Permissions and Access Control in CmapTools

Technical Report IHMC CmapTools 93-03

Alberto J. Cañas, Greg Hill, James Lott, Niranjan Suri
Institute for Human and Machine Cognition
40 South Alcaniz St., Pensacola Fl 32502
www.ihmc.us

Introduction

The *CmapTools* software suite is an environment that enables and encourages collaboration and sharing in the construction and manipulation of *Knowledge Models*[1] based on concept maps (Cmaps). In a shared environment, it is important that the owner have control over the type of

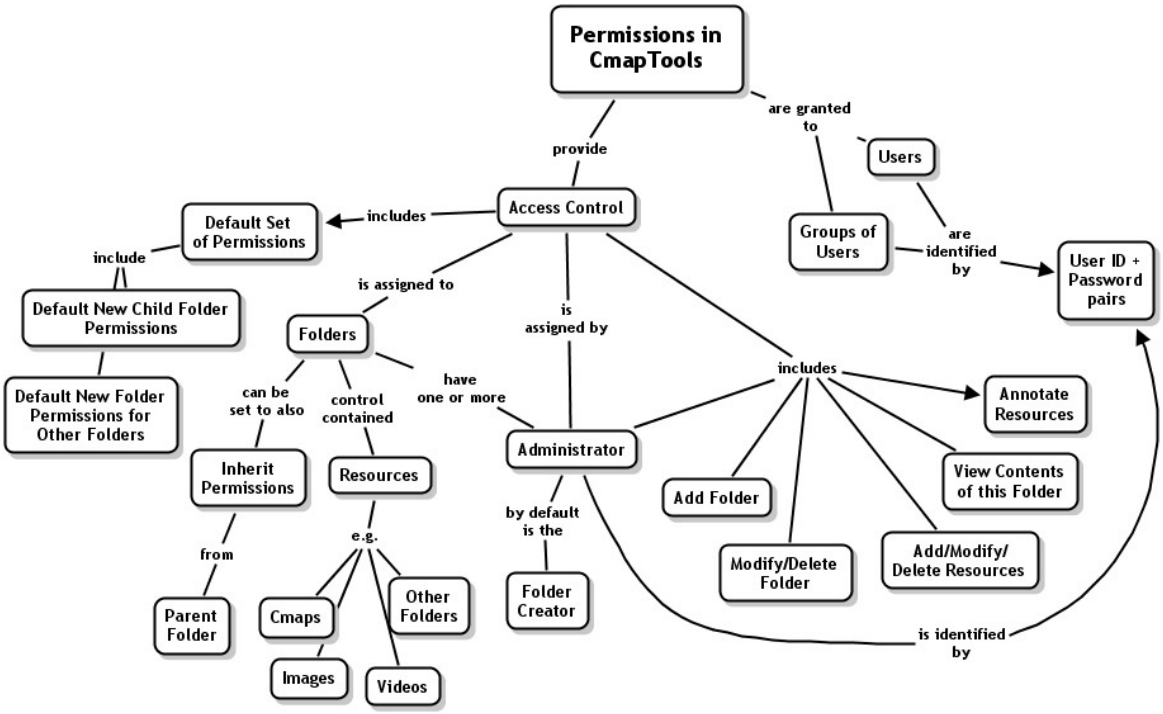

Figure 1: Concept Map on Permissions and Access Control in CmapTools

[1] A *Knowledge Model* is a collection of concept maps and associated resources about a particular domain of knowledge.

access others have over shared Cmaps and resources. For example, all members of a team may have permissions to modify the Cmaps in a *Knowledge Model*, while others may only have permissions to *View contents of this folder*, which allows the user to open and display them. In a classroom environment, each student may be the only one with *Add, modify, and delete resources* permissions over Cmaps in his/her personal folder, while all students may have *View contents of this folder* and *Annotate resources* (allow the user to add comments to a Cmap) permissions over each other's folders, fostering peer-review. Alternatively, each team of students could have its own folder with members of the team having *Add, modify, and delete resources* permissions, while teams could have *Annotate* permissions on other teams' folders.

The *CmapTools* permissions scheme is a simple yet powerful mechanism by which users can control who can access their Cmaps and resources, and the type of access they are allowed. Figure 1 summarizes the scheme.

Folder Level Permissions

Permissions in *CmapTools* are granted or assigned at the folder level. There is no mechanism to provide different permissions to resources within the same folder. This means that the access restrictions for any resource are determined by the folder in which it is contained. At the top level of a *Place*, the notion of a "root folder" for the *Place* exists, although it is not displayed with a regular folder icon in the *Views* window. Its permissions are set and accessed via the *Place* icon. Resources at this top level have the permissions specified in this "root folder".

In Figure 2, all the Cmaps and resources under the folder "Science" have the same permissions, as set in the folder "Science". A user with *Add, modify, and delete resources* permissions can

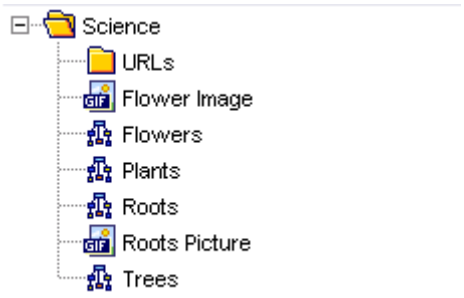

Figure 2: Folder level permissions.

modify any Cmap in the folder or delete any of the resources. Someone with *Annotate resources* permissions can add *Annotations* or *Discussion Threads* to one of the Cmaps, while users with only *View contents of this folder* permission will only be able to open and display the Cmaps and resources. To be able to delete a folder under "Science", the user would need *Modify and delete folders* permissions.

It is important to note that the permissions to perform an action on a folder are determined by the folder containing it. In Figure 2, the permissions set on folder "Science" determine who can delete or rename folder "URLs". The permissions set in the folder "URLs" affect the resources (including folders) it contains.

User IDs and Passwords

In *CmapTools*, permissions are assigned to *User IDs*. When the user first starts the *CmapTools* client program, the software requests that the he or she provide, among other information, a *User ID* and *Password*. These are used by default by the program to identify the user for collaboration purposes. The *User ID* and *Password* are also the "default" identifying pair that the program attempts to use when dealing with permissions, as summarized in the Cmap in Figure 3.

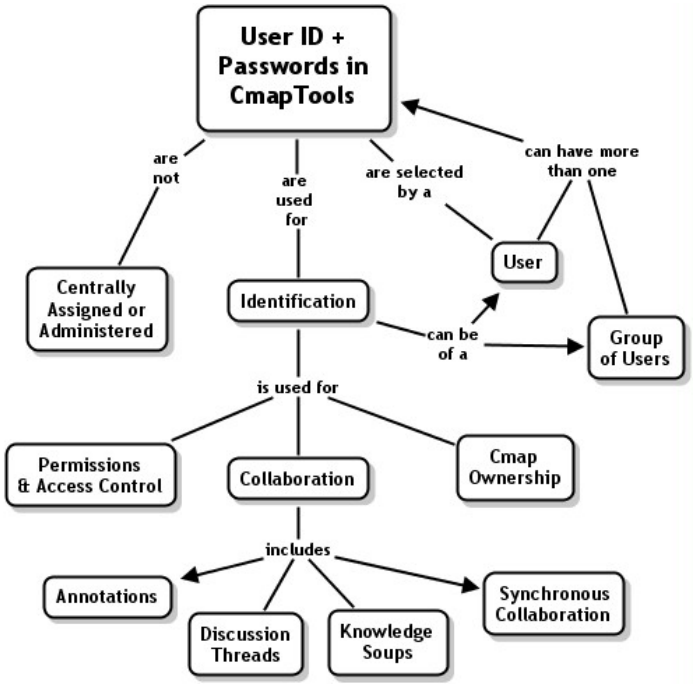

Figure 3: Concept Map on User IDs and Passwords in CmapTools

User IDs are not unique within the *CmapTools* network. There is no central registration system that "assigns" *User IDs* or that guarantees that two or more users do not have the same *User ID*. In fact, when the information is requested, the user can choose any *User ID* and *Password* he or she desires. It is the combination of *User ID* and *Password* that is used as identification, but again there is no guarantee that another person does not have the same *User ID* and *Password* combination. Although this may seem peculiar, the likelihood of somebody else selecting the same *User ID* and *Password* is very low if chosen reasonably, and the security is the same as protecting from somebody "guessing" some user's *User ID* and *Password* to have access to their resources.

A user, however, is not limited to using one *User ID* and *Password* pair. Different combinations of *User IDs* and *Passwords* may be required to access different folders or *Places*. When establishing permissions for a folder, new *User ID* and *Password* pairs may be created as needed on the fly by the person establishing the permissions. For example, a *User ID* and *Password* pair may be established as a mechanism to provide group *Add, modify, and delete resources* permissions to a particular *Place*. All users that are to construct or modify *Knowledge Models* in that *Place* will need to provide that *User ID* and *Password* combination when opening the *Place*. This *User ID* and *Password* effectively becomes a "group" permission for all authorized users. Once a user has provided a *User ID* and *Password* pair when accessing a resource, the *CmapTools* client program will save that pair combination in that computer and will attempt to use it automatically when that user, in that computer, attempts to access other protected resources.

The special *User ID "Everybody"* is used within the permissions scheme to refer to "all" users. So, for example, if a folder has *Add, modify, and delete resources* permissions set to *Everybody*, then any user can create maps, add resources, change the resource's names, or delete resources in that folder.

Types of Permissions

CmapTools manages six types of permissions, as described below. These permissions are not mutually exclusive, and can be granted in combinations as needed for different users.

Administrator

View contents of this folder

Annotate resources

Add, modify, and delete resources

Add folders

Modify and delete folders

Setting Permissions

On a selected folder (or *Place*), the Edit/Properties menu entry provides access to the Folder's permissions. Figure 4(a) shows the *Properties* window for the "Science" folder in Figure 2. At the bottom left of the window, the *Permissions* button opens the *Permissions* window in Figure 4 (b). This window displays each *User ID* and the corresponding permissions that have been assigned to that *User ID*. You need to have *Administrator* permissions on a folder to be allowed to display the dialogue box in Figure 4 (b) for that folder. To grant a permissions to another user (or to "create" a *User ID*), the *Add User...* button is clicked in the *Permissions* window, displaying the dialogue box in Figure 5. Selecting one of the *User IDs* in Figure 4 (b) and clicking the *Edit…* button would open a similar window to change the permissions for that user.

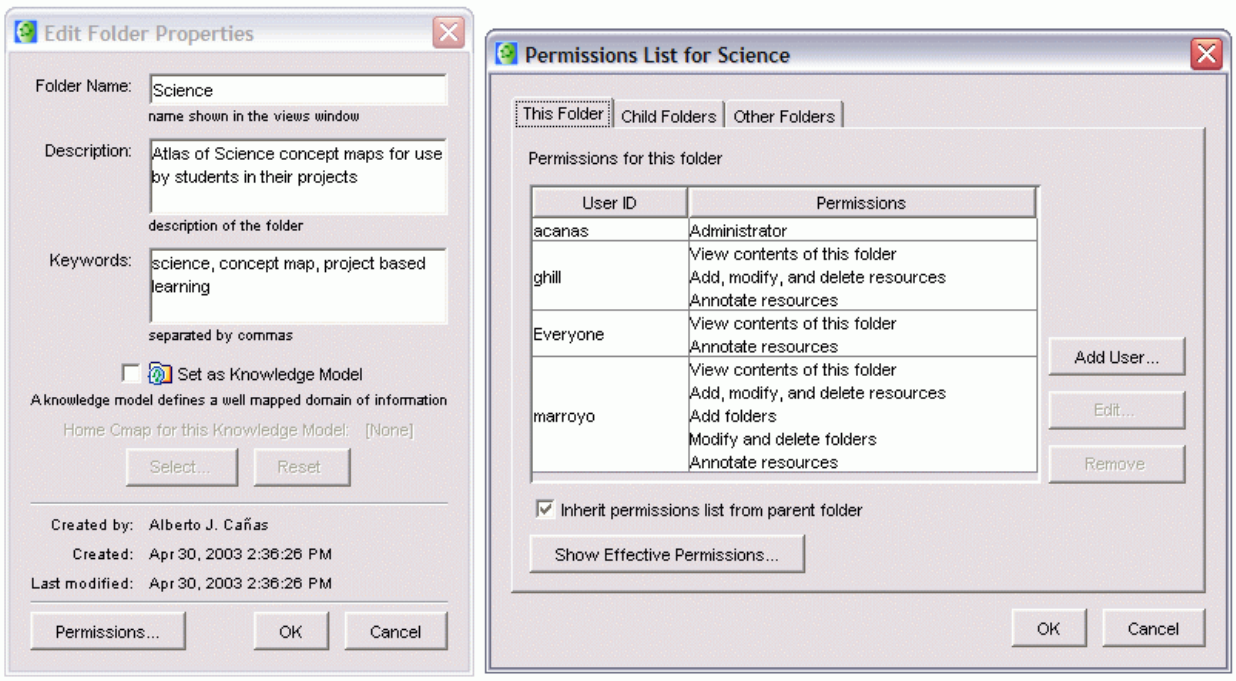

Figure 4: (a) The Properties window of a folder or Place shows the Permissions button
(b) The Permissions lists shows the User IDs and permissions set for the folder

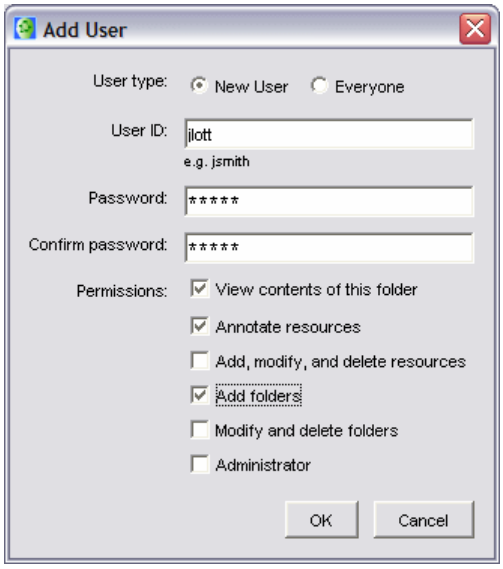

Figure 5: Dialogue box to add a User ID and corresponding permissions to a folder

In the dialogue box in Figure 5, the Administrator can assign the various permissions to the corresponding *User ID* and *Password* pair, or to *Everyone.*

Administrator Permission

A user who has *Administrator* permission over a folder has not only all the other permissions on that folder, but is also allowed to grant or deny permissions on that folder. That is, the user can give or take away other users' permissions on the folder. There always has to be at least one user with *Administrator* permissions for each folder, and there can be more than one user with *Administrator* permissions on a folder.

View contents of this folder Permission

The *View contents of this folder* permission allows the user to list the contents of the folder (in the *Views*) and display the contents of any of the resources (images, Cmaps, etc.). This permission does not grant the user authorization to create, modify, annotate, or delete Cmaps, in this folder, nor does it authorize the user to add or delete other resources (e.g. images, videos, URLs) to it.

Annotate resources Permission

Annotate resources allows a user to add *Annotations* (post-it type notes) to a Cmap without the need to have *Add, modify, and delete resources* permissions, and to create a *Discussion Threads* (*DT*) on Cmaps or participate in *DTs* already created in Cmaps in that folder. This permission is aimed at allowing collaboration, critique, and peer-review without allowing others to modify one's concept maps.

Add, modify, and delete resources Permission

To create, modify, rename or delete a Cmap in a folder, or to add, rename or delete any type of resource in folder, the user needs *Add, modify, and delete resources* permission on that folder. Having this permission also gives the user *Annotate resources* permission.

Add folders Permission

The *Add folders* permission allows the user to create new folders within the folder on which the permissions are granted. In general (the defaults can be changed as explained below), the user who creates a folder becomes the *Administrator* (and thereby given *Administrator* permissions) of the new folder.

Although it may seem strange to have an *Add folders* permission separate from the *Add, modify, and delete resources* and *Modify and delete folders* permissions, there are situations where one wants to grant users the permission to create their own folder where each user can start constructing Cmaps, but not grant permissions to create those Cmaps in the current folder or to delete somebody else's folders (e.g. in a classroom environment). This type of scenarios are discussed below.

Modify and delete folders Permission

The last permission, *Modify and delete folders*, allows the user to delete or rename folders within the folder on which the permission is granted. Having *Modify and delete folders* permissions does not authorize the user to grant other users permissions on the folder – that is only allowed with *Administrator* permission.

Permissions and My Cmaps

Permissions have no effect on resources and folders in *My Cmaps*. Since *My Cmaps* is not shared, it is assumed that the user has complete permissions over all the resources and folders.

Effective Permissions and Inheritance of Permissions

The scheme described above for assigning permissions can become cumbersome to manage in an environment where folders are constantly being created, which may imply setting permissions for each one of them.

Consider Figure 6. Suppose you create, as the project manager, the *Knowledge Model* folder "Knowledge Capture Project". You set a group *User ID* and *Password* on the "Knowledge Capture Project" folder allowing all users to *View Contents of this Folder*. You now create folders Expert 1, Expert 2, Expert 3 and Expert 4, and to each you assign *Add, modify, and delete resources, Add folders, and Modify and delete folders* permissions to the respective members of the teams that are going to work on each of those folders.

Now assume that one of the members of the Expert 1 team creates a folder "Images", as seen in Figure 6. How can he or she restrict the permissions so that only the members of the team get *Add, modify, and delete resources, Add folders,* and *Modify and delete folders* permissions when he/she may not even know the *User ID* and *Passwords* of the members of the team, let alone the hassle of having to assign the permissions to each user every time a new folder is created under Expert 1? To deal with these issues, the *CmapTools* permissions scheme allows for the *Inheritance* of effective permissions.

Effective Permissions

Figure 6: Knowledge model folder

Permissions to perform an action, whether it is to create a Cmap, add a folder, drag and drop an image to *Views*, copy a *Knowledge Model*, delete a resource, or any similar action, are checked at the moment the operation is executed. At that point, the *CmapTools* client (and the *CmapServer*, if the operation involves any of the *Places*) checks the effective permissions to determine if the user is allowed to perform the operation.

Inheritance of Permissions

A folder can be set to "inherit" the effective permissions from its parent folder. If the "inheritance" is set, then, when checking for permissions, if the authorization needed is not found in the folder, the program will check for the permissions in the parent folder. This will continue until the needed permissions are found, the "inheritance" is not set in one of the folders in the path to the root, or after the "root folder's" permissions are checked. The dialogue box in Figure 4(b) shows the *Inherit Permissions list from the parent folder* checkbox, which allows the *Administrator* of a folder to set the *Inheritance* on.

In Figure 6, for example, the folder "Images" may be set to inherit the effective permissions from its parent folder, "Expert 1". When a user tries to perform an action in the "Images" folder (e.g. drag and drop an image file into it), the program will check the user's permissions in folder "Images" to determine whether he/she has authorization to *Add, modify, and delete resources*. If the permission has been granted (either explicitly or because the user has *Administrator* permissions on "Images"), the permission checking stops and the action is performed. If the user does not have the required authorization on "Images", but "Images" is set to inherit the effective permissions from its parent, the program will check the folder "Expert 1" for the required permissions. If the permissions needed are found in "Expert 1", the checking stops. Otherwise, the program checks whether the "inheritance" is set in "Expert 1". If it is not, since the permissions were not found the action is denied[2]. If the inheritance is set, the checking continues up the hierarchy. This way permissions granted in "Expert 1" are effective down the hierarchy, as long as the "inheritance" option is set. By setting up the appropriate permissions for each folder ("Expert 1", "Expert 2", "Expert 3", and "Expert 4)" and making sure that the folder underneath each of them has the "inheritance" flag on, the project manager needs to set the permissions only once.

[2] Actually, the program will continue to check whether the user has *Administrator* permissions all the way to the root folder. *Administrator* permissions are automatically inherited, as described in the next section.

Inheritance of Administrator Permission

If you create a folder, and allow others to create folders underneath yours, you want to be able to delete any of those folders or their contents independent of the permissions the folders have. For example, the *Administrator* of the "root folder" of a *Place* should have permissions to delete any resource or folder in the *Place*.

To achieve this, the inheritance of *Administrator* permissions is independent of whether the *Inheritance* is set on or off in the folder. If the user does not have the required permissions to perform an action given the permissions set in the folder, the program will check through the hierarchy of folders according to the inheritance mechanism described above, and if necessary will continue up to the "root folder" checking if the user has *Administrator* permissions in any of the folders up to the root.

Default Permissions

The example in Figure 6 assumed that folders created under the "Expert 1" folder have the "inheritance" flag set. In some other cases, you may not want the "inheritance" flag set by default. Consider the following situation:

A "Public Cmaps" *Place* wants to be established where any user can create folders that they own and where they can construct their own *Knowledge Models*. To allow users to create their own folders, the "root folder" for the Place must have *Add folders* permissions for *Everybody*. Figure 7 shows the result of four users creating their own folder.

However, if the folders "User 1", "User 2", "User 3", and "User 4" have the "inheritance" on, then any user could create new folders under them. This may not be desirable. We may want each of these folders to have the "inheritance" off, and that *Everybody* have only *View contents of this folder* and *Annotate resources* permissions.

Figure 7: Folders from various users at the same hierarchical level

As the users start creating their new folders, as shown in Figure 8, “Images” and “JPGs” should probably inherit the permissions from “User 1”. If folders are created under “User 2”, they most likely should inherit the permissions from folder “User 2”.

To handle these types of conditions, the *CmapTools* permissions scheme allows the definition of two *Default Permission Sets* for each folder: *Default child folder* under this folder permissions

Figure 8: Folders within the folder of one of the users from Figure 5

and *Default Other Folders* under these folder permissions. These work in combination with the permissions for the particular folder for which they are set.

These default permissions are particularly relevant for the “root folder” of a *Place*. We first explain how they work in terms of the “root folder” of a *Place* and then explain their general functionality for other folders.

Default permissions for the “Root Folder” of a Place

By default, the “root folder” of a *Place* is set after installation with the following set of permissions, also shown in Figure 9:

<CmapServer Administrator>	*Administrator*
Everyone	*View Contents of this Folder*
Everyone	*Add folders*
Inherit: No (does not apply)	

This set of permissions allows anybody to create their own folders and view the contents of others’ folders, but does not allow them to delete somebody else’s folder. In addition, it prevents users from adding resources at the top level of a *Place*. The *Administrator* of the “root folder” (by default the same user as the *Place*/*CmapServer’s Administrator*) can change these permissions at any time.

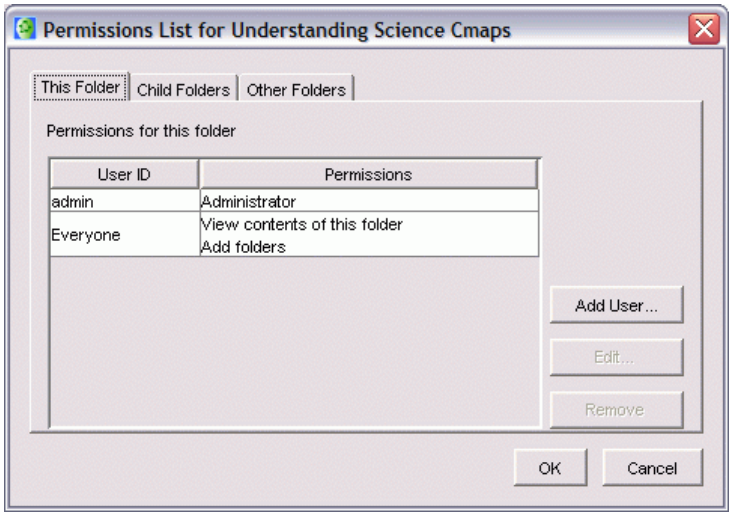

Figure 9: Default permissions for the "Root Folder" of a *Place*

Default Child Folders Permissions of the Root Folder

The administrator of a *Place* can determine what permissions are set by default when a folder is created at the "top level" of the *Place's* folder hierarchy, right under the *Place's* icon. For permissions purposes, when setting the permissions for a particular folder, any folder created immediately under it is referred to as a *Child Folder*. All folders underneath a *Child Folder* falls under the term *Other Folders*.

The "root folder" – as all folders – has default permissions that are assigned to its *Child Folders.* The Administrator of the "root folder" can change these permissions. Any folder that is created at this "top level" will have these permissions set by default. Unless the *Place's* administrator changes them, the *Default Child Folders* permissions for the "root folder" are set during installation as shown in Figure 10.

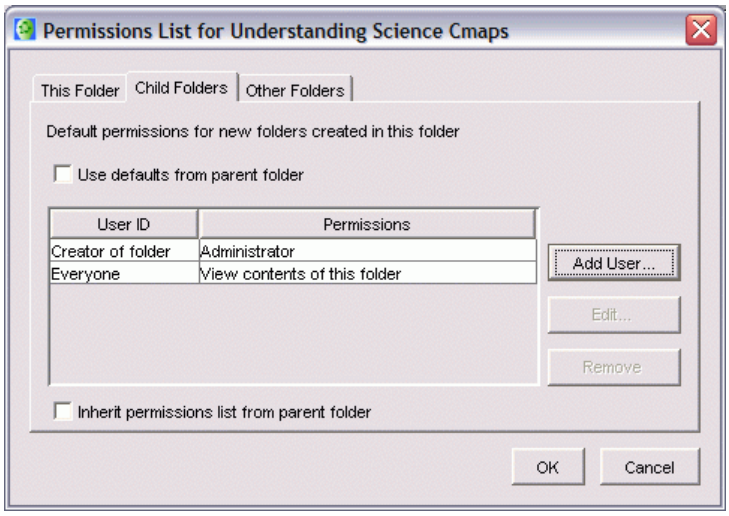

Figure 10: Default *Child Folder* permissions for the "Root Folder"

The permissions make sure that only the creator (*Administrator*) of the folder can create resources or add new folders to the newly created folder. By setting the *inheritance* off, the permissions of the "root folder" are not inherited into this folder.

Default Other Folders Permissions of the Root Folder

The administrator of a *Place* can also define the default permissions for folders other than the "top level" Child Folders. Any folder created within one of the "top level" folders will have these permissions set by default, but they can be changed at any time (even during folder creation) by the folder *Administrator*.

Unless the *Places's Administrator* changes them, the *Default New Folder Permissions* are set as shown in Figure 11.

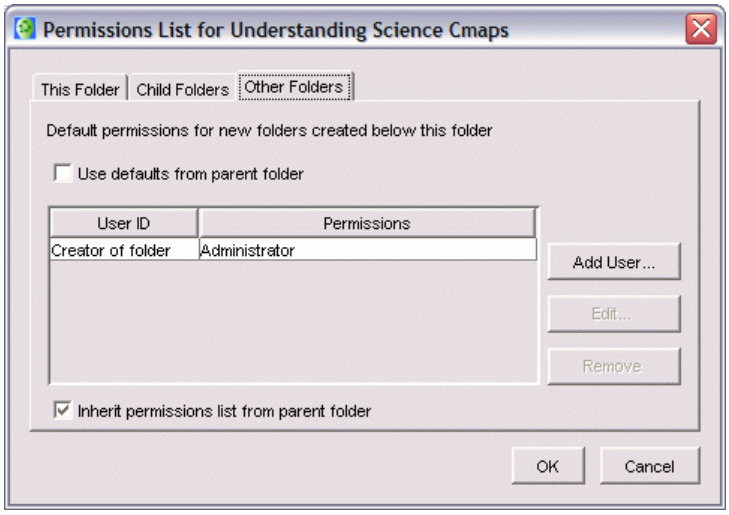

Figure 11: Default *Other Folders* permissions for the "Root Folder" of a *Place*

These default permissions set the creator of the folder as its *Administrator* ands inherits the permissions from the parent folder. If the defaults are used – that is, if none of the permissions on the folders along the path to the "top level" folder are changed -- this means the permissions are inherited from the "top level" folder.

Back to Figure 8, if "User 1", "User 2", "User 3" and "User 4" are top-level folders of a *Place*, and the default permissions are set as described above, then the owner/creator of each of these folders would be its *Administrator* and *Everyone* would have *View contents of this folder* permission. The "Images" folder would, by default, have it's owner as *Administrator* and inherit permissions from "User 1". Likewise, "JPGs" would have its owner as *Administrator* and inherit permissions from "Images" (and thereby from "User 1").

Default Child Folders and Other Folders Permissions – Generalization

The default *Child Folders* and default *Other Folders* permissions can be set for any folder, not only the "root folder" of a *Place*. Consider the example in Figure 12. The "Students" folder contains folders for two classes: "Math 5" and "Science 7". Within "Math 5", we see individual folders for "Student 1" and "Student 2", and within "Science 7" folder for groups: "Group 1" and "Group 2". The "Teachers" folder will have folders for the various teachers. Most likely, the default permissions needed for the folders under "Math 5" (for individual students) and for

folders under “Science 7” (for groups of students) will be different, and definitely different from the permissions under the “Teachers” folder.

To support these differences, *CmapTools* allows that, for any folder, the default *Child Folder* permissions and default *Other Folder* permissions be set by the *Administrator*. Consider the “Students” folder: if the default *Child Folders* permissions is set, this will determine the default permissions for “Math 5”, “Science 7” and other class’s folders that are created. The folders under “Math 5” (“Student 1” and “Student 2”) and under “Science 7” (“Group 1” and “Group 2” would have as default the permissions established by *Other Folders* in the “Students” folder, unless the permissions for *Child Folders* are set for “Math 5”or “Science 7” respectively.

To clarify how this scheme works, let us assume a student creates a new folder under “Student 1” in Figure 12. To determine what its default permissions will be, the program looks for the *Child Folders* permissions in the “Student 1” folder. If this default has been set, then it defines the default permissions. If is not set, then the system starts going up the hierarchy of folders (“Math 5”, then “Students”, up to the “root folder”) looking for the first set of *Other Folders* permissions. Since the “root folder” always has default permission set, then it is guaranteed that a default set will be found.

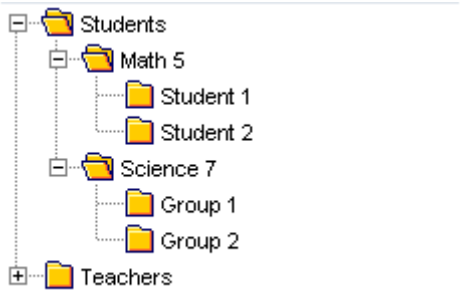

Figure 12: Default permissions for top level and other folders

This way, by establishing the *Child Folders* and/or *Other Folders* permissions in the “Students”, “Math 5”, “Science 7” and “Teachers” folders, we can establish different collaboration and access schemes depending on how the students, groups and teachers will be working. In the Scenarios section below, we consider cases like this in detail.

Additional Rule

Consider the set of folders in Figure 13. Assume that the "Team Project" folder has permissions:

Everyone	*Add folders*
Everyone	*View Contents of this folder*

You create your own folder "Planets". This give you *Administrator* permissions on "Planets" and you can set the permissions for resources, Cmaps and folders within "Planets". You can also determine whether "Planets" *inherits* permissions from "Team Project" or not.

Suppose you want to delete or rename the folder "Planets". The permissions to rename or delete the folder are determined by the "Team Project" folder, where you do not have *Modify and delete folders* permission. Therefore, even though you are *Administrator* for the folder, you cannot modify or delete it.

The following rule is followed with respect to *Administrator* permissions on a folder:

> *The Administrator of a folder may delete or modify the folder (e.g. rename) even if he/she does not have Modify and delete folders permissions in the containing folder.*

Given this rule, as *Administrator* you are able to rename or delete the "Planets" folder.

Figure 13. Modifying or deleting folders you own.

Displaying the Effective Permissions

The *Permissions* button in a folder's *Properties* dialogue box displays a window that shows the permissions set for a particular folder. However, as was explained above, the permissions are checked when a resource is being accessed and the *inheritance* plays a key role in determining

whether a user has the required permissions. Thus, the permissions displayed in the *Permissions List* window shown, for example, in Figure 4(b) may not include all users who have access to resources in that particular folder, as some permissions may be inherited from ancestor folders along the path to the "root folder".

To determine who can perform operations on resources in a folder, the *Effective Permissions* button shown in Figure 4(b) will open a window that displays all users that have any type of permissions on the folder, with the corresponding permissions.

Permissions and Copy, Publish & Move

When you *Move*, *Publish* or *Copy* folders within a *Place* or between *Places*, the permissions are affected. We describe the *Copy* and *Publish* together, since they have similar behaviors, and the *Move* separately:

Copy and Publish

If the *Copy* involves only resources (no folders) then the permissions on the resources are determined by the target folder. To *Copy* a resource, the user needs to have *View contents of this folder* permissions on the folder containing the source resource, and *Add, modify, and delete resources* permissions on the destination folder.

If the *Copy* involves one or more folders, the action can be considered -- for permissions purposes -- as if the user who is performing the *Copy* was creating, at the destination, each folder being copied. The user performing the *Copy* will be the *Administrator* of each of the new folders, and the default permissions for folder creation will establish the permissions for each folder, unless the user changes them. Thus, permissions in the source folders are not "transferred" to the destination during the *Copy*. The source folders and their permissions remain unchanged.

Publish behaves the same as a folder *Copy.*

To *Copy* or *Publish* a folder, the user needs to have *View contents of this folder* permissions on the folder containing the source folders, and *Add folders* permission on the destination folder.

Move

Resources moved to another folder will have their permissions determined by the target folder. To *Move* a resource, the user needs to have *Add, modify, and delete resources* permissions at the source and target folders.

Moving a folder carries the folder permissions with it. The user needs to have *Modify and delete folders* permissions on the folder containing the folder to be moved and *Add folders* permissions at the destination folder. The permissions of the folder moved are unchanged.

Scenarios

The following scenarios show how to configure the permissions schemes for different environments.

Protected Team Place

A *Place* is to be installed for a team project, where only members of the team should be able to create Cmaps, but these should be available for everybody to browse. Each member of the team will be working on his/her own separate folder for which he/she is responsible, but other members of the team should be able to modify and annotate their Cmaps.

The following set of permissions would handle this situation:

"Root folder" permissions:

> A specific *User ID* and *Password* can be created to allow access at the "root folder" that is shared by all the members of the team (for illustration, let us use *User ID*: TeamID, *Password*: TeamPasswd)
>
> Figure 14 shows the corresponding permissions. The *Add, modify, and delete resources* permission was omitted to avoid having Cmaps and resources stored at the "top level".

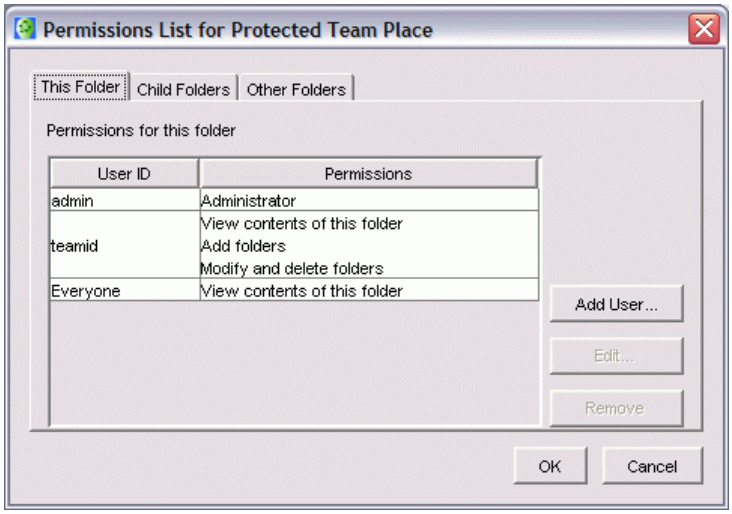

Figure 14: "Root folder" permissions for the "Protected Team Place"

Default Child Folders permissions for the "root folder"

Since each member of the team is to be working on a separate folder for which he or she is responsible, but other members of the team should be able to modify and annotate their Cmaps, the permissions could be:

Figure 15 shows the default permissions for *Child Folders*.

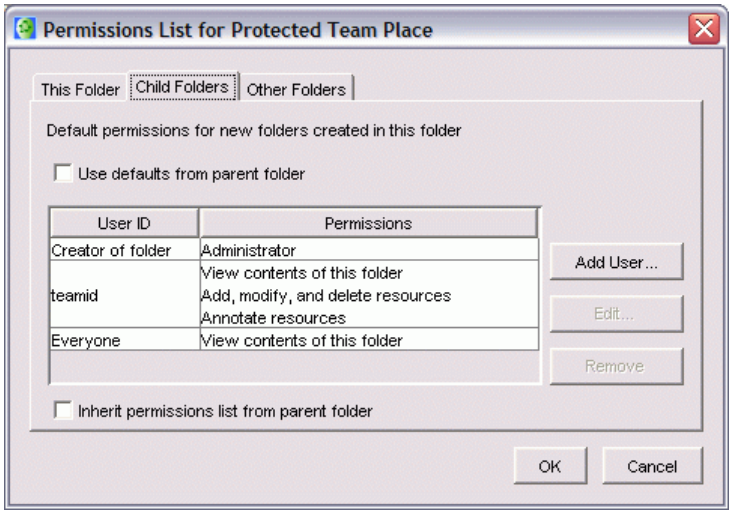

Figure 15: Child Folders permissions for the "Protected Team Place"

If we change the conditions so that members of the team should only be able to annotate/criticize – not modify – each other's Cmaps and resources, the permissions could be as shown in Figure 16.

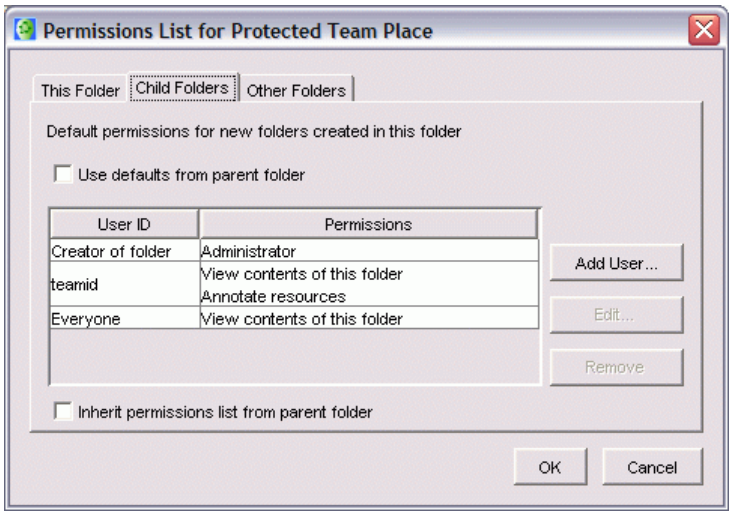

Figure 16: Child Folders permissions for the "Protected Team Place" with only *Annotate* permissions for the members of the team

Default Other Folders permissions for the "root folder"

The *Default new folder* permissions in any of the two cases could be set as shown in Figure 17.

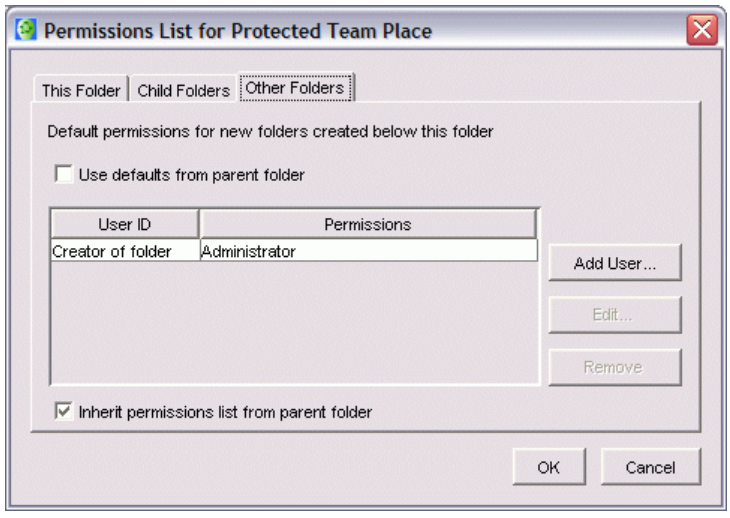

Figure 17: Other Folders permissions for the "Protected Team Place"

Shared Place

Consider a *Place* where a large group of users construct their *Knowledge Models*. They are interested in sharing these with the community-at-large, and welcome comments and annotations on their Cmaps from anybody, but want to protect their *Knowledge Models* from modification or deletion by those not authorized.

Within the group, there are teams working on various projects, and each team will have their own folder under which they will build their *Knowledge Models*.

The following set of permissions would handle this scenario:

"Root Folder" permissions:

A specific *User ID* and *Password* can be created for access at the "root folder" that is shared by all the members of the group (for illustration, let us use *User ID*: GroupID, *Password*: GroupPasswd). Figure 18 shows the corresponding permissions.

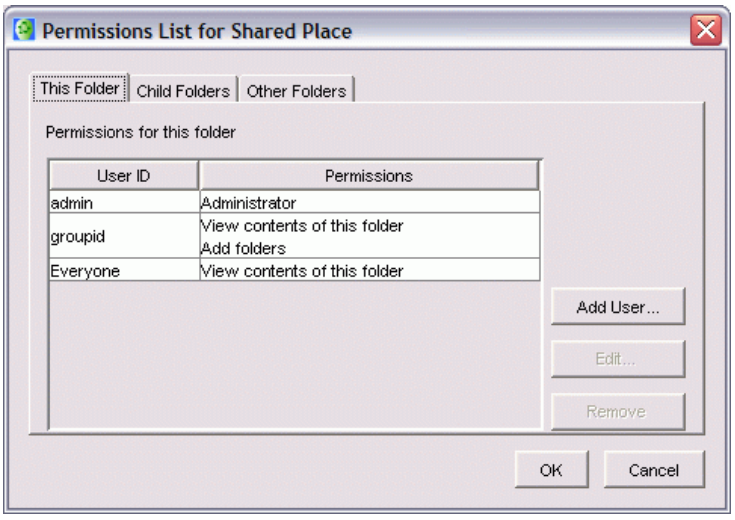

Figure 18: "Root folder" permissions for the "Shared Place"

Default Child Folders permissions

Each team will be creating its own Folder, and should use a TeamID that allows all members access to this folder:

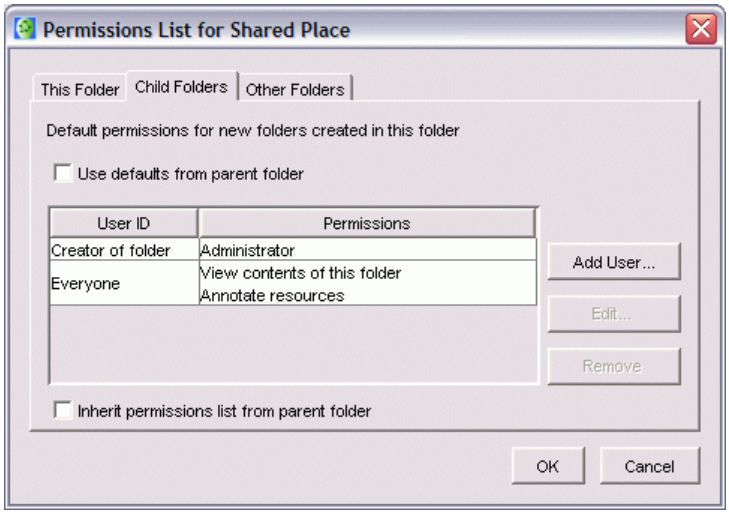

Figure 19: Child Folders permissions for the "Shared Place"

The member of the team that creates the Team's folder should then add the following permission (which cannot be added to the default because each team has its own unique *User ID*):

TeamID	*Add folders*
TeamID	*Add, modify, and delete resources*
TeamID	*Modify and delete folders*

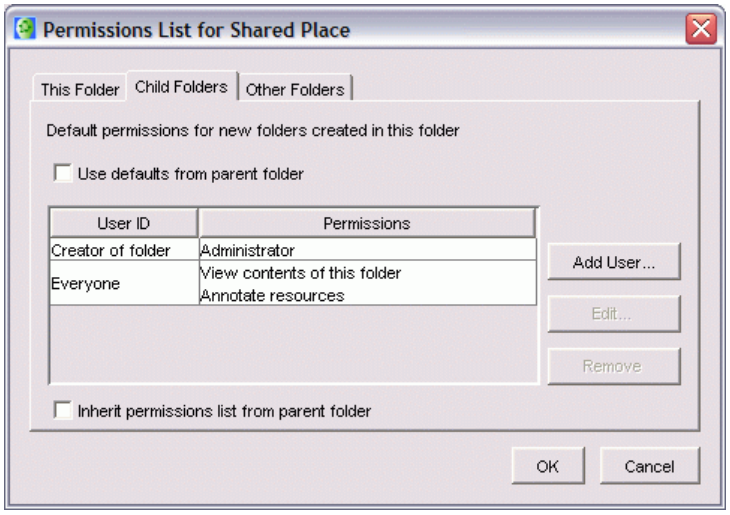

Figure 20: Other Folders permissions for the "Shared Place"

Other Folders permissions

The *Other Folders* permissions could be set as displayed in Figure 20.

Public Place

The *CmapTools* network has always included various servers that are open for anybody to share their *Knowledge Models*. These servers must allow anybody to create folders at the "top level" and at the same time make sure that each user's *Knowledge Models* are protected from being deleted by others. Lets assume we want allow *Annotations* by other users by default.

The following combination of permissions allows for this scenario:

"Root Folder" permissions:

Everybody is allowed to create new folders at the "top level" in this *Place*, as shown in Figure 21.

Figure 21: "Root Folder" permissions for the *Public Place*

Default Child Folders permissions

Each user creates his/her own folder and becomes the *Administrator*. Others are allowed to comment, annotate, and criticize the Cmaps, as shown in Figure 22.

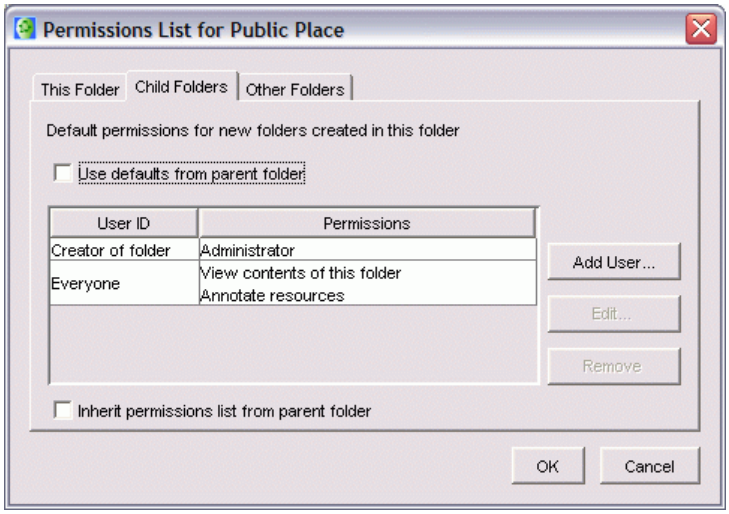

Figure 22: Child Folders permissions for the *Public Place*

Default Other Folders permissions

The default *Other Folders* permissions could be as shown in Figure 23

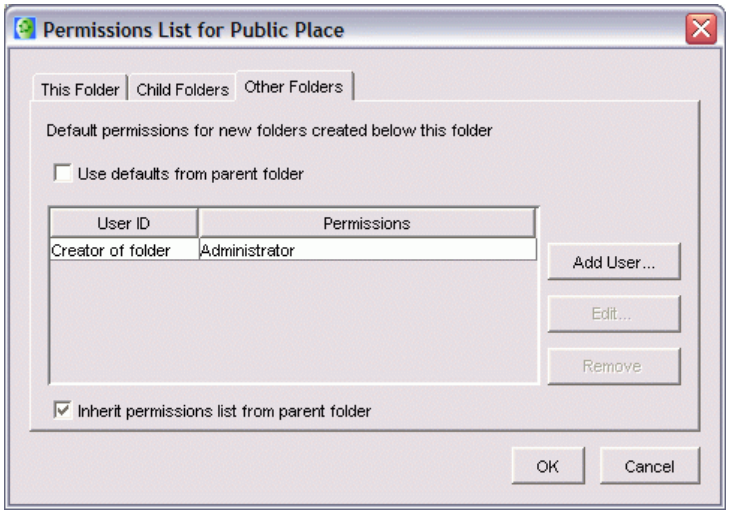

Figure 23: Other Folders permissions for the *Public Place*

A Place in a School Classroom

A *Place* (*CmapServer*) has been installed in a classroom. Only the teacher is allowed to create folders at the top-level, which he/she does for individual students, teams, groups, etc. as necessary. By default, students are allowed to create folders within each top-level folder, and are the owners of those folders. Other students, by default, can only *Annotate* each other's Cmaps. The *Place* is available on the Internet for collaboration with other schools, but by default, students from other schools can only *Annotate* the Cmaps and resources.

The following set of permissions would handle this scenario:

"Root Folder" permissions:

A specific *User ID* and *Password* can be created for the teacher (for illustration, let us use *User ID*: TeacherID, *Password*: TeacherPasswd), and another general *User ID* and *Password* to allow students to create their folders (lets use StudentID as the *User ID*). In addition, each student has a *User ID* and *Password* within the LAN and that was provided to *CmapTools* the first time the program was executed.

Figure 24 shows this set of permissions.

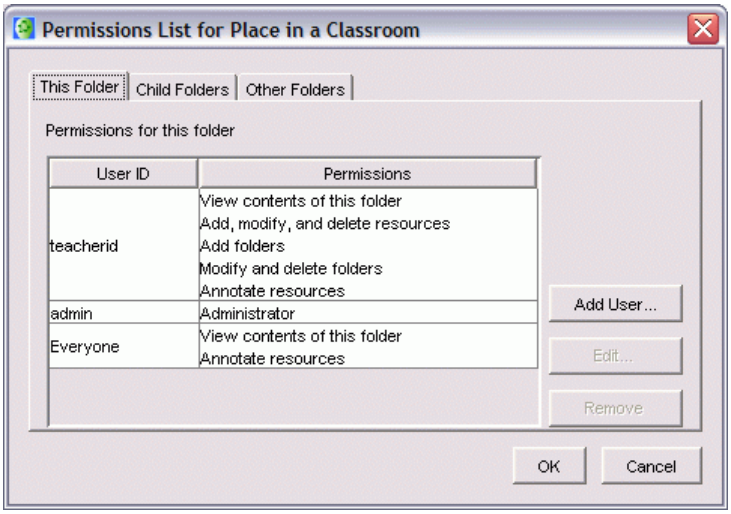

Figure 24: Root Folder permissions for the *Classroom Place*

Default Child Folders permissions

Only the teacher can create folders at the top-level. The teacher can afterwards change permissions as necessary. Observe in the permissions in Figure 25 that the *Inheritance* flag is set to on.

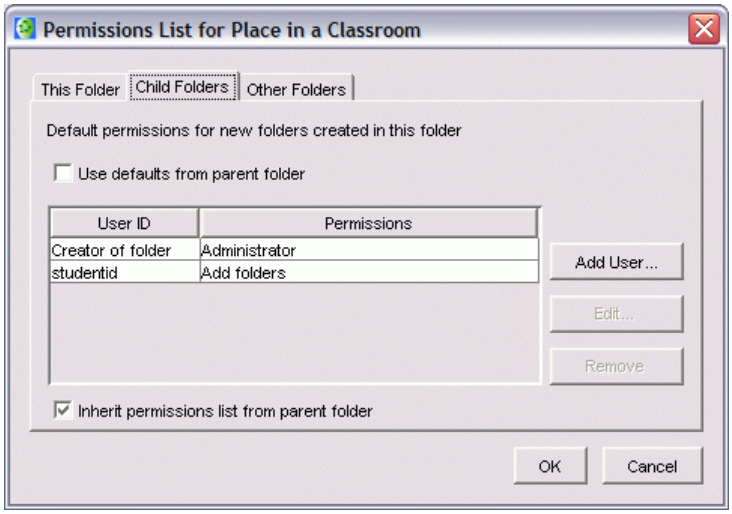

Figure 25: Default *Child Folders* permissions for the *Classroom Place*

Default Other Folders permissions

Any student can create a folder under the top level folder, and becomes its owner. Others can only *Annotate resources* under it. The *Default new folder* permissions could be as shown in Figure 26. Note that the *inheritance* flag is off.

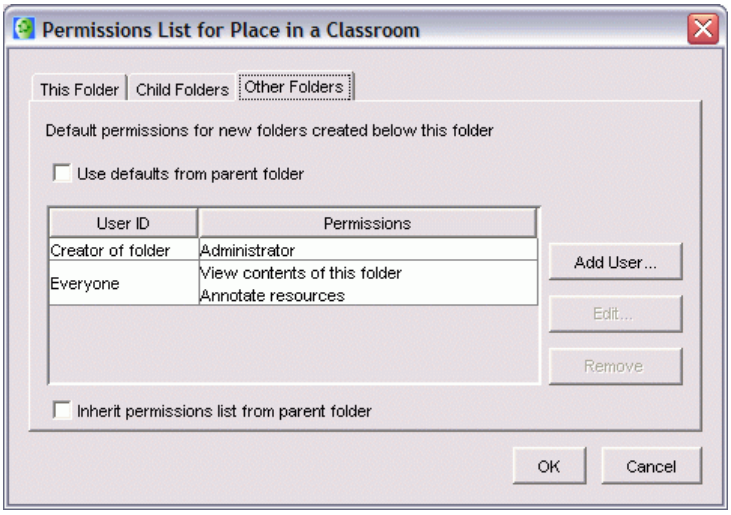

Figure 26: Default *Other Folders* permissions for the *Classroom Place*

Extending this scenario, let us assume that the class is working on a project where there are several groups of students. Each group of students will be competing with the other group, and so they should not be allowed to view each other's work. Within each group, since all students should be able to create their own folders and have all permissions on all folders, Cmaps and resources of other members of the group, the teacher will assign each group a separate *User ID*. The teacher wants to set up this environment without having to modify the whole structure of the *Classroom Place* described above.

The teacher can create a folder "Project" at the top level of the file structure and set the following permissions:

"Project Folder" permissions:

A specific *User ID* and *Password* can be created for the project (for illustration, let us use *User ID*: ProjectID). Students in the project will be able to create their own folders under "Project". Alternatively, the *User ID* for each of the groups could have been listed in the permissions. The permissions are shown in Figure 27. Note that the *inheritance* is off.

To define the permissions on folders created under "Project", the teacher redefines the *Child Folders* permissions for folder "Project" as follows:

Child Folders permissions

Any group can create a folder under "Project", but cannot see the resources or Cmaps

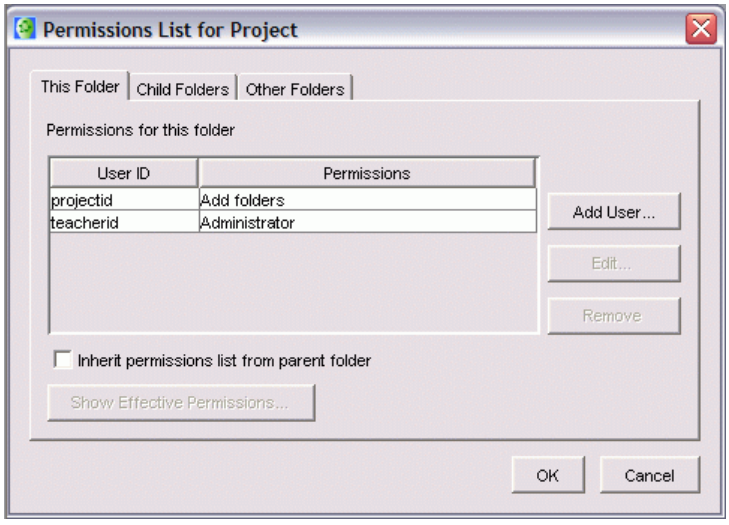

Figure 27: Permissions for the "Project Folder"

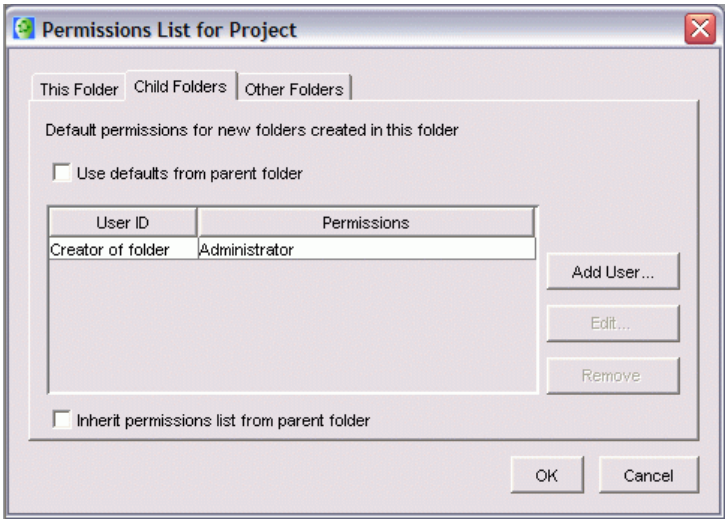

Figure 28: Default *Child Folders* permissions for the "Project Folder"

under folders created by others. Figure 28 shows these permissions, with the *inheritance* flag off.

Other Folders permissions

Below the child level (the child level under "Project", that is), the permissions are as shown in Figure 29. In this case, whether Inheritance is on or off does not make much of a difference except if the group for some reason decided to change permissions in one of the folders.

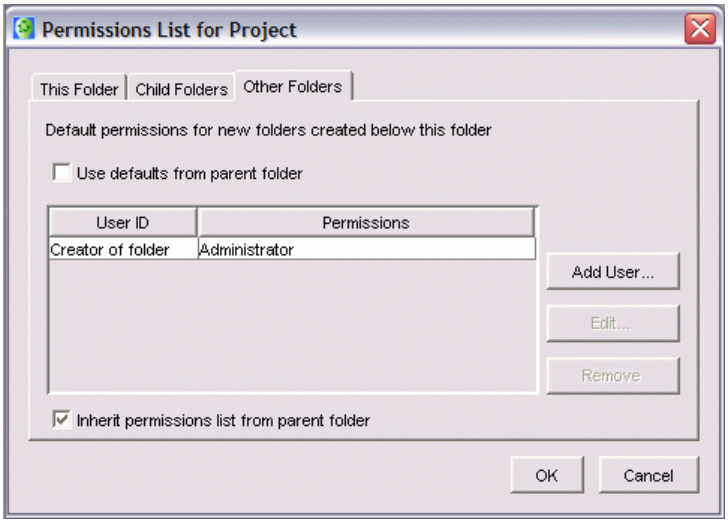

Figure 29: Default *Other Folders* permissions for the "Project Folder"

A Place in a School

Consider the situation of a *Place* (*CmapServer*) being shared by students and teachers in a school. Initially the *CmapServer Administrator* may have created two folders, one for "Teachers", and one for "Students". Within the "Teachers" folders, teachers are allowed to create their own folders and collaborate among them. Within the "Students" folder, teachers are allowed to create a folder for their class. The folder structure would be similar to that of Figure 12. To simplify permissions, lets assume a TeacherID and a StudentID *User ID* exist, which allow for general identification of teachers and students.

"Root Folder" permissions:

At the "root folder", only the *CmapServer Administrator* has permissions to create new folders, as shown in Figure 30.

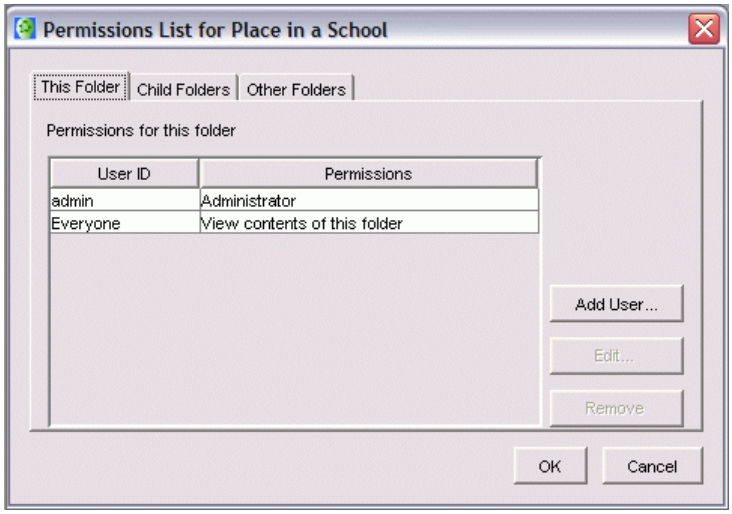

Figure 30: "Root Folder" permissions for the *School Place*

Child Folders permissions

Only the *CmapServer Administrator* can create folders at the top-level, as displayed in Figure 31.

Other Folders permissions

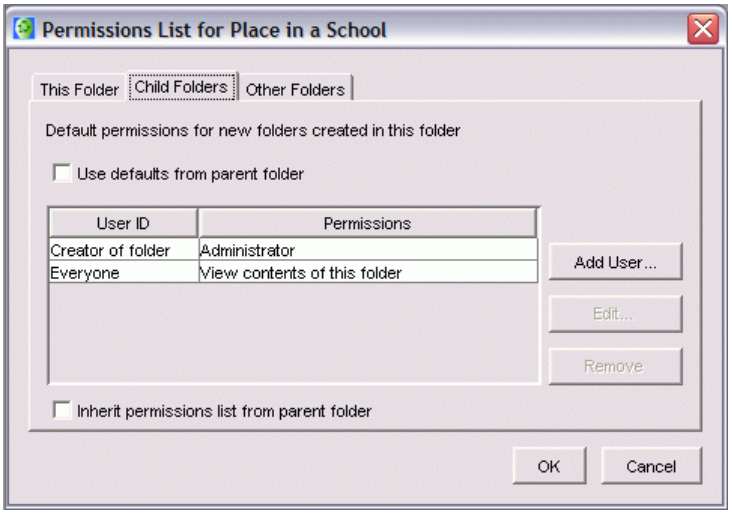

Figure 31: Child Folders permissions for the *School Place*

In this case, it is better to change the *Default Permissions* for the "Teachers" and the "Students" folders, so these permissions will be irrelevant. The *Default new folder* permissions for the "root folder" can be the same as the *Child Folders'* permissions in Figure 31.

The "Teachers" folder will have its permissions, and its *Default Permissions* changed to the following:

"Teachers" folder permissions:

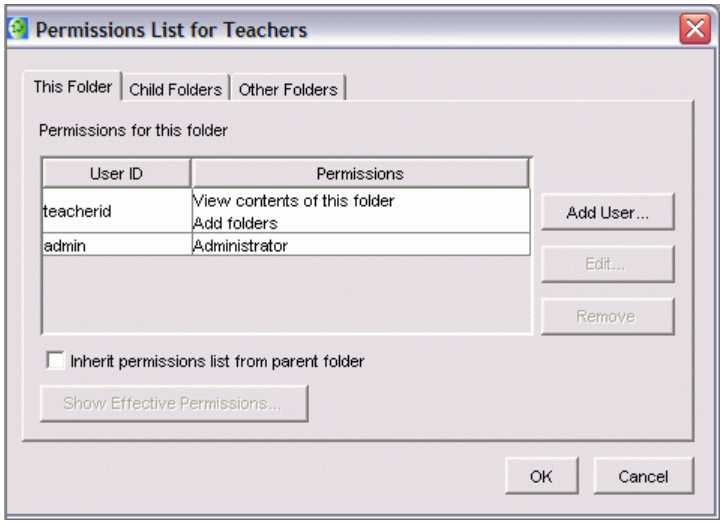

Figure 32: Permissions for the "Teachers" folder

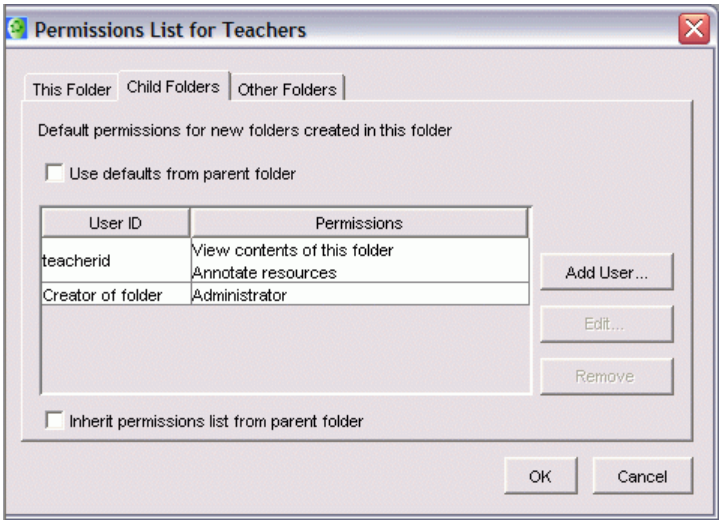

Figure 33: Child Folders permissions for the "Teachers" folder

At the "Teachers" folder, only Teachers are allowed to create folders. Each is owner of his/her folder. Other Teachers are allowed to collaborate and so can *Annotate* each other's Cmaps. Students should not have access to the Teacher's Cmaps. These permissions are shown in Figure 32.

Default Child Folders permissions for "Teachers" folder

The permissions for the *Child Folders* are shown in Figure 33.

Default new folder permissions for "Teachers" folder

The *Default new folder* permissions for "Teachers is simple, as displayed in Figure 34.

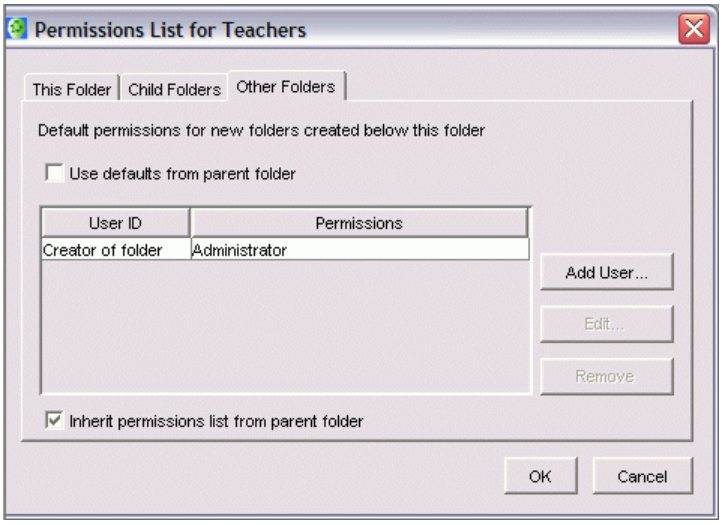

Figure 34: Other Folders permissions for the "Teachers" folder

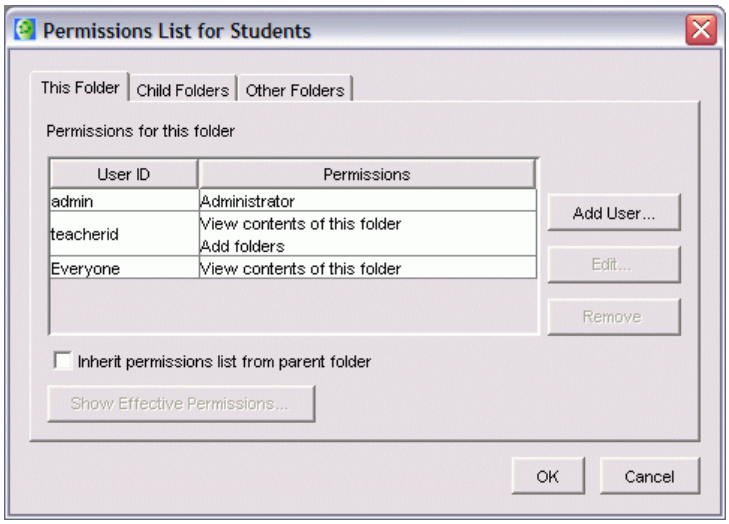

Figure 35: Permissions for the "Students" folder

Creation of folders in the "Students" folder is a two step process. The Teacher must create a folder for the Class, and then determine the permissions according to the way the class work is to be organized. The permissions for the "Students" folder could be as follows:

"Students" folder permissions:

At the top level of the "Students" folder, only Teachers are allowed to create folders. Each is owner of his/her folder. Figure 35 shows this set of permissions.

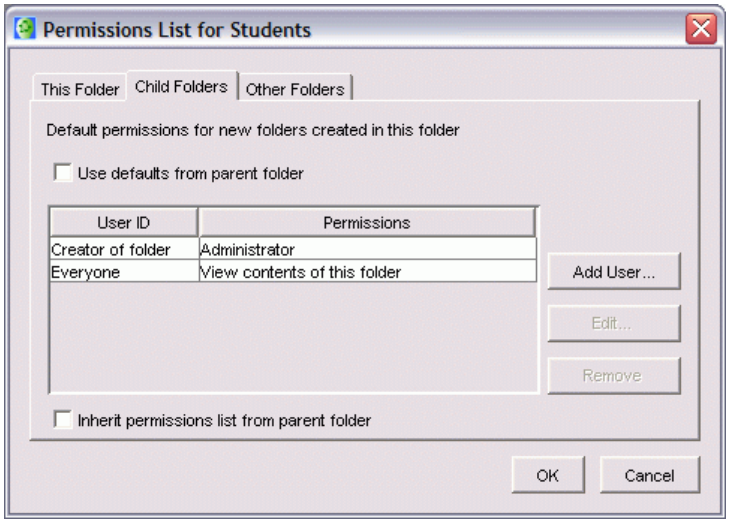

Figure 36: Child Folders permissions for the "Students" folder

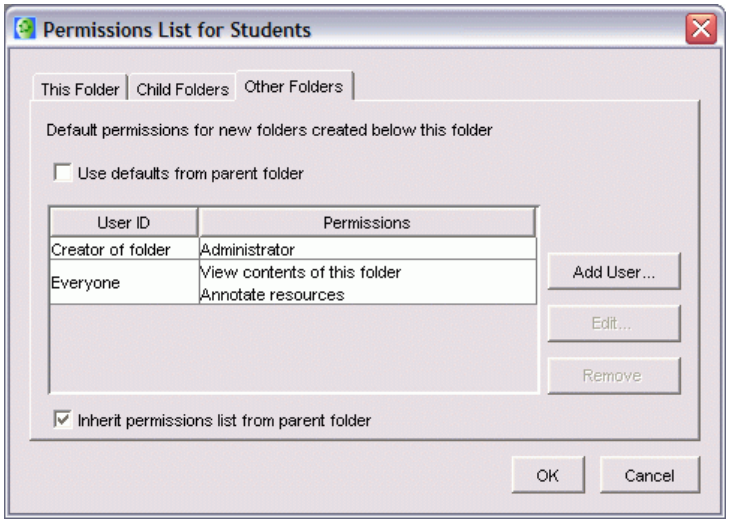

Figure 37: Other Folders permissions for the "Students" folder

Within the "Students" folder, as the Teachers create folders they will most likely alter the default permissions for each folder. Actually, each of these folders becomes a case of the previous scenario (A Place in a School Classroom) and the default permissions could be adapted from that scenario. For the sake of completeness, the default permissions for the "Students" *Child Folders* can be set as shown in Figure 36, and the default permissions for *Other Folders* as shown in Figure 37.

Permissions and Web-Browser Access

Access of Cmaps and resources in a *Place* (*CmapServer*) through a web-browser follows the same restrictions as accessing through the *CmapTools* client program. Permissions are verified in exactly the same way, and access is restricted if the appropriate *User ID* and *Password* are not provided.

Permissions and Search Results

All *CmapServers* are automatically indexed, and the indexes are stored in an *IndexServer* to speedup retrieval. The *Search* mechanism in *CmapTools* verifies permissions when presenting results of searches to guarantee that the user has permissions to view/access the resources resulting from the search before they are displayed.