights and that user rights. Thanks to that, you are not able to give someone right you don't have. The same thing is with protected parameters. The rules of protecting of parameters were described in previous chapter.

Adding a user

You can add user only to groups you are god to. Also, you can’t add a new user to group that is god to group you are no to. This ensures a user cannot obtain greater rights by creating new user and logging in as him.

To add new user click *New user* button below the user list. Enter the name (which will be used as login name) and full name of user, and select desired group of user from the list of available groups in the table. Avoid adding users with identical names or full names. It will be detected after the try to apply data. In such case, rename the user and apply changes.

If there is no groups that you can add user to, the message will be displayed.

Removing a user

You can remove only users that are members of group you are god to. If you are sure you want to remove user, select him in the user list and click *Delete user* button.

3.Administrating groups

Groups are to ease the work connected with setting the rights. Dividing your users into groups is essential to keep track of what users can do what, and which ones' rights should be changed.

Their second task is to allow to move and distribute the brunt of work with granting rights to your staff. If your firm is growing bigger and bigger you have other thing to care than giving rights to each worker you employ or move to another post. And now, you can just cede this duty to your assistants or staff chiefs or just direct boss of the employer. And the most important thing is that giving somebody rights to administrate his staff do not decreases the security of the system, do not allows him to mess something up, do not allows him to receive wider rights than you gave him. He will stay just were he was when you left him.

This is achieved thanks to the hierarchy if groups. The group can change most of properties of its children (that is, groups, which are in hierarchy *below* that group) and theirs users. You can have many levels of hierarchy, and each group can have different set of children and parents.

Managing groups

This part covers the topics of adding, removing and modifying groups. Also it explains how to change group's rights and parameters.

Adding a group

A user can add a group if he has insert basic right to *Security Administrator* dialog. Grant this right cautiously, as adding too much (for example, 1000) groups can slow down the system, or event make it impossible to use.

To insert new group click *New group* button below groups list. In following dialog set the unique name and full name. Set desired password options by clicking password options button. Initially the password options are set to the following:

- Password control to false
- Password can't be empty to true
- Password can't be changed to false
- Change password on next login to true
- Account disabled to false

This is the most common setting.

Now you have to choose the values of rights and parameters of the new group. For your convenience there are implemented means to copy rights and parameters values from other group.

You can designate four basic rights in the *Default rights* frame. You can choose out of three checkbox states. Checked means the right will be always granted unchecked means the right will be always revoked. The grayed state may have two meanings depending on the radio buttons below the checkboxes. If the option *Use right defaults* is selected, basic right with grayed state will derive their values from default ones of each right, which were set by the developer of your application (you can view them by showing the *Rights* page). If the option *Copy from group* was selected, grayed basic rights will be copied from group selected in combo box placed in the frame.

Similarly, you can choose whether to copy parameter values from chosen group or use parameter defaults.

Group's rights and parameters

After adding new groups you can set it's rights and parameters. They will became default for new users of that group, and the user's values will change every time the group's value changes, except the case user's value was explicitly set and is not blue.

What you have to do now is simply choose the Groups tab and use *Rights* and *Parameters* lists as if they were user's ones. You can change the rights and parameters values, of course, you are restricted only to those values allowed by your rights.

Groups also have Password Options similar to those of user. Click the *Change* button and then *Password Options* button to display the dialog the same as in User case, only there is no grayed state.

Setting up group hierarchy

There are two models of hierarchy. In the simple one there is only one group, which can change every other user and group; and any user not in this group cannot change anything. This is the default method. The *Admins* group may modify everything. If you don't have many users, this is ideal one for you. And then, migrating to the advanced model is very simple, if you only divided your users into group carefully. In fact, it is pretty transparent to you - if you simply close the *Group Hierarchy Editor* window after adding a group, you still have rights to that group.

In the SecurityLib system there is implemented a hierarchy only for groups, not for users. That means if group A is higher in the hierarchy than group B, then all users of group A are higher in the hierarchy that all users of group B. This simplifies managing users, as you only have to define few groups, and the assign users to that groups. Also there is improvement in speed.

Note, that in SecurityLib system user is be a member of exactly one group. In particular, that means that if user is a member of specialized group he cannot be in more general group (for example, if user is a member of *Accountants* group, he cannot be a member of *Users* group, although he is a user).

As it was written before, the group hierarchy thing is mainly used to make managing users easier. So, you should divide your workers to some logical groups - you have to remember that all users in group should have similar rights, which will ease controlling them later.

Also you should keep in mind, that casual users shouldn't be able to modify rights of other users in their group. So you should use the mechanism described below.

Supervising groups

Let’s suppose for a moment, that you want to have some group to be managed by one of your directors. But you don’t want them to be able to grant all rights, but only specific subset. In this case you may create a spearate group for the director, and give him all rughts he should be able to grant to his employees, and also a right to create new groups. And the he would create groups for his employees and give them only those rights you gave them, but those users would have no right to change others’ rights or create new groups.

But, from the other hand, you can place the director and his emploees in the same group. The employess would simply had no right to modify dialog, and the director would. The method you choose should depend on the number of directors and the range of rights they grant.

Modify all right

There is a special right, which allows users of the group to change completely everything: groups, users, protected parameters, hierarchy of groups. In addition, when user with that right moves user to other group or add new one, there is no check for right violation performed, so even if the operator has not all rights, he can create user with them.

This right applies to group, which means that you cannot grant it to only one user, not to whole group. It is denoted by group being both above and beyond itself. So if the group has a arrow pointing to itself, it means that it can modify all things. If you want to grant some group the right, you should add such a relation, there is no difference if you use strong or weak relation. You can only grant it if your group has that right itself.

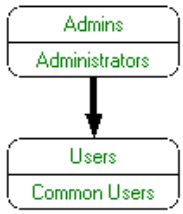
There should be only one group with that right, and that is the *Admins* group. Also, because each member of *Admins* group can remove any other user, including other Admin, in the group should consist of as small number of users as possible. It is better to create one group with *Modify all* right and several groups with all rights, and being beyond all other groups except themselves and *Admins* group, but without *Modify all* right. Then the users of those groups cannot interfere themselves.

Modify all right is symbolized in *Group Hierarchy Editor* as an arrow connected to its group.

Group Hierarchy Editor

When you have divided your users into groups, and have a draft of relations between them, you can start to implement it in SecurityLib. Bring up the *Group Hierarchy Editor* by clicking on *Hierarchy* button on *Users* or *Groups* page. You will see a hierarchy area with toolbar at the left side. The groups are presented as rectangles, and the relations between them as different lines arrows:

- Black thick line and arrow means that the group at the end of line without arrow, and above all groups the latter is. So it is like that the first group inherits rights to modify all groups that second group can, and moreover, the first group can modify the second. This is called a strong relation.
- Black thin line denotes that the, but has no access to its children. That means that this kind of relation is not inheritable – connecting two groups with this arrow means that only the groups being connected are modified. This is called a weak relation.



Example: group Admins is above group Users and all of groups that group Users is above (because of thick arrow).

- Grey thin arrow means that the group at the end with no arrow has no access to the group at the end of line with arrow. This is used when you want to exclude one group from being modified by a group that has a strong relation to one of parents of that group. This is called a negative relation.

You are able to modify only those groups that you have access to. They are marked with green color. You can move that groups and add relations between them. However, you cannot do something that would violate the rules of the system – for example, add a relation is such a way that it would give you access to a group you had no access before.

You can use toolbar or popup menus of groups and relations to add, revert and delete relations and groups. If the option or command is inactive – then you have insufficient rights to perform the operation.

To add a new relation, you can click the specific button on a toolbar or choose the option from the parent group's context menu. If you used the first method, click the parent group and drag the arrow to the child group. If you used menu, just click the child group. You can also double click the toolbar button to hae the option locked – this is useful when you want to add more than one relation.

To delete a relation or a group, select it and then press *Delete* key or choose *Remove* from context menu. To add new group, choose appropriate option from menu or toolbar. The dialog window will pop up. At this time you have to only supply the most important data about group – the rest of information you will have to enter after finishing work with editor (you can also cancel the addition of group or rename it at that time). So when you are planning a hierarchy, you don't have to worry about entering all data, that is irrelevant at the moment. To modify a group just double click it. To change a kind of relation, right click it and choose appropriate option from context menu.

You can exit from the editor by normal Windows way (by clicking the button with cross on it), or choosing one of toolbar buttons:

- it saves the changes and exits the editor.

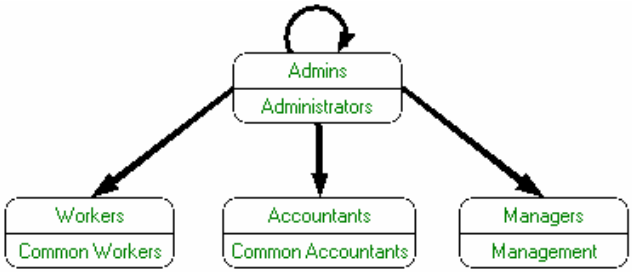
- this one discards all changes and closes the window of editor.

The *Admins* group and the *Admin* user are predefined with *Modify all* right.

Examples

Simple model

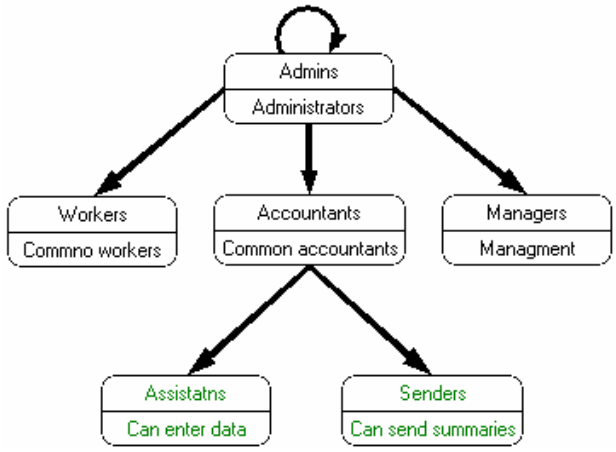
In simple model, you have to do nothing with the hierarchy. Everything will be done by the system.



But the cost of that is you have a very simple, one-level hierarchy, like on the picture below.

As you can see, the only thing that differs the *Admins* group from the rest is a *Modify all* right and strong rights to all of other group. All new groups added by you as a member of Admins group will look like that on the picture (it happens like that because the newly-created group is below group of its creator by default). So if you won't give right to create new groups to other groups than Admins, everything will look like on the previous picture.

But what happens if one of users of Accountants group has a right to add new groups? Well, he can create one. The picture will look like this:



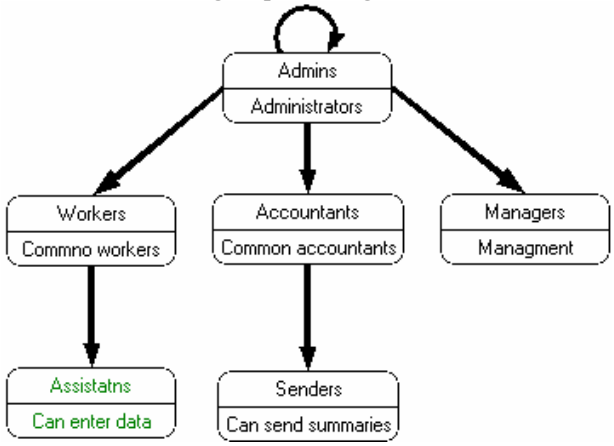
All the user of Accountants group did was adding two new groups. He didn't mess up with the hierarchy. As we can see, the both Accountants and Admins group has by default right to both new groups. That's the general rule – the system will always grant the group, that added a group, right of strong relation to the new group. If you do not want something sophisticated at the time, you can use simple model – which means doing nothing, and let SecurityLib set up default hierarchy for you.

So, let's summarize – by using the simple model, which means doing nothing with the hierarchy – you get the standard tree-like model of hierarchy, which is good in many cases, when you don't want anything special from your security system.

Advanced model

Modification of hierarchy

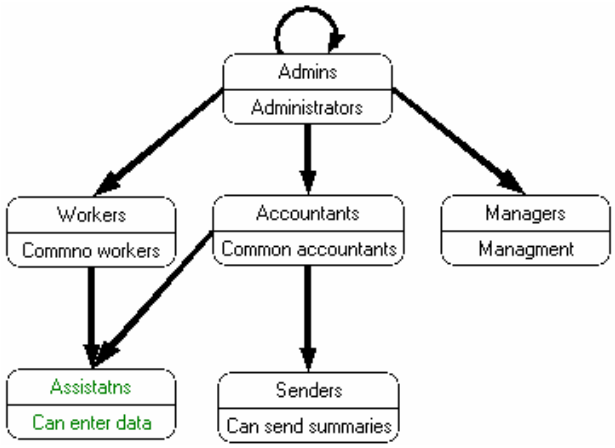
When you want to get something more from your hierarchy than a simple tree, you can do that by modification of hierarchy of groups. There is no "switch" to advanced model, this is the way the possibility to change relationships between groups is called. Usually the default hierarchy is all right, but sometimes you may want to change some relation. For example, if you wanted to change *Assistants* group to be depended on *Workers* and not on *Accountants*, all you have to do is to drag the end of line connecting Accountants group and Assistants group, so that the end without arrow will be attached to Workers group. Note, that none of *Assistants* group rights was not revoked – even if *Workers* doesn't have some of them. And even if *Workers* group had rights to add new users and tried to add one to



Assistants group – he wouldn't have wider rights than the *Workers* group. He would have the wider rights from default group settings overridden by user settings. The same is with changing user's group (for example, moving a user from *Workers* group to *Assistants*).

Multiple parents

Now, let's suppose, that *Accountants* got a new right from *Admins*, which, for example, is connected with some new accounting feature of main application. *Workers* didn't get this right, because they do not need it. *Assistants* do need that right but *Accountants* cannot give it to them because they have no rights to do that. There are two possibilities now: Accountants can ask Administrator to grant the right to *Assistants* or *Workers* can allow *Accountants* to access *Assistants* by adding a relation. The first approach is not very good: Administrators can get very annoyed by constant modifications of rights. So let's look how to implement the second one:

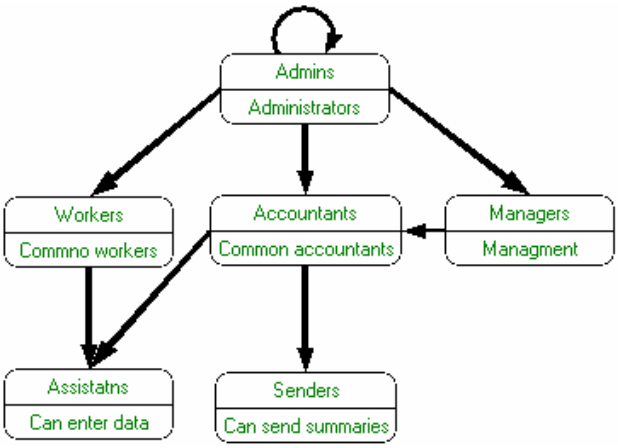


All that *Workers* have to do is add a strong (or weak) relation from *Accountants* to *Assistants*. Note, that *Accountants* cannot by themselves just make other group be below them. Now *Accountants* can grant the right to *Assistants* group as they are above them.

Weak relation

Now we will take a glance at the two other types of relations: weak and negative. As it was written before, weak relations are like strong ones, but the difference is that the group that is above is allowed to modify only the target group, while with strong one the group could also modify all of groups that are below target group. In that sense weak relations are less powerful, they give narrower rights.

Let's focus on our example hierarchy. Now *Managers* want to have access to Accountants group in the sense they want to be able to remove some of accountants from the group in case they defrauded some money. But assistants cannot defraud money, so *Managers* do not need access to that group. In fact, they should have no access, because they could mess something with internal arrangements between *Accountants*, *Assistants* and *Workers*, but if we used strong relation, they could do that. So, the use of weak relation is the best choice:

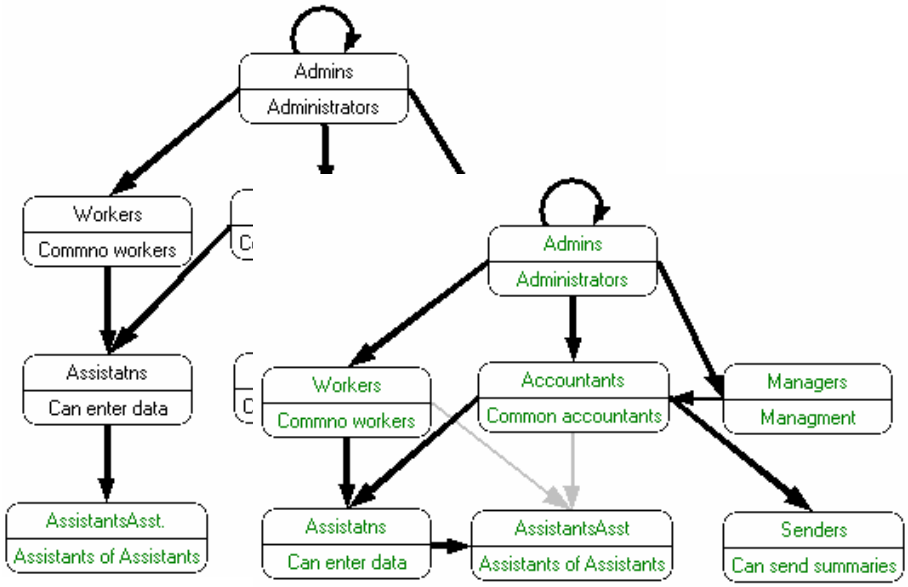


The relation should be added by *Administrators*, as *Managers* cannot do that – the modification changes the security model. Of course, unless Accountants agree to the supervision of *Managers*.

Negative relation

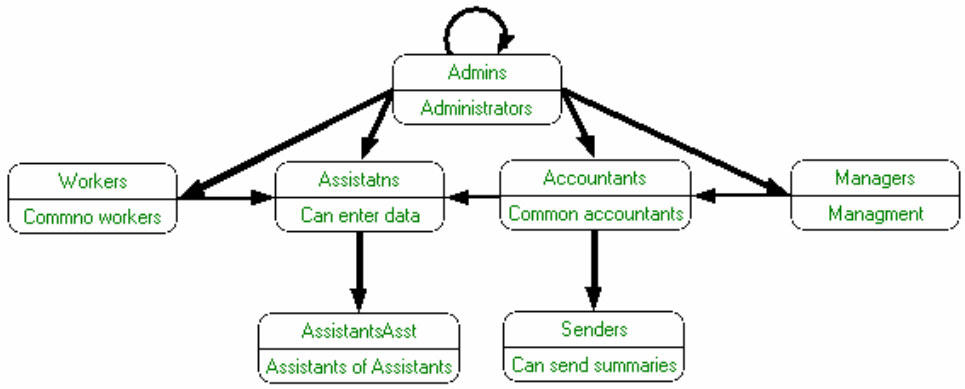
And now, the negative relation. This relations disallows the source group to modify the target group. It can used when some group can modify another because it inherited this right from strong relation, but it is needed to eliminate this relation.

Let's suppose that Assistants created new group: *Assistants Assistants*:



Both *Workers* and Accountants has access to it – but the group is to be local – only for *Assistants*, because they are supervisors of their assistants. So, they have to create negative relations with that groups. But they cannot – you cannot create a group outside the jurisdiction of your parent groups. So, *Assistants* have to ask those groups to add the relations or ask *Administrators* to do that. Then the hierarchy would look like this:

Now neither Workers nor *Accountants* can access *Assistants of Assistants*. Note, that this situation could be accomplished in another way: We could change the relations between *Workers* and *Assisntants* and/or *Accountants* and *Assisntants* to weak. In this case, Those groups would have no access to *Assistants Assistants*, too. But then there should be strong relation between *Admins* and *Assistants*, else that group would have no real parent. This situation is illustrated on this picture:



The choice between these two methods depends what the relations really mean. The best way to choose the right one is to predict what functionality will be needed in the future. You have to think about groups that can be added, and place them in your hierarchy. The hierarchy, which needs fewer changes after adding new groups is better. But it depends also on your view of structure of groups. And, it is not matter of life and death – you can always modify it, and by the time modifications will get complicated you will get enough experience to deal with such problems.

4. Custom parameter editors

Security of editors

The custom parameter editors have access to most of *Security Administrator* functions; it is essential for security and integrity to be sure no one has substituted the right editor with trojan horse. Such editor would be able to change rights or parameters values or get the protected data (although it can't theoretically change values of other users, it is extremely dangerous to let the wrong editor in). There are two main ways of protecting the system.

Changing place of editor files

If you are using network environment or file system which supports restricting access to files, you can change the location of custom editor files to one that is protected form other users. This may be network drive or System directory of local Windows NT system on NTFS partition.

After that, you have to tell SecurityLib, where the editor files are now kept. On the Parameters page on *Security Administrator* click Parameter Types button. Parameter types editor window will pop up. Select desired parameter type and change the path by entering it manually or clicking button with dots to show Open file dialog. Answer no to the question of auto detecting function names - if you are changing only the directory names should be already entered. You can also enter only the name of editor file without path – in such a case file will be searched for in standard directories, for example local system or system32 directory.

Using editor key function

There is provided method for verifying the editor. At first use of the editor some string is passed to the editor. It should encrypt it and pass it back to SecurityLib (the encrypted string is called key). Each next time the system will load the editor it will check first if it encrypts that string in the same way. This ensures that the editor file was not switched to another.

To make use of that function, your editor has to support it. Check its documentation. The second thing is it should be installed using the *auto detect function names* feature. During the detection, the string also will be encrypted and stored in the database. If you are not sure if the editor uses the option, open the

parameter types editor and set the parameter path again. You may need to change it to some other, exit the text box, answer no to the question of auto detection, enter the correct path, and then answer yes to the question.

5. Other

Other documents:

"SecurityLib info"	Basic information and introduction.
"User's Manual"	Information for users of programs based on SecurityLib.
"SecurityLib Administrator Manual"	Information for administrators.
"Interface and programming guide"	Information for programmers.
www.marvic.prv.pl/securitylib/	SecurityLib website.