QuickStart Manual for Updater44D

1. Preparation of Project Folder

Before you start to use Updater for 4D, prepare your Project folder. You can create one project folder for each application you want to update with Updater for 4D.

In the Project folder, create separate folders for each group of 4D application you want to upgrade. You can set up the groups the way it make sense to you – in the example project, we use separate groups for uncompiled, compiled and compiled and merged application.

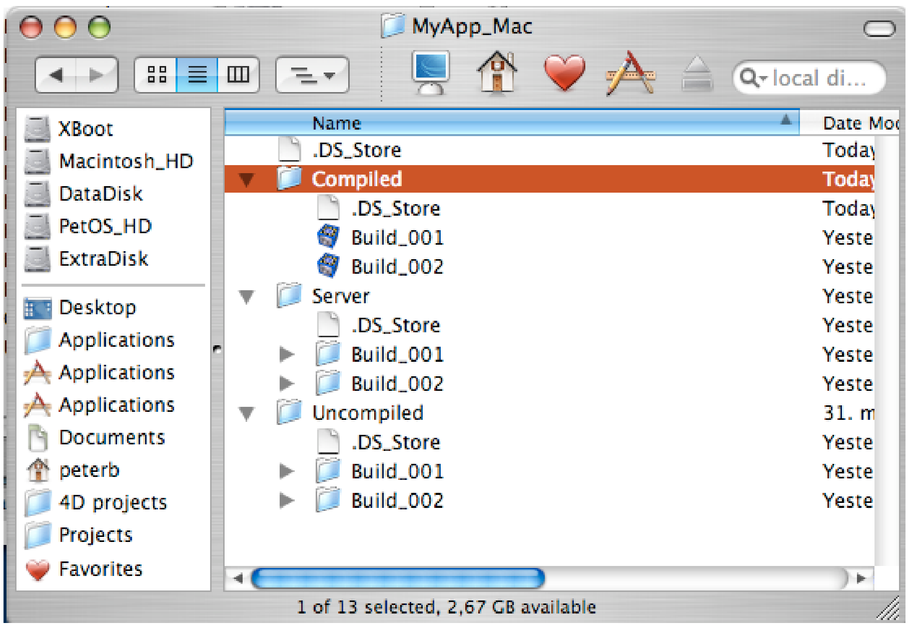
Even if you do not plan to use groups, you have to set up one Group folder.

The names of Group folders are read into Updater for 4D application, so use names that has less then 31 characters and avoid diacritical characters for compatibility.

In the Group folders create Version folders that will be roots of update hierarchies. Updater for 4D can create updaters between two folders – source and destination.

On Mac OS X, you can put compiled and merged application to group folder directly, as Mac OS X application is a bundle (folder.)

The Version folders must be named in manner that, when sorted alphabetically, are ordered from oldest version to latest version. The names of Version folders do not need to be the same as names of folders in which the applications are installed (or names of application.)

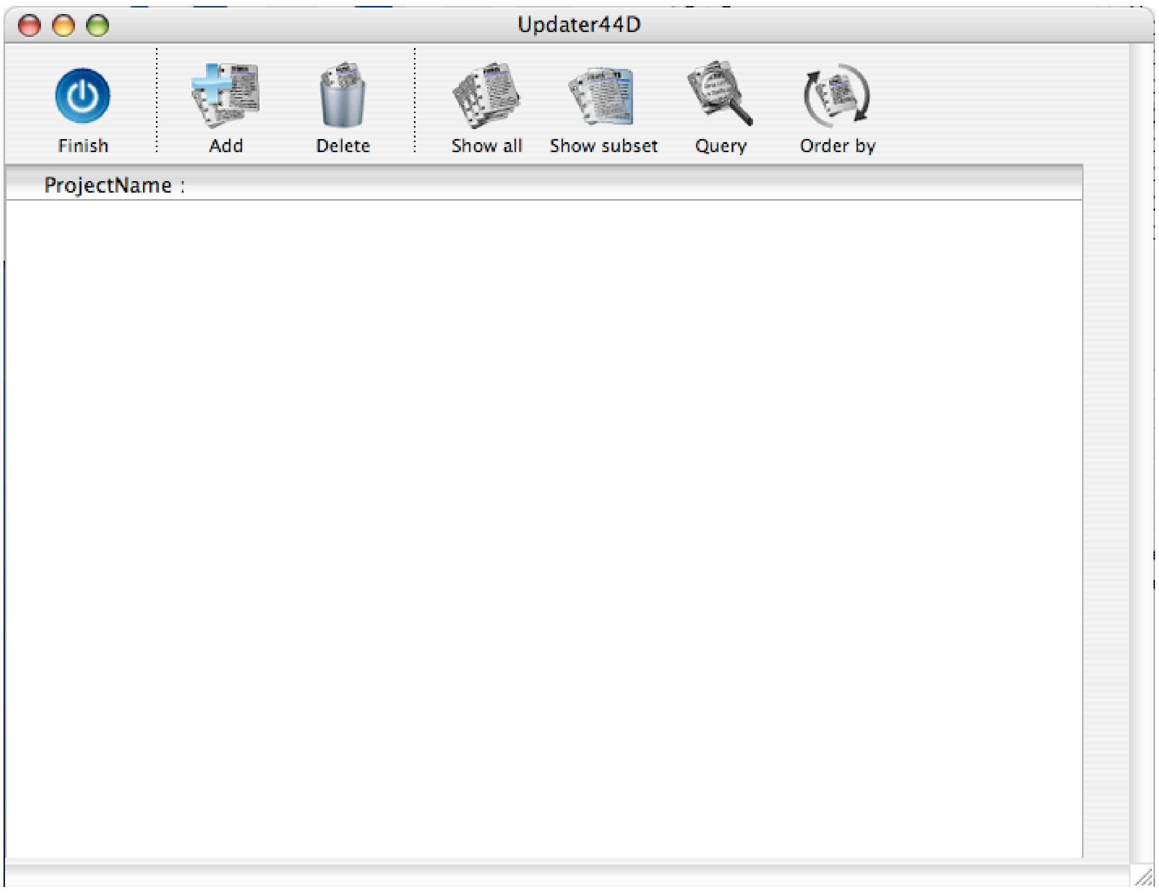


Name	Date Mod
.DS_Store	Today
Compiled	Today
.DS_Store	Today
Build_001	Yeste
Build_002	Yeste
Server	Yeste
.DS_Store	Yeste
Build_001	Yeste
Build_002	Yeste
Uncompiled	31. m
.DS_Store	Yeste
Build_001	Yeste
Build_002	Yeste

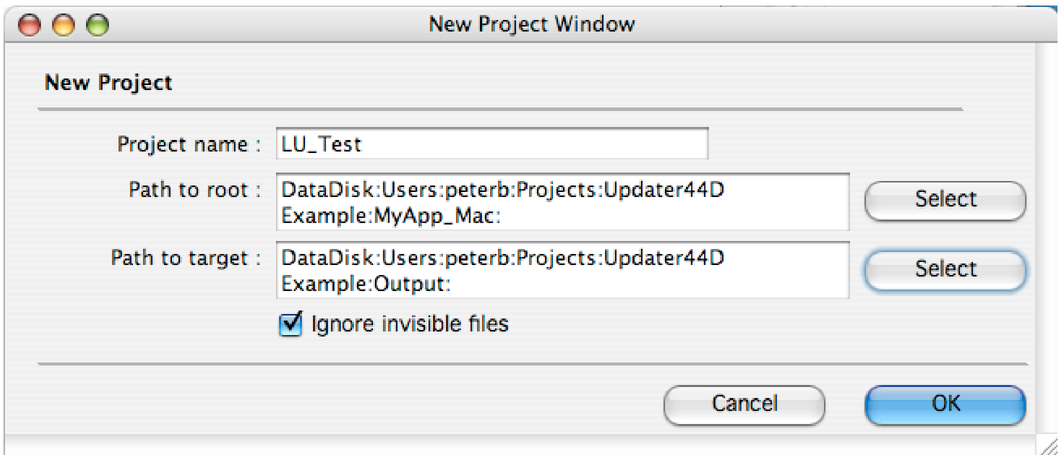
1 of 13 selected, 2,67 GB available

2. Creation of Updater project

Now you can proceed to create a new project in Updater for 4D application. Open Updater for 4D. Updater Projects window should appear. If it is closed, you can always select Updater item from File menu.



Click on New button. The New Project Window will be displayed, where you can set up basic information about project.



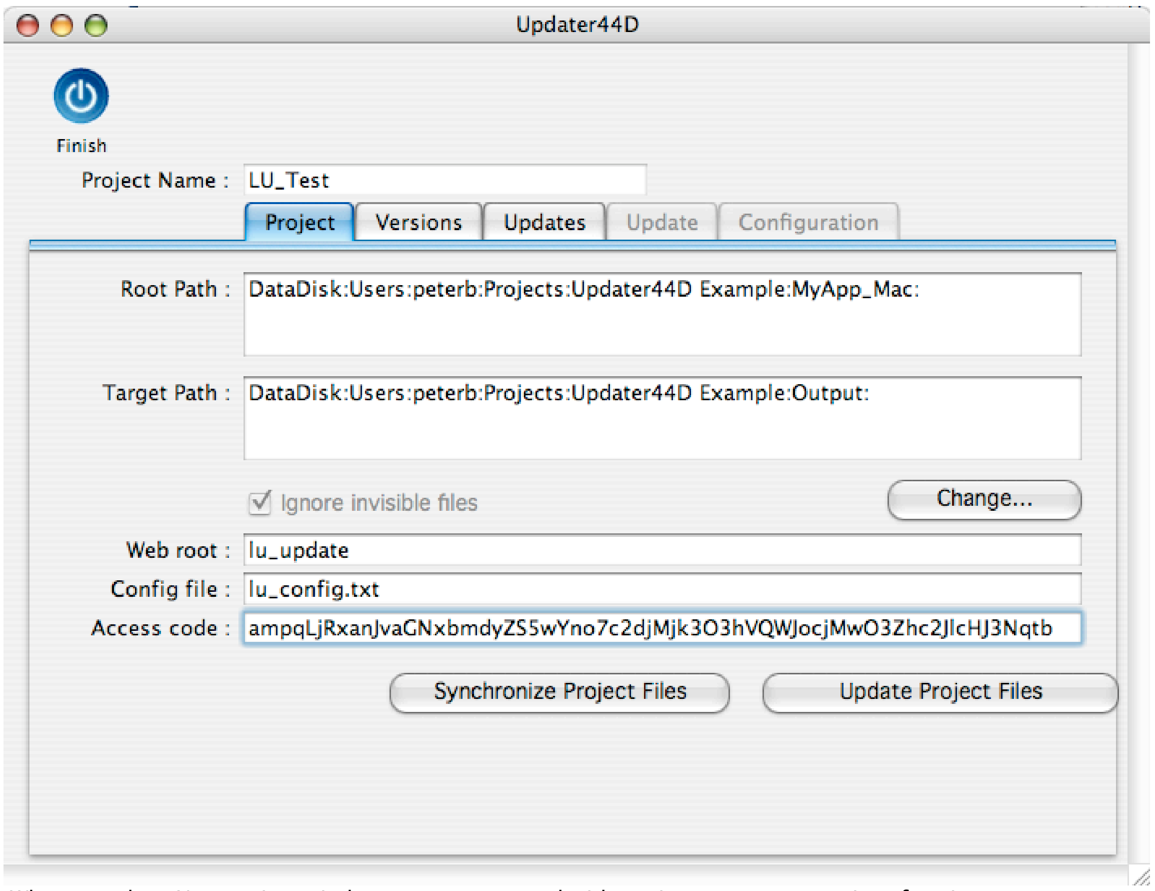
Project name : LU_Test

Path to root : DataDisk:Users:peterb:Projects:Updater44D Example:MyApp_Mac:

Path to target : DataDisk:Users:peterb:Projects:Updater44D Example:Output:

[✓] Ignore invisible files

Path to root should contain path to project root folder. Path to target should contain path to folder where Updater will put created files. The contents of Target Folder will be updated to web server where the files will be accessed by U4D Component.


Updater44D

Finish

Project Name : LU_Test

Root Path : DataDisk:Users:peterb:Projects:Updater44D Example:MyApp_Mac:

Target Path : DataDisk:Users:peterb:Projects:Updater44D Example:Output:

☑ Ignore invisible files

Web root : lu_update

Config file : lu_config.txt

Access code : ampqLjRxanJvaGNxbmdyZS5wYno7c2djMjk3O3hVQWJocjMwO3Zhc2JlcHJ3Nqtb

When you close New project window, you can proceed with setting up rest properties of project.

Web root is name of folder that will be created on web server and that will contain created documents. You can create separate web root folder for each project, or keep one folder for all projects.

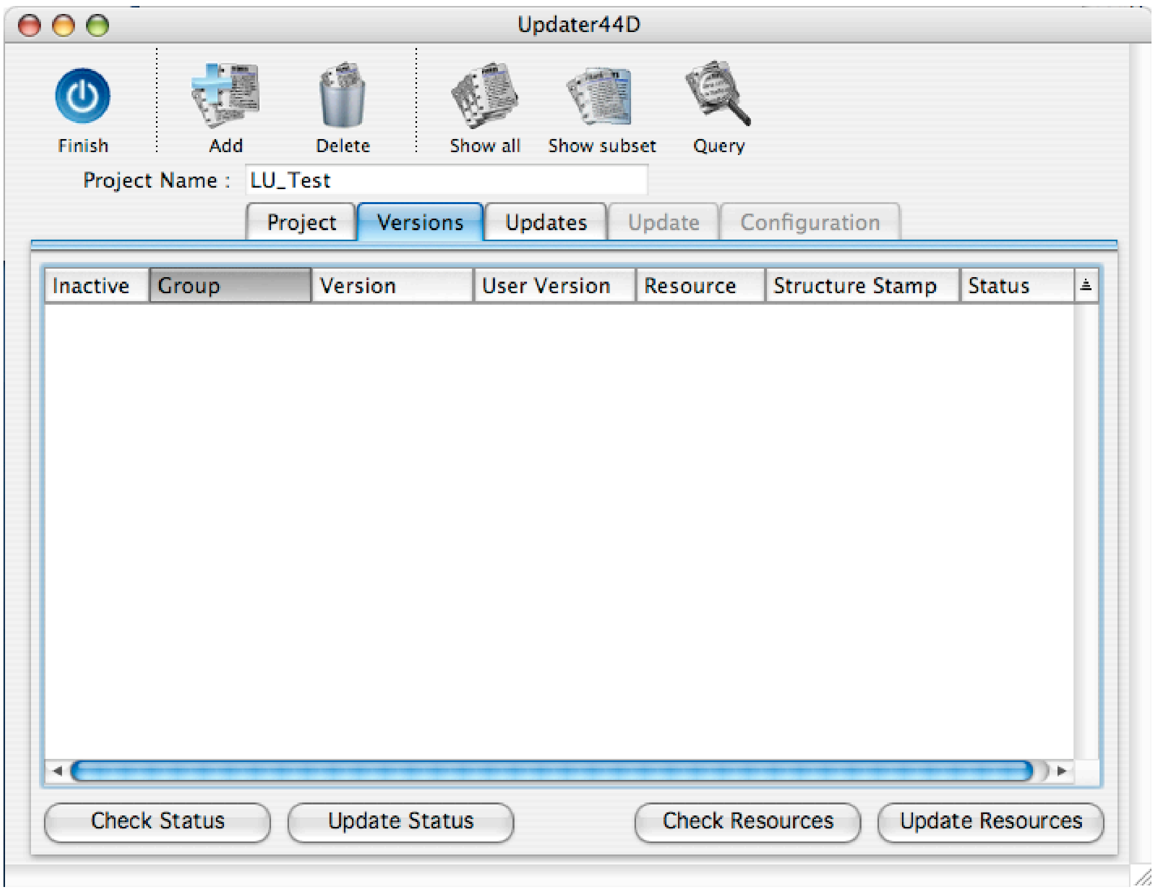
Config file is name of file that will contain description of created updater files. The config file is downloaded and processed by U4D component.

Neither Web root nor Config file can be empty. These strings must be the same as the strings passed as second and third parameter to U4D_Req method of U4D Component.

Access code is code provided by eNode or INFORCE Bratislava. It contains encoded information needed for communication with web server.

3. Preparing the update information

Now you can move to prepare the ingformation needed for the updates. Click on Versions Tab.

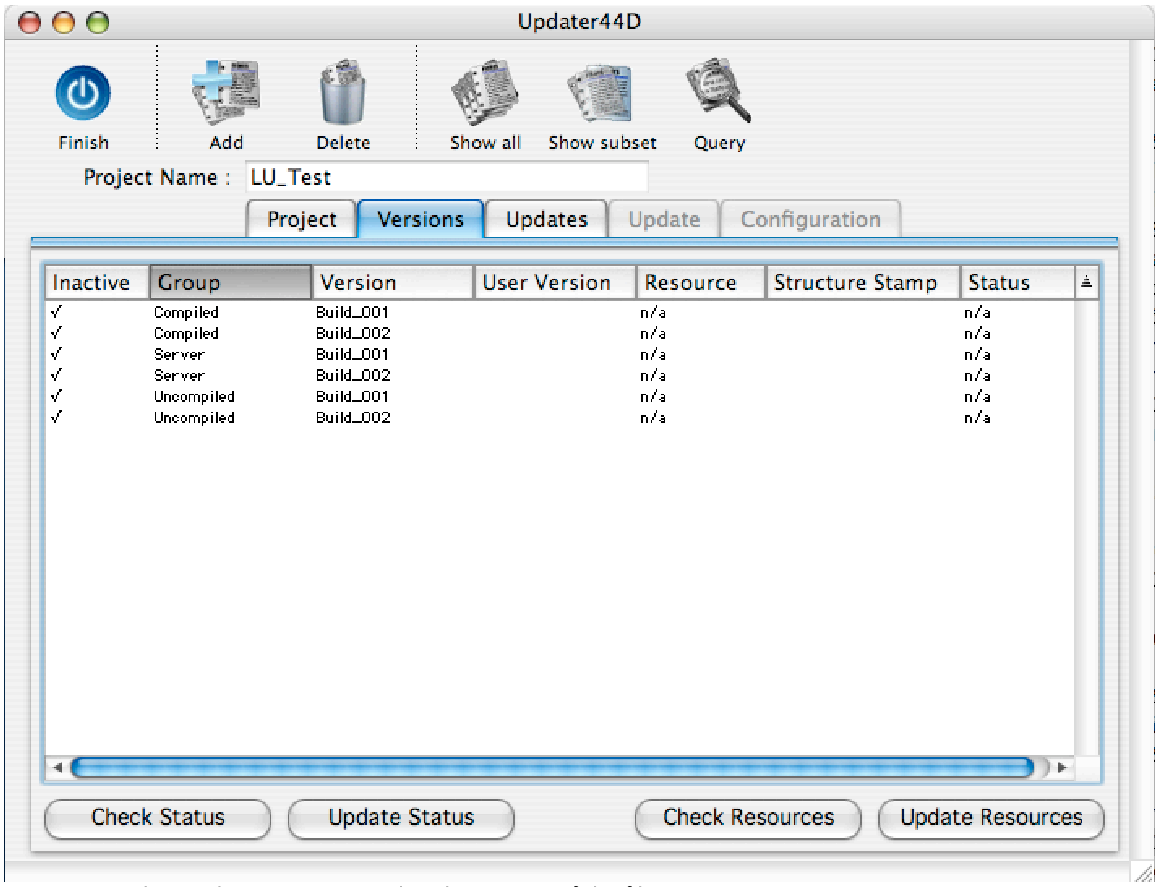


Project Name : LU_Test

Inactive	Group	Version	User Version	Resource	Structure Stamp	Status

Click on Add Button. Updater will parse the project root folder and based on its contents, it will create records for each version it finds.

As example we provided project with three groups, each containing two projects. After parsing, following project records are created:



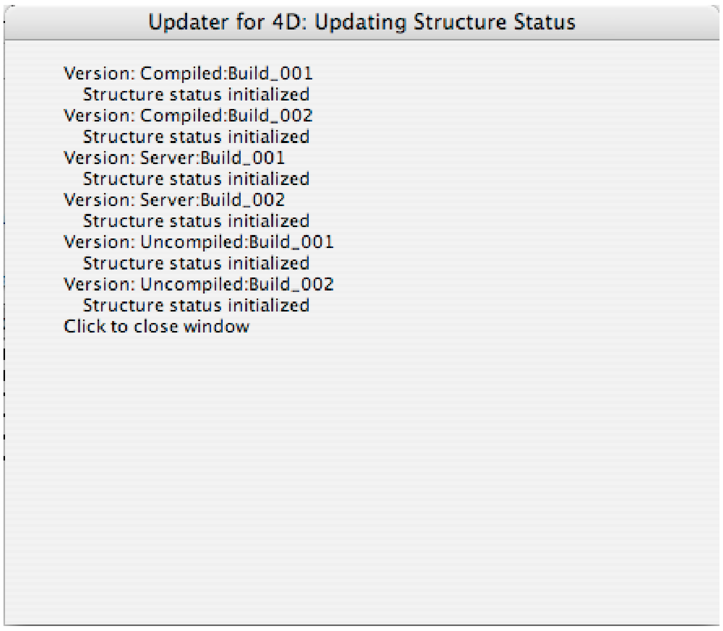
Project Name : LU_Test

Inactive	Group	Version	User Version	Resource	Structure Stamp	Status
√	Compiled	Build_001		n/a		n/a
√	Compiled	Build_002		n/a		n/a
√	Server	Build_001		n/a		n/a
√	Server	Build_002		n/a		n/a
√	Uncompiled	Build_001		n/a		n/a
√	Uncompiled	Build_002		n/a		n/a

Now you need to update resources and update status of the files.

When you update resources, Updater will put into structure file a resource that will be later read by U4D Component. This information is needed o decide which updaters were created from this version and can be applied to this version. Because of this, it is necessary you distribute to customers version with updated resources. Even if you cannot produce any updater yet, because there is no newer version available, you have to put the application into project folder and let Updater update resources before you send the version to the user. Without that, U4D Component will not work correctly.

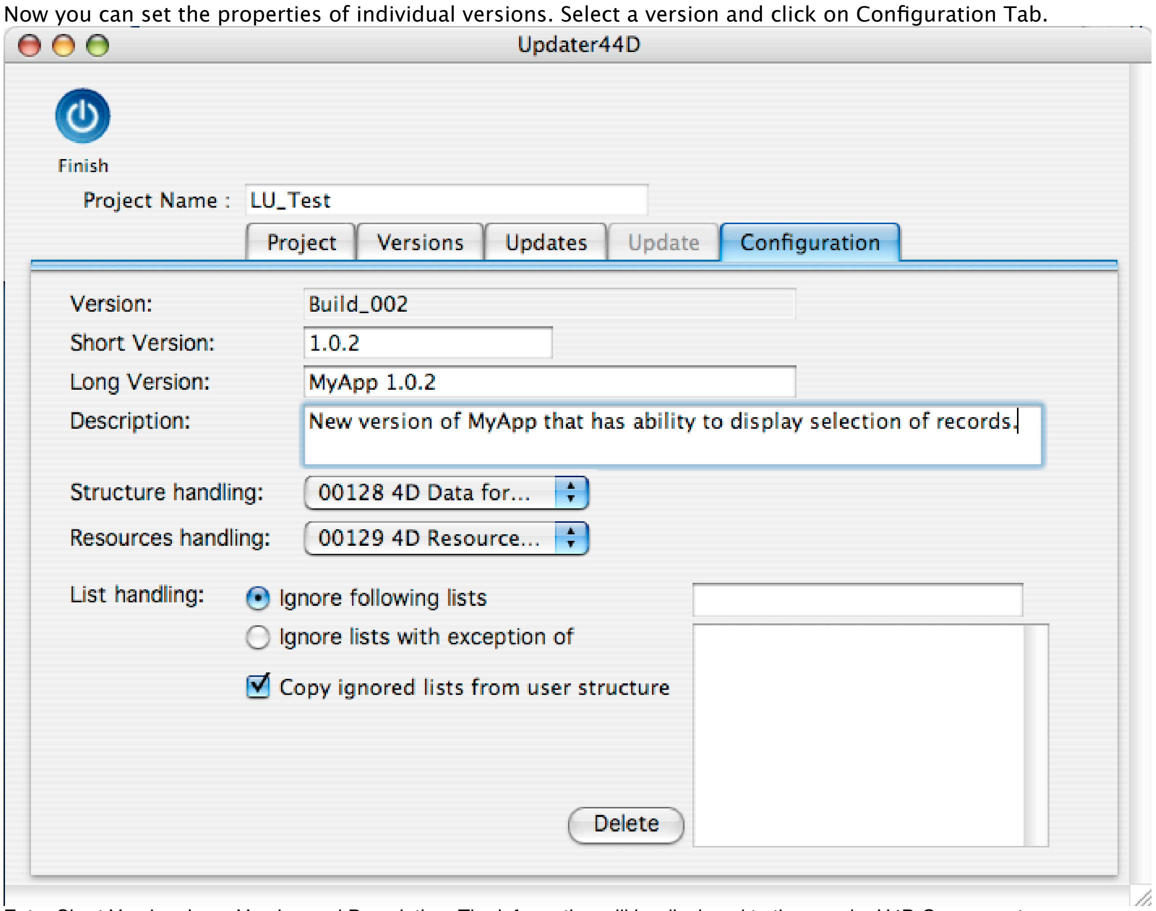
Unlike resources, Status is just an internal information telling that the structure file was not modified. When you Update status, the date time stamp of the structure is saved to the data. Building an update saves the date time stamp too. You can later check the status to make sure the structure was not modified after the update was created.



```
Version: Compiled:Build_001
  Structure status initialized
Version: Compiled:Build_002
  Structure status initialized
Version: Server:Build_001
  Structure status initialized
Version: Server:Build_002
  Structure status initialized
Version: Uncompiled:Build_001
  Structure status initialized
Version: Uncompiled:Build_002
  Structure status initialized
Click to close window
```

During lengthly operations, Updater display its progress in message window. After the operation is finished, the message window is displayed until you click into it. This allows to check the result of individual operations.

Now you can set the properties of individual versions. Select a version and click on Configuration Tab.


Updater44D

Project Name : LU_Test

Version: Build_002

Short Version: 1.0.2

Long Version: MyApp 1.0.2

Description: New version of MyApp that has ability to display selection of records.

Structure handling: 00128 4D Data for...

Resources handling: 00129 4D Resource...

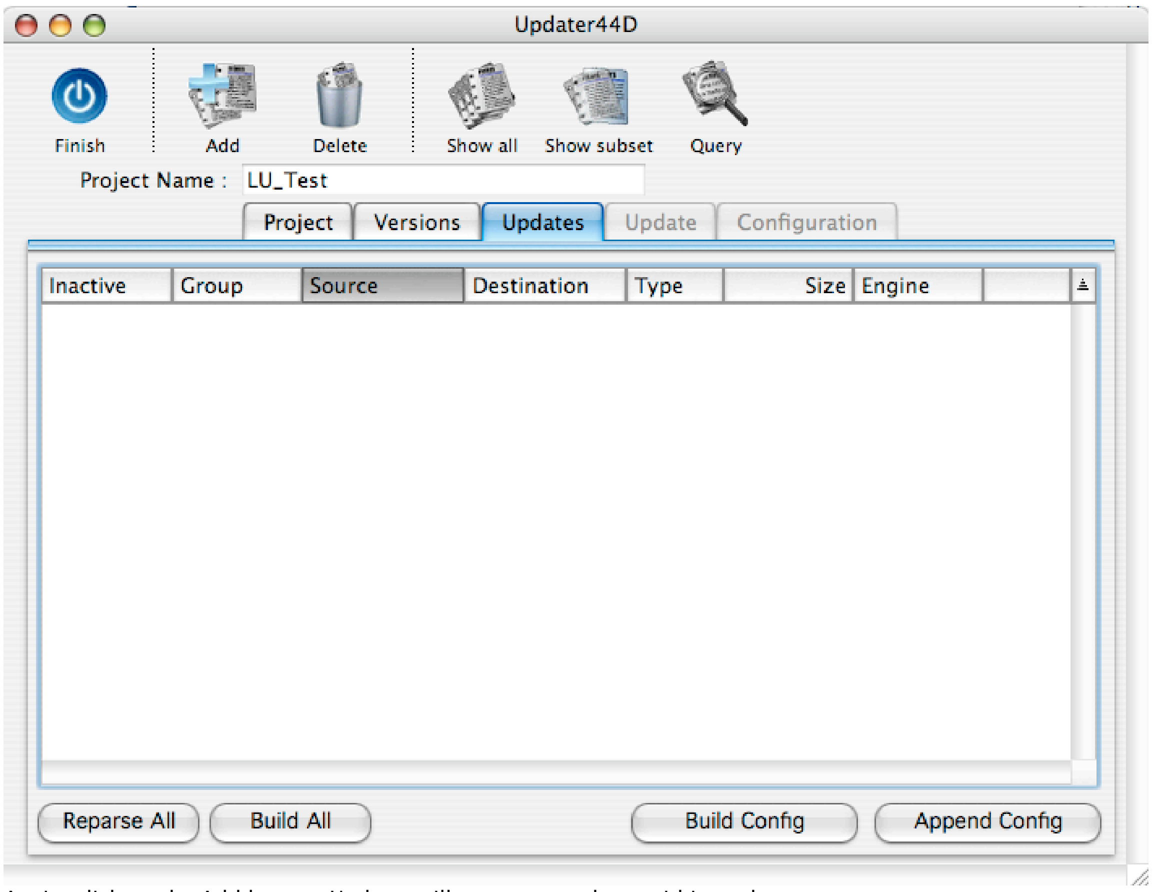
List handling: [✓ Ignore following lists] Ignore lists with exception of

[✓] Copy ignored lists from user structure

Enter Short Version. Long Version and Description. The information will be displayed to the user by U4D Component.

Structure handling and Resources handling allows to set handling of objects in data fork and resource fork of the structure. To achieve correctly working updated application, some objects in both data and resource forks of the 4D structure can be ignored, while others need to be updated. 4D Updater contains few settings that should suit most applications. For compiled application, you can set Structure handling to "Compiled only."

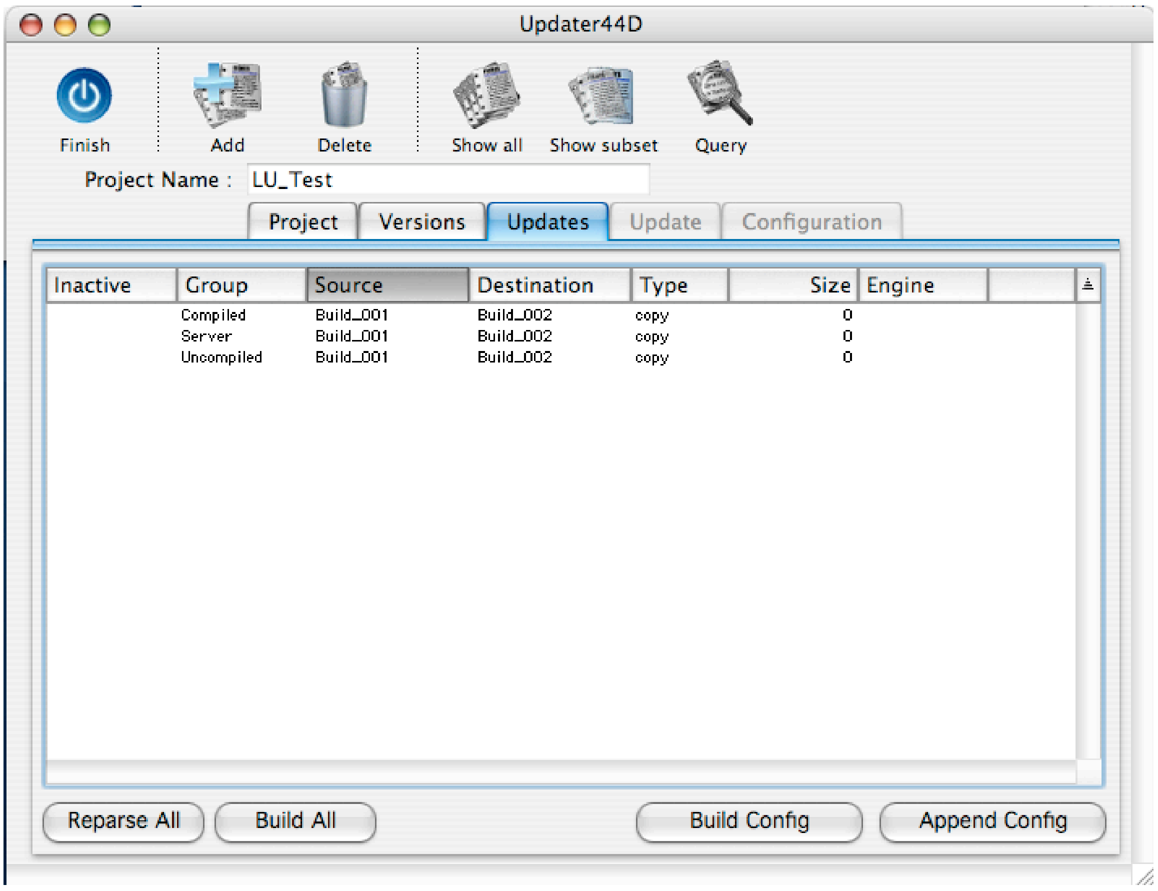
Now we can move to Updates Tab.



Project Name : LU_Test

Inactive	Group	Source	Destination	Type	Size	Engine

Again, click on the Add button. Updater will create an updates within each group.

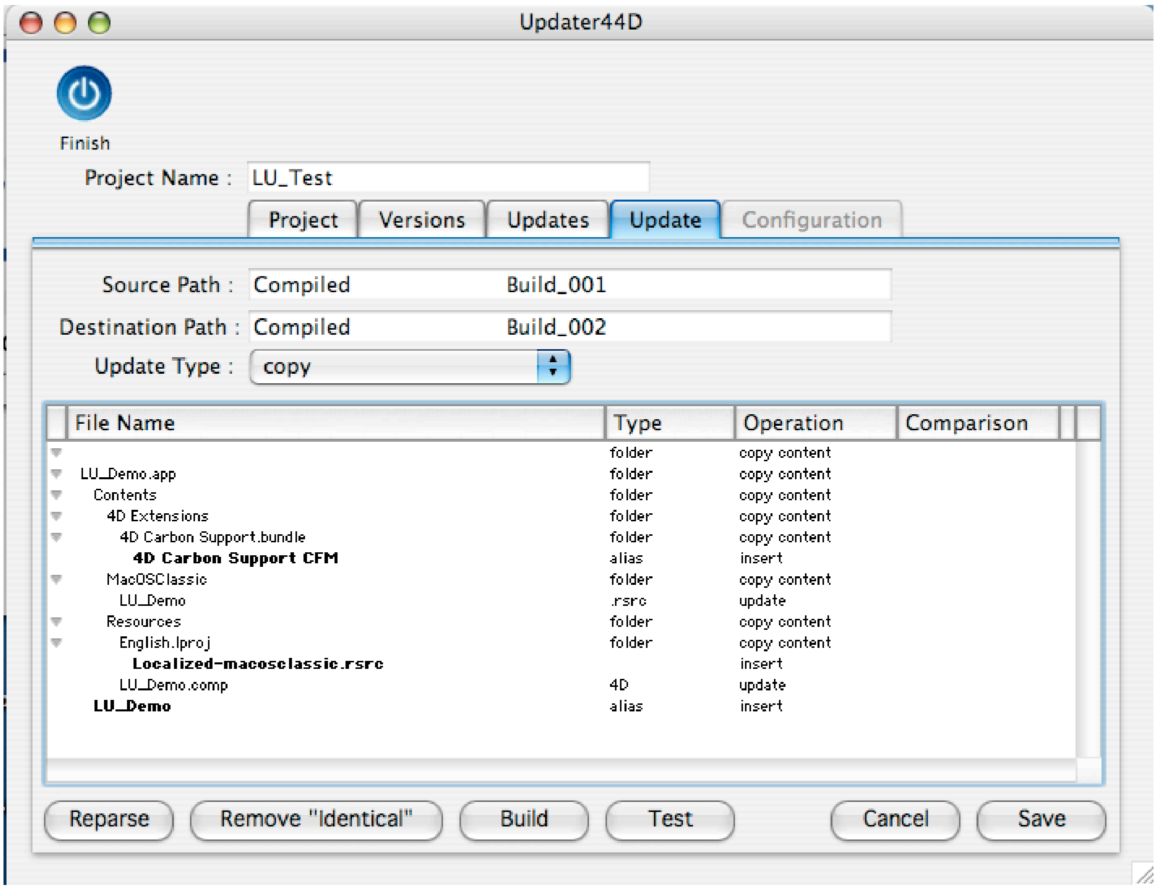


Project Name : LU_Test

Inactive	Group	Source	Destination	Type	Size	Engine
	Compiled	Build_001	Build_002	copy	0	
	Server	Build_001	Build_002	copy	0	
	Uncompiled	Build_001	Build_002	copy	0	

Click on Reparse All button. Updater will go through the updates and for each it will create hierarchy of files in source and destination project folders and compare them.

You can select one update and click on Update Tab to look at it properties.



Project Name : LU_Test

Source Path : Compiled Build_001

Destination Path : Compiled Build_002

Update Type : copy

File Name	Type	Operation	Comparison
	folder	copy content	
LU_Demo.app	folder	copy content	
Contents	folder	copy content	
4D Extensions	folder	copy content	
4D Carbon Support.bundle	folder	copy content	
4D Carbon Support CFM	alias	insert	
MacOSClassic	folder	copy content	
LU_Demo	.rsrc	update	
Resources	folder	copy content	
English.lproj	folder	copy content	
Localized-macosclassic.rsrc		insert	
LU_Demo.comp	4D	update	
LU_Demo	alias	insert	

Update type can be following:
copy - new version will be created (copied into) new folder, old version will be preserved,
in-place - new version will replace the old version
"safe in-place" is similar to "in place" option, but instead to replacing files in old version, it builds new version (as if it was creating separate folder with new version) and it moves old version to temporary files and replaces it with new version.
"safe rename is similar to "copy" option, but will, after successful creation of new version, append "old" to old version folder name and set folder name of new version to former name of old version. As a result, user will have new version in place of old and still keep old version.

Except in-place version, other option may require significant space in disk. Second, you should make sure user does not put data file in project folder hierarchy, as it would be copied while 4D application still runs.

For each item in update hierarchy you can change its attributes by double-clicing on the item line in hierarchical list.

The files can have following atributes:

Update - the source and destination hierarchies both contain the file. Updater will create diff file and put it in update document. The user's hierarchy that is going to be updated must contain the file and it must be identical to the file from which was the update created.

Insert - default action in case when the file exists only in destination hierarchy. Whole file will be put in update document.

Ignore - this file will be ignored. In case of in-place update it will not be removed if it is already present in updated application and the user will find it in his new version.

No change - the file is in both source and destination hierarchies and both files are identical. The user's source hierarchy must contain this file and it must be identical to the file in the hierarchies from which was update document created.

In case of resource files (.RSR, .4RD, .rsrc files) and 4D structure files (.4DB and .4DC files), Updater for 4D may show that the files are "internally identical." It means that while files are binary distinct, the data contained in them are identical (for example, resource files contain the same resources with identical data, but the resources are stored within the files in different order.)

The user's source hierarchy may contain additional files compared to source hierarchy from which was update document created.

Folder atributes define what will Updater do with such files:

Copy - this will copy the additional files into the new version. Additional files (not present in update document) will be present in new version of the application.

Ignore (delete) - this will ignore or delete the additional files, depending on settings of the update. As a result additional files will not be present in new version.

Updater for 4D offers following actions:

Reparse will parse again source and destination hierarchies. Attributes of files and folders will be reset in this version of Updater for 4D.

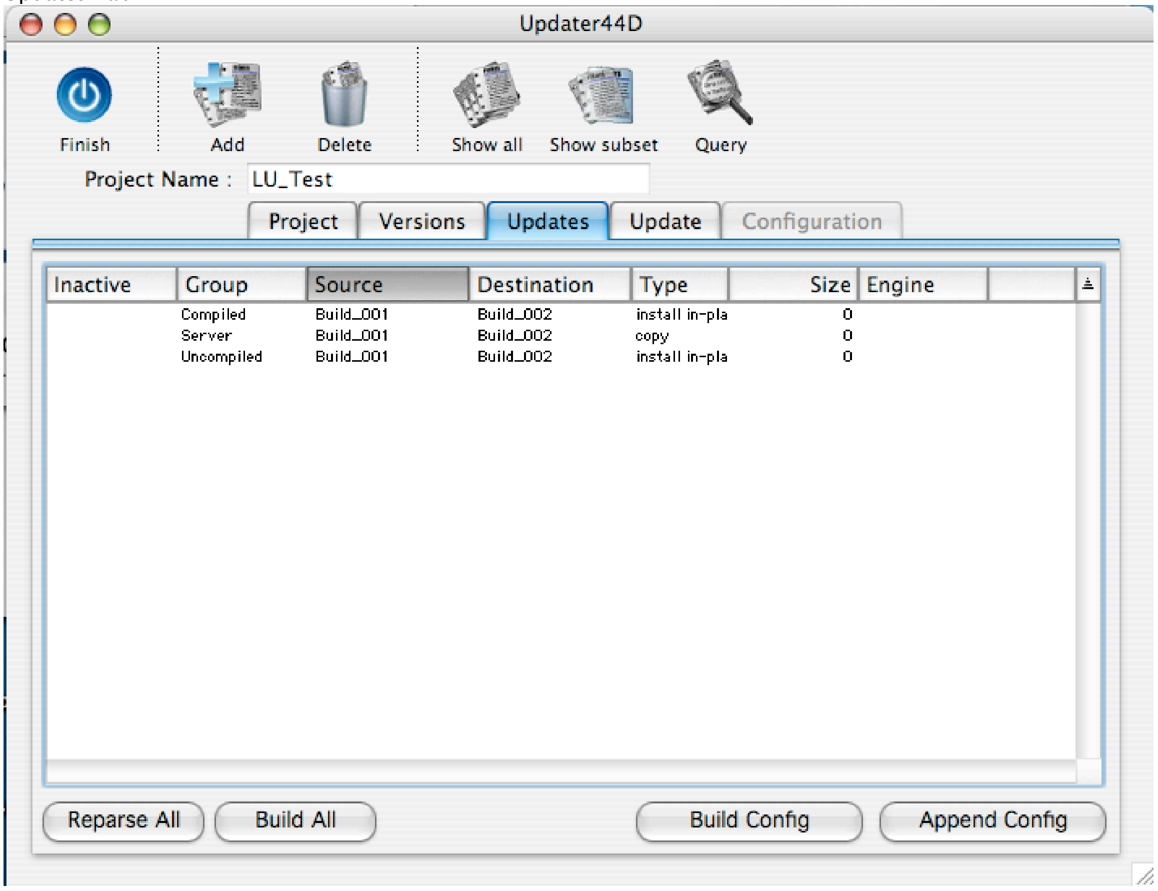
Remove "Identical" will remove identical files from project. To include identical files into update is superfluos.

Build will build an update document. The update document will be created in Output folder and will have an extension of .4UP.

Test will build new version of the application using old version from source path and built update document. The built version will be placed in folder "_TEST_" in Output folder. The test process will create two files, *Project Name*.log and *Project Name*.4UP.txt. These files can be checked to solve problems.

4. Building the updates

While the updates can by build in Update Tab, it is easier to build all updates by clicking at Build All button in Updates Tab.



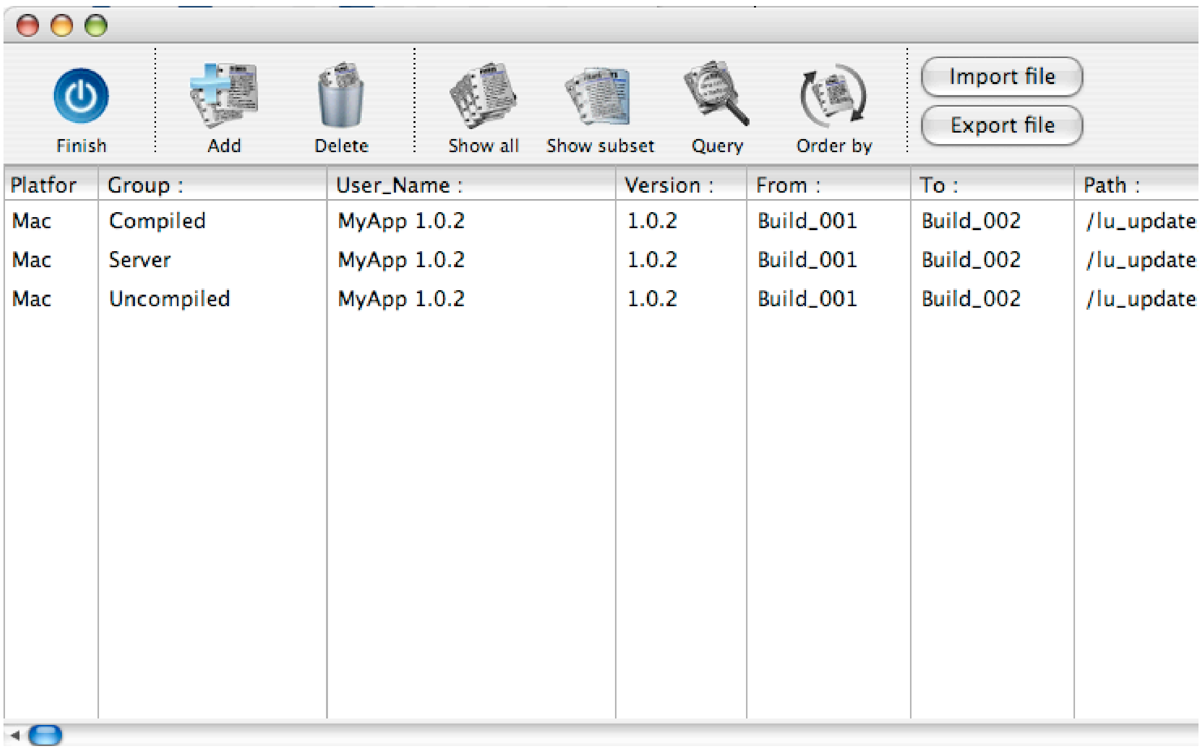
Project Name : LU_Test

Inactive	Group	Source	Destination	Type	Size	Engine
	Compiled	Build_001	Build_002	install in-pla	0	
	Server	Build_001	Build_002	copy	0	
	Uncompiled	Build_001	Build_002	install in-pla	0	

The updates will be build and put into Output folder.

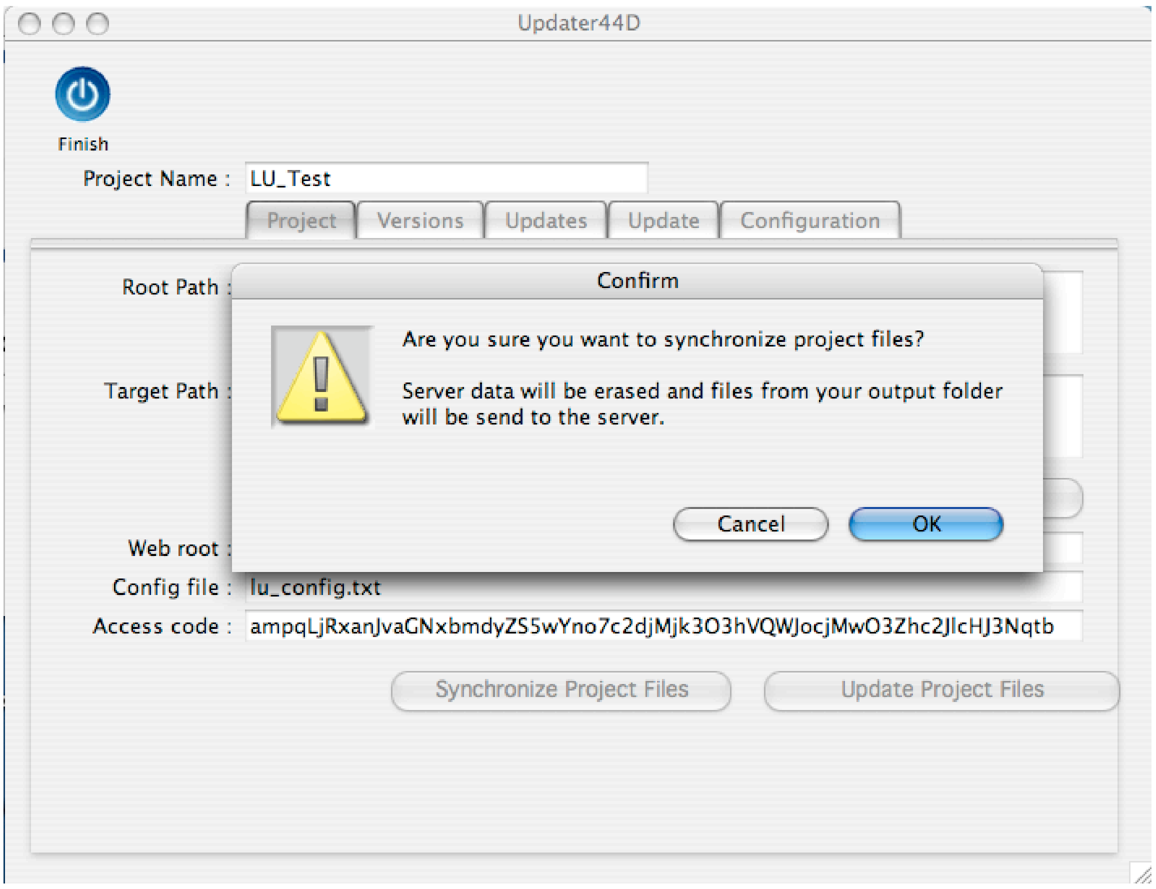
After the updates are build, you have to produce Config file. The Config file is a text file and contains description of build updates. It is placed in Output folder too.

You can check if Output folder contains all updates and Config file. Names of update files consist of name of Group, source version and destination version, separated by underscore. The content of Config file can be checked by through the "Configuration file" menu item. The command will open window where you can import, modify and export Config files.



Platfor	Group :	User_Name :	Version :	From :	To :	Path :
Mac	Compiled	MyApp 1.0.2	1.0.2	Build_001	Build_002	/lu_update
Mac	Server	MyApp 1.0.2	1.0.2	Build_001	Build_002	/lu_update
Mac	Uncompiled	MyApp 1.0.2	1.0.2	Build_001	Build_002	/lu_update

After you check the content of Output file, you can upload it to the web server. Go to the Project Tab and click on the Synchronize Project Files button.



Project Name : LU_Test

Root Path :

Target Path :

Web root :

Config file : lu_config.txt

Access code : ampqLjRxanJvaGNxbmdyZS5wYno7c2djMjk3O3hVQWJocjMwO3Zhc2JlcHJ3Nqtb

Are you sure you want to synchronize project files?

Server data will be erased and files from your output folder will be send to the server.

After you confirm the dialog, the files from the Output folder will be send to the web server.

Synchronize Project Files button will remove all files from the web server first, while Update Project files will just send the files in output folder to the web server.