Updater

User Guide

CatenaLogic

Author	CatenaLogic
Version	1.0.0.1
Date	2007-03-21
Website	http://www.catenalogic.com

Disclaimer

Table of contents

Definitions, acronyms and abbreviations

Word	Explanation
DLL	Dynamic Link Library
FTP	File Transfer Protocol
HTTP	Hyper Text Transfer Protocol
LAN	Local Area Network
OCX	OLE Control Extension

1. Introduction

This document explains all features available in Updater. This document is divided into several chapters to make it as easy as possible to find the information you might be looking for.

All features explained in this user guide can be used in the newest version of Updater. Be sure you are using the right user guide. To find out which user guide version to use, check out the Updater readme. The readme file is included in the Updater package which can be downloaded at http://www.catenalogic.com.

Only use this document when searching for features of Updater. It is recommended to use Updater Tool to create the update files. With the tool, it is possible to easily create update files in a clear, document based, user interface.

The settings file can also be edited by a tool which is located in the compiled Updater package, which can be downloaded at http://www.catenalogic.com. It is recommended to use this tool to minimize user errors.

2. Features

Updater is an application which enables the user to check if there is a new version available or an application. It is very easy to integrate Updater into any (existing) project. The key-features of Updater are:

- Selective updates
- Event driven with custom actions at each event
- Different user interfaces, from full wizard to hidden modes
- Proxy support
- Rollback & restore to rollback current updates or even restore an older version of the application
- Software protection to prevent users using illegal versions
- Multilingual
- Different (customizable) update actions:
    - Download
    - Copy
    - Run
    - Delete
    - Unzip
    - Set file attributes
    - Register
- Modify registry keys
- Modify ini-files

3. Steps

3.1. Welcome

This step will welcome the user to Updater and explain what actions Updater is going to perform to make sure the products the user is using in combination with Updater will be up-to-date.

3.2. Check version

Updater will check if there is a new version available. When a new version is available, Updater will show the new (most important) features available in the newest version.

When no new version is available, Updater will quit.

3.3. Minimum requirements

Updater will check if the system of the client meets the minimum requirements of the software that is updated. If not, the user will have to work with the old version.

3.4. HTML message

Updater will show a HTML message in this step. It is possible to set the minimum time the user should take a look at the HTML message. The message can be customized, for example to advertise for some other interesting products available.

3.5. License

This step can be used to add a license which must be accepted by the user. If the user does not accept the license, he / she can't continue updating.

3.6. Protection

Software protection is a very important feature of Updater, since a lot of applications are *cracked* nowadays. The protection step checks if the files are not patched by comparing hashes. Also, a custom check method can be used by customizing the protection DLL.

3.7. Select update

When an application contains different parts, this function can be very useful. For example, an application can have a lot of plug-ins or DLL's. With Updater, it's possible to let the user decide which parts (in Updater, these parts are called sections) to update at that time.

3.8. Download

In this step, all files needed for the updates are downloaded.

3.9. Install

In this step, all actions will be performed to make sure the product gets updated.

3.10. Validate

In this step, Updater will validate a list of files specified by the developers to check if the update was successful. This step is optional.

3.11. Finish

Updater will explain to the user that the update is successful.

3.12. Rollback

This step will be started when the user cancels the update of when an error occurs. In this step, Updater rolls back all the actions already performed in the update to make sure the product will keep working.

4. Architecture

In this part of the user guide, the architecture of Updater will be explained. The best way to explain the architecture is by using an image.

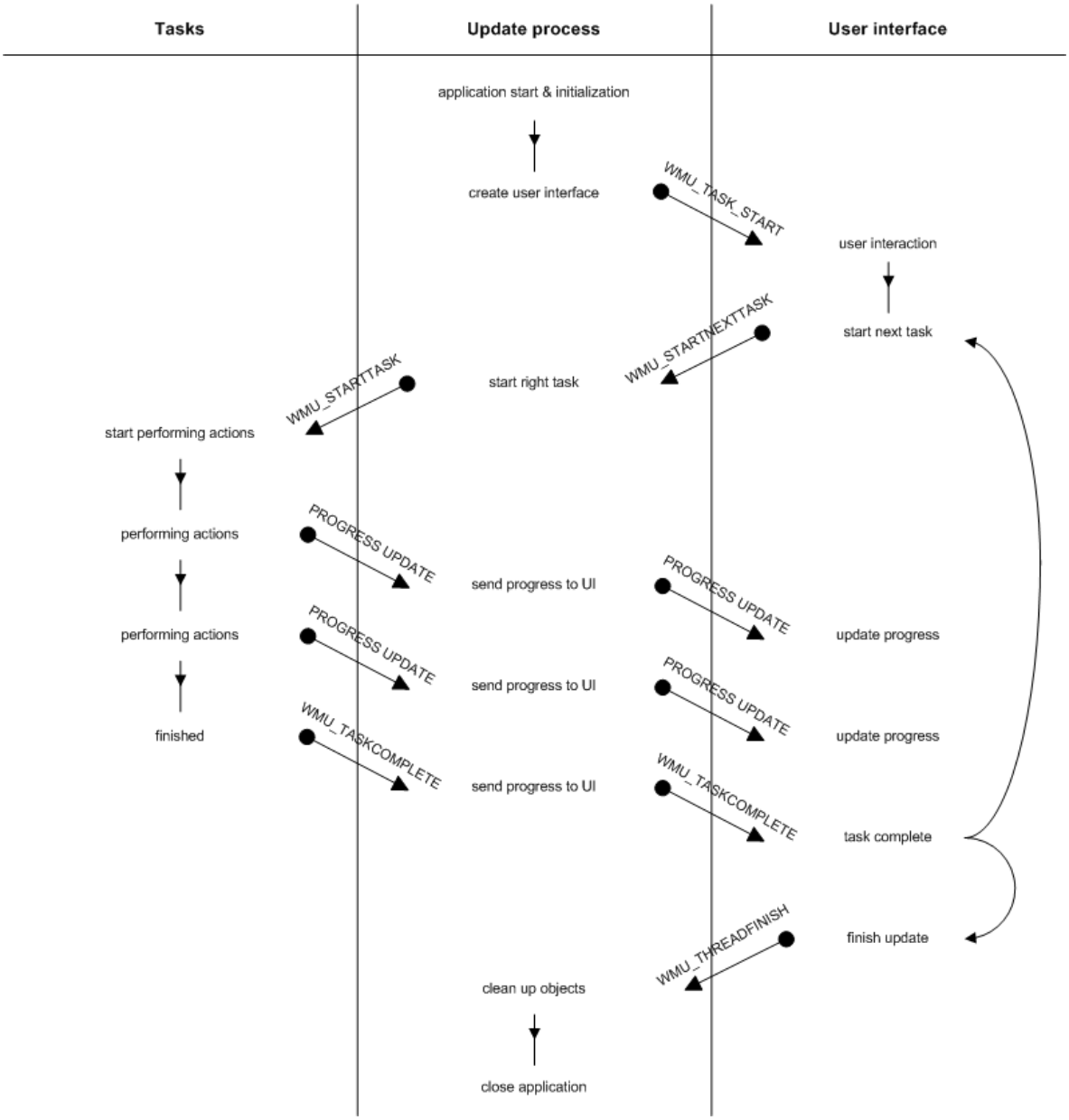


Figure 4-1 Updater architecture

First, Updater initializes itself and creates the needed directories. It also loads a language file and starts a log file.

After the start-up, the main process is starting a user interface. What kind of user interface is loaded is not important for the main process because all the user interfaces receive the same messages and should response in the same way. The user interface decides whether the user can interact or not. The user interface informs the main process that is it ready to start the next task. The user interface cannot decide which task is started. The main process knows which task is coming next.

The main process starts a new thread with the new task. This task will calculate all the actions necessary and start performing the actions. Again, it is not important what kind of task is started because all the tasks receive the same messages and should send the same messages. The task itself decides if the user should see a progress update or not by sending update progress messages with the status information. After a task has finished, the task will send a final message to the main update process.

These steps are repeated until all tasks are finished. After finishing all tasks, the user interface will send a message that the user wants to end the application by sending it a thread finish message.

Finally, the main process will clean up all objects and remove the temporary folders. The language xml is released and the log is closed. Finally, the main process itself is closed.

As visible, Updater is very flexible because all actions and user interfaces receive and send the same messages. This way, it is very easy to implement a new user interface in a very short amount of time. Most time will be spent to the designing of the new user interface.

Also, when a new task must be added, this is very easy because the other tasks in Updater will still work without adjusting them to the new tasks.

5. Modes

There are several different modes available. It is possible to switch to another mode at runtime. An explanation of the modes will follow.

To find out how to set the default mode, read chapter run mode (11.1.1).

5.1. Full mode

In this mode, the user will have full control over Updater. Updater will run as a wizard and will guide the user step by step.

The user will see all messages and is able to cancel the update process at any time.

5.2. Silent (automatic) mode

Silent mode will run in the system tray using icons. Each icon represents a state of the update process. The user is able to view the progress by hovering over the icon. The information will appear in a tool tip.

Each event is introduced by a tool tip balloon with additional information.

When the update is finished, it is possible to enable a popup menu so the user can perform actions using the popup menu.

All steps are started automatically, so the user can't start steps manually.

5.3. Very silent (automatic) mode

Very silent mode works exactly the same as silent mode. But, Updater will not show the tool tip balloon with additional information at each event.

5.4. Hidden (automatic) mode

This mode is fully hidden. No user interface is visible to the user. All steps are performed in the background, and there will be no interaction with the user. This means Updater will not ask the user when closing an application, but it will immediately close the application.

All steps are started automatically, so the user can't start steps manually.

This step is recommended for server applications where no users are controlling the computer.

5.5. Restore mode

Sometimes, it happens that an update contains bugs. In time-critical systems, this can be very dangerous and expensive.

Using this mode (which can only be started using parameters), the user will be able to restore an older version of the application.

Only use this mode when it is really needed to restore an older version of the product.

6. Command actions

To support the most common actions that might be performed when updating an application, Updater supports the actions described in this chapter.

6.1. Download

This action enables the developer to download files to the client. A detailed description about the implementation of this action can be found in chapter download (12.9.1).

6.2. Copy

This action enables the developer to copy a file on the client pc. A detailed description about the implementation of this action can be found in chapter copy (12.9.2).

6.3. Delete

This action enables the developer to delete a file on the client pc. A detailed description about the implementation of this action can be found in chapter delete (12.9.3).

6.4. Execute

This action enables the developer to execute a file on the client pc. A detailed description about the implementation of this action can be found in chapter execute (12.9.4).

6.5. Unzip

This action enables the developer to unzip a file on the client pc. A detailed description about the implementation of this action can be found in chapter unzip (12.9.5).

6.6. Set File Attributes

This action enables the developer to set the file attributes of a file on the client pc. A detailed description about the implementation of this action can be found in chapter set file attributes (12.9.6).

6.7. Register

This action enables the developer to register a file (OCX or DLL) on the client pc. A detailed description about the implementation of this action can be found in chapter register (12.9.7).

6.8. Registry

This action enables the developer to create or change registry settings on the client pc. A detailed description about the implementation of this action can be found in chapter registry (12.10).

6.9. Ini-files

This action enables the developer to create or change ini-files on the client pc. A detailed description about the implementation of this action can be found in chapter ini (12.11).

7. Parameters

There are several parameters available to control Updater. Most parameters are used to override the settings in the settings file. When a parameter is available for a specific setting, the parameters is noted at the specified setting.

7.1. Special parameters

Some special modes require special parameters. These are the special parameters not used for settings available in the settings file:

Parameter	Description
-checkforupdates	This parameter can be used to check for updates without actually using a UI or update immediately. This can be very useful when an application wants to inform the user that there is a new version available. The user can then start Updater to update the product.
-log	View log of the latest update
-nolog	No log file will be created
-noskins	No skins file are allowed to be loaded (excludes skin library calls)
-proxysettings	Run in proxy settings mode
-restore	Run in restore mode
-settingsfile [value]	Custom location of the settings file

When providing additional information with a parameters, use double quotes (“ and ”) to use data with spaces. It is recommended to always use double quotes. Some examples:

```
Updater.exe –settingsfile test\myfile.xyz          works

Updater.exe –settingsfile “test\myfile.xyz”        works

Updater.exe –settingsfile my test\myfile.xyz       will not work

Updater.exe –settingsfile “my test\myfile.xyz”     works
```

Note: No space is allowed between the minus (-) and the parameter name.

7.2. Settings parameters

All settings parameters that are available within Updater are listed below. Each parameters is described in detail in chapter 11. All parameters listed below must be used in combination with a value.

Parameter
-acceptcertificates
-applocation
-appname
-approot
-appversion
-checkconnection
-ftppassword
-ftpusername
-httppassword
-httpusername
-language
-mode
-selfupdatelocation
-timeout
-updatefile
-updateselector

8. Return codes

Updater uses several different return codes to inform the application that is launching Updater about its status.

This is a table with all the return codes of Updater:

Return code	Description
-1	An error occurred. Check the log file for more information about this error.
0	No errors, no additional information available.
1	New version available or new version is installed successfully. [1]
2	New version which is mandatory (forced) is available.
3	No new version available.
4	User cancelled the process.

[1] If the parameter -checkforupdates is used, Updater will only check if there is a new version or not. When Updater is run normally, this return code will be returned when the update is installed successfully.

9. File locations

Updater needs some different files to work correctly. A list of the available files:

- Settings file
- Update file
- Sections file
- Self update file

The files should be located as following:

Client

Server

Settings file

Update file

Section file

Self update file [1]

[1] *This file is normally hosted by CatenaLogic. This file should only be hosted on a different location when a custom version (build) of Updater is used.*

10. Server example

When the updates are finished and the update file is generated by Updater Tool, it is time to put all the files on the server. There are several techniques that can be used, but the user guide will describe the most interesting technique.

This technique described below enables the developer to keep all versions on the server, so even sequenced updates are still available after several new versions.

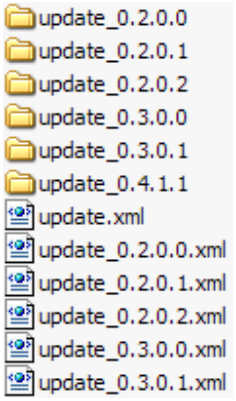


Figure 10-1 Example of server file lay-out

The figure above shows that for each update, a new directory and new update file is created. This way, the settings file can always point to the newest update file by pointing to updatefile.xml.

When a new version is released, there are only 3 steps to perform:

1. Rename *update.xml* to *update_[version].xml* so the update stays available when using sequenced updating.

2. Create new folder *update_[newversion]* and copy the files needed for the update of that specific version to the new folder.

3. Create new *update.xml* file which includes the newest update.

The figure below shows how the server should look like when a new update, 0.5.0.0, is released. The highlighted files and folders are the files that are added and/or changed.

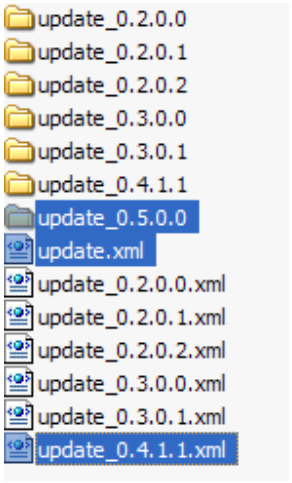


Figure 10-2 Example of server file lay-out after adding a new version

11. Settings file

The settings file is created to store settings for Updater. This file should be located in the folder where Updater.exe is located and must be named as settings.ini. There are several settings that can be defined in the settings file.

All values in settings.ini that are surrounded by %% will be treated as registry values. Updater will try to retrieve the value from the key. If that fails, Updater will handle the data as value instead of a registry key.

11.1. Updater

This group specifies all the settings for Updater. Use [UPDATER] in settings.ini for this group.

11.1.1. Run mode

The default mode of Updater. This will be the default value, meaning this is the start mode. The mode can be changed at run-time at each event.

Setting

runmode

Parameter

-mode [value]

Type

string

Possible values

full
silent
verysilent
hidden

Remarks

Default value is full

11.1.2. Language

Use this option to set the language that Updater will use. The chosen language must be located in the *lng* directory. The name of the xml file (language is located in an xml file) is the same as the name of the language. If the language is not found, Updater will use default language (English).

Setting

language

Parameter

-language [value]

Type

string

Possible values

[empty]	Automatic language detection
user	User can choose a language at start-up

Remarks

A list which language will be chosen:

Language xml file	ID	Code	Language description
arabic.xml	unknown	-	-
chinese_simplified.xml	unknown	-	-
czech.xml	0x0405	CSY	Czech
danish.xml	0x0406	DAN	Danish
dutch.xml	0x0413 0x0813	NLD NLB	Dutch (Standard) Belgian (Flemish)
finnish.xml	0x040B	FIN	Finnish
french.xml	0x040C 0x080C 0x0C0C 0x100C	FRA FRB FRC FRS	French (Standard) Belgian Canadian Swiss
german.xml	0x0407 0x0807	DEU DES	German (standard) Swiss

	0x0C07	DEA	Austrian
greek.xml	0x0408	ELL	Greek
hungarian.xml	0x040E	HUN	Hungarian
icelandic.xml	0x040F	ISL	Icelandic
italian.xml	0x0410 0x0810	ITA ITS	Italian (standard) Swiss
lithuanion.xml	unknown	-	-
norwegian.xml	0x0414 0x0814	NOR NON	Norwegian (bokmal) Norwegian (nynorsk)
polish.xml	0x0415	PLK	Polish
portuguese.xml	0x0416	PTB	Portuguese
russian.xml	0x0419	RUS	Russian
slovak.xml	0x041B	SKY	Slovak
spanish.xml	0x040A 0x080A 0x0C0A	ESP ESM ESN	Spanish (Standard/Traditional) Mexican Spanish (Modern)
swedish.xml	0x041D	SVE	Swedish
turkish.xml	0x041F	TRK	Turkish

11.1.3. Language pack version

The version of the language pack currently used by Updater. Make sure to set this settings correctly if the language pack should also be updated on a self update.

Setting

languagepackversion

Type

string

Possible values

Version number (for example 2.4)

Remarks

Updater can parse some different version separators:

Version separator	Example
.	4.3.1
,	4,3,1
_	4_3_1
|	4|3|1
-	4-3-1
[space]	4 3 1

11.1.4. Self update

If the updates of Updater are hosted on another server than the default one, the location can be specified here.

Setting

selfupdate

Parameter

-selfupdatelocation [value]

Type

string

Possible values

Location of a file, located on a local or remote computer

Remarks

Only use this feature when the default self update feature is not working

11.1.5. Logo small

Use this option to customize the small logo at the top of Updater when running in full mode.

Setting

logosmall

Type

string

Possible values

Location of a local file

11.1.6. Logo large

Use this option to customize the large logo at the left of Updater when running in full mode.

Setting

logolarge

Type

string

Possible values

Location of a local file

11.1.7. Icon

Use this option to customize the default icon used by Updater. This will not replace the icons used to display the download and install progress in silent or very silent mode.

Setting

icon

Type

string

Possible values

Location of a local file

11.1.8. Custom notify

Use this option to enable the custom notify messages in silent mode.

Setting

customnotify

Type

boolean

Possible values

true
false

11.1.9. Link color – link

Use this option to customize the link color of all the hyperlinks in Updater.

Setting

linkcolorlink

Type

RGB Color

Possible values

For each color group, a number between 0 and 255 can be used

Remarks

Default color is light blue (R175, G175, B255)

11.1.10. Link color – hover

Use this option to customize the hover color of all the hyperlinks in Updater.

Setting

linkcolorhover

Type

RGB Color

Possible values

For each color group, a number between 0 and 255 can be used

Remarks

Default color is blue (R0, G0, B255)

11.1.11. Skinfile

The skin file that should be used by Updater.

Setting

skinfile

Type

string

Possible values

Location of a local file

Remarks

Updater uses no skin by default (so the users will see Updater in the normal Windows-style). When this setting is used, Updater will load this skin and use it to skin Updater.

11.1.12. Skinning

When the skinning engine is causing Updater to crash, it is possible to disable the skinning engine by setting this value to false.

Setting

skinning

Type

boolean

Possible values

true
false

Remarks

Default color is true

11.2. Updateinfo

This group specifies all the settings of the update info. Use [UPDATEINFO] in settings.ini for this group.

11.2.1. URL

This option represents the location of the file with all update information. This option should be an URL or a file on the hard-disk. An example is *c:\updater\list.xml* or *http://www.catenalogic.com/updatelist.xml*.

Setting

url

Parameter

-updatefile [value]

Type

string

Possible values

Location of a file, located on a local or remote computer

11.2.2. Update selector

Some products are released in different editions like professional and enterprise. This setting can include a dll which includes a function with this header:

```
string SelectUpdate();
```

This function will be called. In this function, the developer is able to determine which update file should be used for the installed version. The return value is the update file that should be used for Updater.

Setting

updateselector

Parameter

-updateselector [value]

Type

string

Possible values

Location of a file, located on a local or remote computer

Remarks

When this setting is used, the setting URL is not required

11.2.3. Server time-out

It is always possible that a server is not available. To prevent Updater to wait forever, it is possible to set the time-out time for a server. When the server is very fast, a short time-out time can be used. This way, the user won't have to wait a long time, until the default time-out is called.

Setting

servertimeout

Parameter

-timeout [value]

Type

integer

Possible values

Any integer value, value is parsed in milliseconds (1 second = 1000 ms)

Remarks

Default value is 5000 (5 seconds)

11.3. Application

This group specifies all the settings for the application. Use [APPLICATION] in settings.ini for this group.

11.3.1. Name

The name of the application is used to inform the user that this application is being updated. It is possible that two (or even more) update processes of different applications are running. With the name, the user knows exactly which application he/she is currently updating.

Setting

name

Parameter

-appname [value]

Type

string

Possible values

Any string value

11.3.2. Version

This setting sets the current version of the application that will be updated. Updater needs a version to compare with the newest version. It is also possible to add a filename, for example %app%\MyApplication.exe. If the file exists, Updater will check the version of the application itself by reading the executable or dll information. If the file is not available on the hard-disk, Updater will handle this value as the current version.

Setting

version

Parameter

-appversion [value]

Type

string

Possible values

Version number (for example 2.4)
Location of an executable or dll on the hard-disk

Remarks

Updater can parse some different version separators:

Version separator	Example
.	4.3.1
,	4,3,1
_	4_3_1
|	4|3|1
-	4-3-1
[space]	4 3 1

11.3.3. Location

This setting will be used to get a valid value for the %app% constant. Also, this value will be used to close the application and to start it again if requested. This value must contain the location of an executable.

Setting

location

Parameter

-applocation [value]

Type

string

Possible values

Location of a local file
Registry key

Remarks

It is not possible to use %approot% constants here, because the %app% variable is retrieved from this setting. If this field is left blank, %app% cannot be used anywhere in the update file

11.3.4. Root

Sometimes, the root of an application differs from the real application executable path. Therefore, this value can be used. This value is available in the update file using the %approot% constant. When this value is empty, %approot% will have the same value as %app%.

Setting

root

Parameter

-approot [value]

Type

string

Possible values

Local path
Registry key

Remarks

If this field is left blank, %approot% will have the same value as %app%

11.4. Constants

Constants can be used inside the update file to make the file more flexible. Most constants are defined in the update file itself, but sometimes, it is necessary to store some paths in the settings file.

Note: it is not possible to use constants inside other constants in the settings file. Constants can only be used inside the update file.

The constants section looks like this

```
[CONSTANTS]
%name%=value
```

When using constants, *%[CONSTANT]%* must be used (percentages surrounding the constants). It is even possible to override (re-assign) the default constants, but this is not recommended.

There is no limit to the number of constants that can be defined in the settings file.

The custom constants can only be used inside the update file.

Object	Type	Possible values
name	%string%	Any string value
value	string	Any string value

Object	Description
name	The name of the constant as it can be used inside the update file. This value should be surrounded by %. If the % are not used, Updater will add them when parsing the settings file.
value	The value of the constant. If the value is surrounded by %, Updater will treat the value as registry key where the value should be obtained from.

11.5. Connection

This group specifies all the settings for the connection. Use [CONNECTION] in settings.ini for this group.

11.5.1. Check connection

It is possible to check if the user is connected to the internet before starting with any update. When Updater is used in a LAN, this value should be false.

Setting

checkconnection

Parameter

-checkconnection

Type

boolean

Possible values

true
false

Remarks

Default value is false

11.5.2. Accept certificates

When certificates are required to perform a valid update or to obtain files on the internet, enable this setting so Updater accepts certificates automatically.

Setting

acceptcertificates

Parameter

-acceptcertificates

Type

boolean

Possible values

true
false

Remarks

Default value is false

11.5.3. FTP username

FTP username which can be used to authenticate a user to an FTP server. When this value is entered, it is possible to use %ftpusername% in the updatefile.

Setting

ftpusername

Parameter

-ftpusername

Type

string

Possible values

Any string value

11.5.4. FTP password

FTP password which can be used to authenticate a user to an FTP server. When this value is entered, it is possible to use %ftppassword% in the updatefile.

Setting

ftppassword

Parameter

-ftppassword

Type

string

Possible values

Any string value

Remarks

When using this value in the settings file, it should be encrypted. When passing this value via the parameter, it must not be encrypted.

11.5.5. HTTP username

HTTP username which can be used to authenticate a user to a HTTP server. When this value is entered, it is possible to use %httpusername% in the updatefile.

Setting

httpusername

Parameter

-httpusername

Type

string

Possible values

Any string value

11.5.6. HTTP password

HTTP password which can be used to authenticate a user to a HTTP server. When this value is entered, it is possible to use %httppassword% in the updatefile.

Setting

httppassword

Parameter

-httppassword

Type

string

Possible values

Any string value

Remarks

When using this value in the settings file, it should be encrypted. When passing this value via the parameter, it must not be encrypted.

11.6. Proxy

This group specifies all the settings for the proxy the user might be behind. Use [PROXY] in settings.ini for this group.

To enable the client user to change these settings in a user-friendly way, see the chapter about parameters.

11.6.1. Type

This will set the proxy type. There are four types available:

Name	Description
directconnection	A direct connection to the target URL will be made.
autodetect	The proxy server will be auto detected. This is the default and most used value.
manual	It is also possible to specify the http and ftp proxies manual. Use this value to enable the manual proxy type.
autoconfigurl	When an auto config url is used, use this value to select this type.

Setting

type

Type

string

Possible values

directconnection
autodetect
manual
autoconfigurl

Remarks

Default value is autodetect

11.6.2. Username

The username to authenticate the client to his / her proxy server. This setting must be provided by the client user.

Setting

username

Type

string

Possible values

Any string value

11.6.3. Password

The password to authenticate the client to his / her proxy server. This setting must be provided by the client user.

Setting

password

Type

string

Possible values

Any string value

11.6.4. HTTP proxy

This setting can be used to specify a manual HTTP proxy server.

Setting

http

Type

string

Possible values

The format is [IP]:[PORT], for example 127.0.0.1:80

Remarks

This setting is only required when proxy type is manual

11.6.5. FTP proxy

This setting can be used to specify a manual FTP proxy server.

Setting

ftp

Type

string

Possible values

The format is [IP]:[PORT], for example 127.0.0.1:80

Remarks

This setting is only required when proxy type is manual

11.6.6. Automatic configuration URL

This setting can be used to specify an automatic configuration URL.

Setting

autoconfigurl

Type

string

Possible values

Any string value

Remarks

This setting is only required when proxy type is autoconfigurl

11.7. Custom notify

This group specifies all the settings for the custom notification. Use [NOTIFY] in settings.ini for this group.

Custom notifications are only available for silent mode.

11.7.1. Skin

The skin that will be used for the custom notifications. Normally, this is a bitmap which contains some space for a title and description.

Setting

skin

Type

string

Possible values

Name of a bitmap file located in the same folder as the Updater executable

11.7.2. Title color

The color of the title text.

Setting

titlecolor

Type

RGB Color

Possible values

For each color group, a number between 0 and 255 can be used

Remarks

Default color is black (R255, G255, B255)

11.7.3. Title rectangle

The rectangle where the title will be shown. This rectangle is relative to the bitmap's left top position.

Setting

titlerect

Type

rectangle

Possible values

Any rectangle value

Remarks

The rectangle should be formatted like:
[LEFT], [TOP], [RIGHT], [BOTTOM]
0, 0, 150, 150

11.7.4. Description rectangle

The rectangle where the description will be shown. This rectangle is relative to the bitmap's left top position.

Setting

descriptionrect

Type

rectangle

Possible values

Any rectangle value

Remarks

The rectangle should be formatted like:
[LEFT], [TOP], [RIGHT], [BOTTOM]
0, 0, 150, 150

11.7.5. Close rectangle

The rectangle where the user is able to close the notification. This rectangle is relative to the bitmap's left top position.

Setting

closerect

Type

rectangle

Possible values

Any rectangle value

Remarks

The rectangle should be formatted like:
[LEFT], [TOP], [RIGHT], [BOTTOM]
0, 0, 150, 150

11.7.6. Title font face

The font face for the title text.

Setting

titlefontface

Type

string

Possible values

Any available font face name

Remarks

Default value is verdana

11.7.7. Title font size

The font size for the title text.

Setting

titlefontsize

Type

integer

Possible values

Any font size

Remarks

Default value is 12

11.7.8. Title font style

The font style for the title text.

Setting

titlefontstyle

Type

string

Possible values

A combination of these values:

Value
Underline
Italic
Bold
Strikeout

An example of a combination of values:

bold italic

11.7.9. Fade in speed

The fade in speed of the notification.

Setting

speedfadein

Type

integer

Possible values

A value between 1 and 255 where 1 is slow and 255 is fast

Remarks

Default value is 10

11.7.10. Fade out speed

The fade out speed of the notification.

Setting

speedfadeout

Type

integer

Possible values

A value between 1 and 255 where 1 is slow and 255 is fast

Remarks

Default value is 10

11.7.11. Show time

The time the notification will be shown to the user. The user can close the notification when a valid close rectangle is provided. This value will make sure the notification will be closed after a specific period of time.

When the user is hovering the notification, the time will be paused so the user can keep the notification up as long as he / she wants.

Setting

showtime

Type

integer

Possible values

Any integer value, value is parsed in 100 milliseconds (1 second = 10) (30 means 30 * 100 ms = 3000 ms=3 s)

Remarks

Default value is 30 (3 seconds)

11.7.12. Transparency

The transparency of the notification.

Setting

transparency

Type

integer

Possible values

A value between 1 and 255 where 1 is fully transparent and 255 is fully visible

Remarks

Default value is 200

11.7.13. Transparent color

The color in the bitmap that should be removed from the bitmap. At all the locations where this color is located, the color will be replaced by the desktop background.

Setting

transparentcolor

Type

RGB Color

Possible values

For each color group, a number between 0 and 255 can be used

Remarks

Default color is purple (R255, G0, B255)

12. Update file

The update file contains all the actions that will be performed by Updater to update the product.

This chapter will explain all the XML codes that should be used inside the update file. It is recommended to use Updater Tool to create the update files instead of writing the XML files by hand.

12.1. XML Template

The update file must have this format:

```
<?xml version="1.0" encoding="UTF-16"?>
<UPDATEINFO>

</UPDATEINFO>
```

12.2. Constants

There are some default constants that can be used inside the update file. All constants are listed in the table below. When a value is needed which is not in the list, a custom constant can be defined.

Code	Directory
%app%	Application pathname
%appdata%	Pathname to Application Data path
%approot%	Application root folder
%commonappdata%	Pathname to Common Application Data path
%desktopallusers%	Pathname to the desktop of all users
%desktopcurrentuser%	Pathname to the desktop of current user
%ftppassword%	FTP password, only available when the FTP password is set in the settings file
%ftpusername%	FTP username, only available when the FTP username is set in the settings file
%HKCC%	HKEY_CURRENT_CONFIG
%HKCR%	HKEY_CLASSES_ROOT
%HKCU%	HKEY_CURRENT_USER
%HKLM%	HKEY_LOCAL_MACHINE
%HKUS%	HKEY_USERS
%httppassword%	HTTP password, only available when the HTTP password is set in the settings file
%httpusername%	HTTP username, only available when the HTTP username is set in the settings file
%menustartallusers%	Pathname to the start menu of all users
%menustartcurrentuser%	Pathname to the start menu of current user
%programfiles%	Pathname to System Program Files Path (same as %ProgramFiles% environment variable)
%quicklaunchallusers%	Pathname to the quick launch menu of all users
%quicklaunchcurrentuser%	Pathname to the quick launch menu of current user
%system%	Pathname to System Files Path (same as %SystemRoot% environment variable)
%temp%	Pathname to the system temp
%updater%	Pathname to Updater program

%updatertemp%	Pathname to Updater temp. All files will be downloaded to this folder. The pathname is constructed by appending \updater_temp to the system temp path
%updateserverfile%	Location of the update file on the server
%updateserverpath%	Location of the update file folder on the server
%userprofile%	Pathname to User Home directory
%windows%	Pathname to Windows directory

12.3. Custom constants

Constants can be used inside the update file to make the file more flexible. For example, it is very wise to store the web server location in a constant. When the address of the web server changes, the address only have to be changed once (which is the constant value).

The section of the constants must be defined as following:

```
<?xml version="1.0" encoding="UTF-16"?>
<UPDATEINFO>

  <CONSTANTS>
    <CONSTANT name="%site%" value="http://www.site.com"></CONSTANT>
    <CONSTANT name="%version%" value="1.0.0.1"></CONSTANT>
  </CONSTANTS>

</UPDATEINFO>
```

When using constants, *%[CONSTANT]%* must be used (percentages surrounding the constants). It is even possible to override (re-assign) the default constants, but this is not recommended.

The custom constants can only be used inside the update file. It is possible to use Updater defined constants in your constants, like:

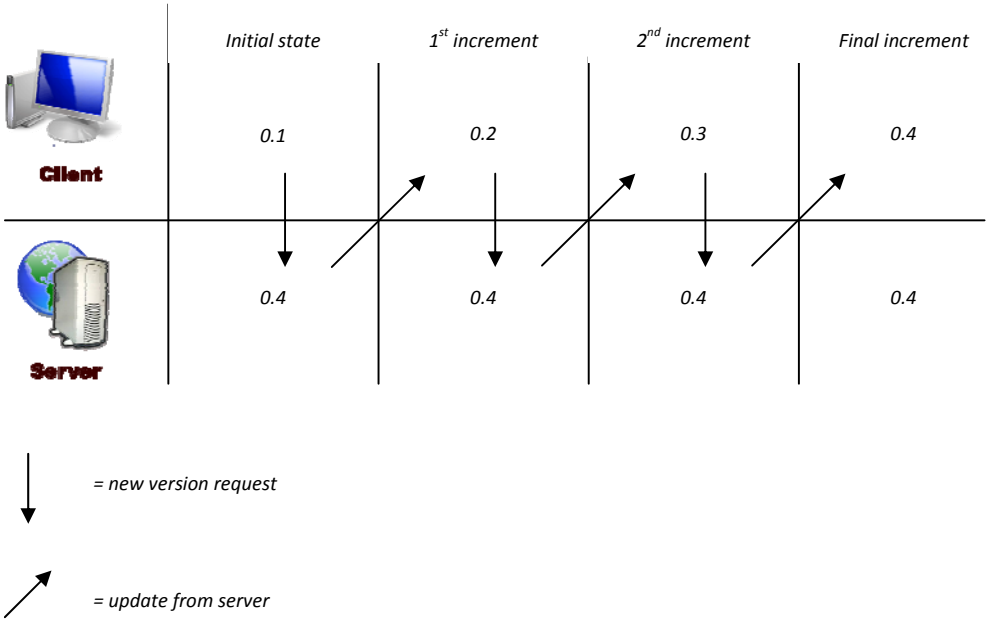
```
<CONSTANT name="%myfile%" value="%programfiles%\File.exe"></CONSTANT>
```

Object	Type	Possible values
name	%string%	Any string value
value	string	Any string value

Object	Description
name	The name of the constant as it can be used inside the update file. This value should be surrounded by %. If the % are not used, Updater will add them when parsing the update file.
value	The value of the constant.

12.4. Sequenced updating

It is possible to update a product in sequences. This means the user can be obligated to update from version 0.1 to 0.4 in sequences. An example is given by the image below:



To use sequenced updating, all update files (also the older versions) should be located on the server.

Sequenced updates should be formatted as following:

```
<?xml version="1.0" encoding="UTF-16"?>
<UPDATEINFO>

  <SEQUENCE>
    <MINIMUMVERSION>0.5.1.9</MINIMUMVERSION>
    <LOCATION>http://www.site.com/update_0.5.1.9.xml</LOCATION>
  </SEQUENCE>

</UPDATEINFO>
```

Updater will first read this part of the update file. If the current application version does not match at least the minimumversion in the update file, Updater will download the file located at the location element. This way, it is possible to create a chain of update files which will lead to the very first update which has no sequence fields or where the current application version matches the minimumversion field.

Note:

You can’t use constants in this part of the update file.

Object	Type	Possible values
minimumversion	String	Version string
location	String	Any string value

Object	Description
minimumversion	The minimum version required for the update.
location	The location of the previous update file that should be used when the user application version does not match the minimum version.

12.5. General information

This part of the update file contains all the general information needed by Updater. The general part must be formatted as following:

```
<?xml version="1.0" encoding="UTF-16"?>
<UPDATEINFO>

  <GENERAL>
    <VERSION>1.0.0.1</VERSION>
    <CLOSEAPPLICATION>false</CLOSEAPPLICATION>
    <CHECKSEPARATELY>true</CHECKSEPARATELY>
    <ENABLEROLLBACK>true</ENABLEROLLBACK>
    <LICENSE>You must agree to this license</LICENSE>
  </GENERAL>

</UPDATEINFO>
```

Object	Type	Possible values
version	string	Version string
closeapplication	boolean	True False Restart Restartsystem Restartsystemapp User
checkseparataly	boolean	True False
enablerollback	boolean	True False
showerrors	boolean	True False
selfupdate	boolean	True False
resumedownloads	boolean	True False
license	string	Any string value
forceupdate	boolean	True False

Object	Description
version	The new version of the application.
closeapplication	Use this option to close the application that will be updated: **True** the application needs to be closed **False** the application does not need to be closed **Restart** the application should close, but also start again after the update process **Restartsystem** application is closed. System should be restarted after update **Restartsystemapp** application is closed. System should be restarted after update, application should be loaded at first start up **User** the user must close the application himself
checkseparataly	If true, each file is version or date checked. Only the needed files are downloaded. Default, this value is false.
enablerollback	If true, all actions are rolled back when an error occurs or the user cancels the process. Default, this value is false.
showerrors	If true, errors will be notified to the user. If not, when an errors occurs, Updater will jump to hidden mode and finish rollback. **Note:** be aware that user will not know that an error occurred
selfupdate	If true, Updater will check if there is a new version for Updater and update itself if possible. Default, this value is true. **Note:** Updater will update itself after the update of the application.
resumedownloads	If true, Updater will resume downloads if possible. This is useful when the internet connection is interrupted when downloading large files. Default, this value is true.
license	The license that will be shown in the license step.
forceupdate	If true, user will not be able to stop the update after he starts it.

	Default, this value is false. **Note:** After each task, the eventaction startnexttask must be used to step to the next task.

12.6. Minimum requirements

This part of the update file contains all the information that is used by Updater to check if the client's computer meets the minimum requirements for the software update.

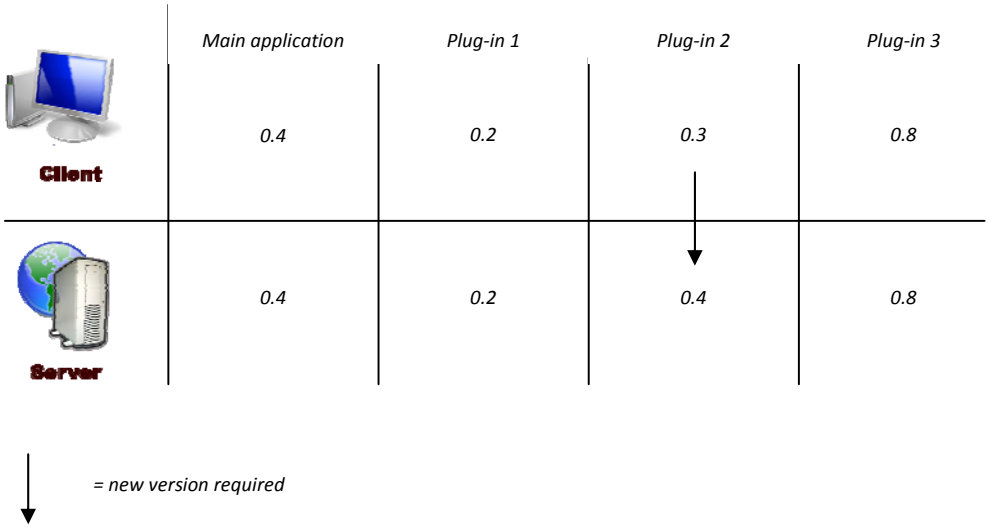
```
<?xml version="1.0" encoding="UTF-16"?>
<UPDATEINFO>

  <MINIMUMREQUIREMENTS>
     <CPU>100</CPU>
     <RAM>256</RAM>
     <VIDEOMEMORY>128</VIDEOMEMORY>
     <FREESPACE>
       <DISK type="%app%">50</DISK>
     </FREESPACE>
     <OPERATINGSYSTEMS>
       <OS type="95" supported="false" />
       <OS type="98" supported="false" />
       <OS type="ME" supported="false" />
       <OS type="NT3" supported="false" />
       <OS type="NT4" supported="false" />
       <OS type="2000" supported="true" />
       <OS type="XP" supported="true" />
       <OS type="2003" supported="true" />
       <OS type="Vista" supported="true" />
     </OPERATINGSYSTEMS>
  </MINIMUMREQUIREMENTS>

</UPDATEINFO>
```

Object	Type	Possible values
cpu	integer	Any integer value
ram	integer	Any integer value
videomemory	integer	Any integer value
freespace	Contains disk elements	Disk elements
disk – type	string	Any string value
disk	integer	Any integer value
operatingsystems	Contains os elements	Os elements
os – type	string	Any OS (supported os are included in sample above)
os – supported	boolean	True False

Object	Description
cpu	The minimum speed of the CPU in Mhz.
ram	The minimum amount of RAM in MB.
videomemory	The minimum amount of video memory in MB.
freespace	This element holds disk elements.
disk – type	The disk type: **Disk name** the name of the disk that should have enough disk space (for example c:\) **Path** the path of a file or folder. The disk will automatically be detected (for example C:\Program files\MyApplication or %app%).
disk	The minimum amount of disk space required on the specific disk in MB.
operatingsystems	This elements holds os elements.
os – type	The type of the os. Supported types: 95 98 ME NT3 NT4 2000 XP 2003 Vista
os – supported	True if OS is supported by the update, false if not. Default, this value is true for all operating systems.

12.7. Sections

Sections can be very handy when using different parts in an application. The image below shows an example of a situation where sections can be used:



It is possible to let the user decide which sections to update and which should not be updated. It is also possible to obligate the user to install one or more sections. The user can select sections in the select update step.

Sections can be added to the update file using the following formatting:

```
<?xml version="1.0" encoding="UTF-16"?>

  <SECTIONS>
    <SECTION name="main" title="Main application files" version="1.0"
             check="true" enabled="true" description="The main
             application files for application MyApplication"
             url="www.site.com/sections/main_info.html">
    </SECTION>
  </SECTIONS>

</UPDATEINFO>
```

Object	Type	Possible values
name	string	Any string value
title	string	Any string value
check	boolean	True

		False
version	string	Version string
enabled	boolean	True False
description	string	Any string value
url	string	Any string value

Object	Description
name	The name of the section as it can be used inside the update file to add a file or action to a specific section.
title	The title of the section as it will be available for the user.
check	If true, the section will be checked in the tree control where the user can select the sections to update. Default, this value is true.
version	The newest version of the section.
enabled	If true, the section will be enabled so the user can check the update state of the section. Default, this value is true.
description	The description of the section. The user will be able to read the description when he / she clicks on the section in the tree control.
url	When the description is too short, this url can be used to link the user to a website with more information about the section. When this value is used, Updater will add a link to the section description.

12.8. Features

When there is a new update available, Updater can show some of the most important features to the user.

When features are used, the following formatting must be used:

```
<?xml version="1.0" encoding="UTF-16"?>
<UPDATEINFO>

  <FEATURES>
    <FEATURE>
      <TEXT>Feature 1</TEXT>
      <LINK>http://www.catenalogic.com/product/feature1.html</LINK>
    </FEATURE>
    <FEATURE>
      <TEXT>Feature 2</TEXT>
    </FEATURE>
  </FEATURES>

</UPDATEINFO>
```

It is possible to enable the user to read more information about a specific feature. This can be done by using the link object inside the feature element. When only the text object is used, the feature will be visible, but not clickable.

Note: The maximum amount of features that can be added to the update file is 5.

Object	Type	Possible values
text	string	Any string value
link	string	Any string value

Object	Description
text	The text of the new feature. Keep it short and simple.
link	The url of the location where the user can read some detailed information about the feature.

12.9. Files

Updater works with files which need to be updated. Each file can contain one or more file actions which makes sure the file will get updated.

To insert the general files section, use the following formatting:

```
<?xml version="1.0" encoding="UTF-16"?>
<UPDATEINFO>

  <FILES>
  </FILES>

</UPDATEINFO>
```

Each file must at least use the following formatting:

```
<FILES>
  <FILE name="CD Database" location="%app%\CD Database.exe"
        version="0.1.0.0" check="version">
  </FILE>
</FILES>
```

Object	Type	Possible values
name	string	Any string value
check	string	Date Datetime Version Hash
location	string	Any string value
version	string	Version string
date	date / time	Any date / time value
hash	MD5 file hash	Any MD5 file hash
section	string	Any existing section

Object	Description
name	The name of the file as it will be used for progress updates to the user.
check [1]	The way Updater should check this file. **Date** The files will be compared using the date of the files. The date is formatted as *yyyy-mm-dd*. **Datetime** The files will be compared using the date and time of the files. The date and time is formatted as *yyyy-mm-dd/hh:mm:ss* **Version** The files will be compared using the version of the files. **Hash** The files will be compared using the MD5 hash of the files.
location [2]	The location of the file on the client pc. The location of the file is used for the calculation of the current date, date / time, version or hash of the file.
version [3]	The newest version of the file.
date [4]	The newest date or date / time of the file. When this value is empty, Updater will automatically retrieve the date / time of the file on the internet.
hash [5]	The MD5 file hash of the newest file.
section [6]	The name of the section where this file belongs to.

[1] *This field is only required when checkseparately is used.*

[2] *This field is only required when check value is not empty.*

[3] *This field is only required when check value is **version**.*

[4] *This field is only required when check value is **date** or **datetime**.*

[5] *This field is only required when check value is **hash**.*

[6] *This field is only required when sectons are used and this field must belong to a specific section.*

There are several file actions supported in Updater. Each action is explained in this chapter and each action will be briefly explained.

Each action must use the following formatting:

```
<FILE ...>
  <ACTION type="actiontype"></ACTION>
</FILE>
```

Note: All actions are performed in the order they appear in the update file.

Object	Type	Possible values
type	string	File action type

Object	Description
type	The file action type. Each action explained in this chapter contains the action type to use for this field.

12.9.1. Download

This action can be used to download a file to the pc of the user. The file will be downloaded to the *%updatertemp%* folder.

It is possible to add mirrors (multiple download locations) by adding more location fields to this action. It is wise to add the fastest servers at the top. If the file does not exist on the server, or the server is unreachable, Updater will use the next download location. There is a maximum of 100 download locations for each download action. The locations are used in the order they appear in the updater file.

Some servers require authentication. To identify a user to the server, use the *%[protocol]username%* and *%[protocol]password%* variables in the location value.

The following formatting must be used:

```
<FILE ...>
  <ACTION type="download">
    <DESTINATION>setup/cddb.exe</DESTINATION>
    <LOCATION>%primarysite%/setup/cddb.exe</LOCATION>
    <LOCATION>%mirror1%/setup/cddb.exe</LOCATION>
  </ACTION>
</FILE>
```

Object	Type	Possible values
destination	string	Any string value
location	string	Any string value

Object	Description
destination	The destination of the download action. There are three ways to use this field: **1) Leave it empty** Updater will strip the filename from the download location and use it as download destination. The example will be downloaded to *%updatertemp%\cddb.exe*. **2) Relative path** Updater will add this path to the %updatertemp% folder. For example, when *test* is used, Updater will download the example to *%updatertemp%\test\cddb.exe*. **3) Full path** Updater will use this full path to store the file. **Note:** It is not possible to use a filename to specify a different

	filename in the destination, so only a folder is allowed.
location	The location of the file on the internet. Updater will use the filename to store it on the hard-disk. For example, *http://www.site.com/files/myfile.exe* will be downloaded to *%updatertemp%\myfile.exe*. It is possible to add multiple location fields to add mirrors. Updater will randomly choose which mirror to use to download each file.

12.9.2. Copy

This action can be used to copy a file.

The following formatting must be used:

```
<FILE ...>
  <ACTION type="copy">
    <OLDLOCATION>%updatertemp%\cddb.exe</OLDLOCATION>
    <NEWLOCATION>%app%\cddb.exe</NEWLOCATION>
    <OVERWRITE>true</OVERWRITE>
  </ACTION>
</FILE>
```

Object	Type	Possible values
oldlocation	string	Any string value
newlocation	string	Any string value
overwrite	boolean	True False

Object	Description
oldlocation	The old location of the file.
newlocation	The new location of the file.
overwrite	If true, Updater will overwrite the file when it already exists. If false, Updater will not overwrite the file when it already exists. Default, this value is false.

12.9.3. Delete

This action can be used to delete a file.

The following formatting must be used:

```
<FILE ...>
  <ACTION type="delete">
    <LOCATION>%updatertemp%\cddb.exe</LOCATION>
    <ASKUSER>true</ASKUSER>
  </ACTION>
</FILE>
```

Object	Type	Possible values
location	string	Any string value
askuser	boolean	True False

Object	Description
location	The location of the file that should be deleted.
askuser	If true, Updater will ask the user to confirm the deletion of the file. If false, Updater will delete the file without asking the user for confirmation. Default, this value is true.

12.9.4. Execute

This action can be used to execute a file.

The following formatting must be used:

```
<FILE ...>
  <ACTION type="run">
    <LOCATION>%app%\cddb.exe</LOCATION>
    <PARAMETERS>-p "c:\program files\" -s</PARAMETERS>
    <WAIT>true</WAIT>
  </ACTION>
</FILE>
```

Object	Type	Possible values
location	string	Any string value
parameters	string	Any string value
wait	boolean	True False

Object	Description
location	The location of the file that should be executed.
parameters	The parameters that should be used when executing the file. This value is not required.
wait	If true, Updater will wait until the execution is finished and then continue updating. If false, Updater will execute the file and continue updating immediately. Default, this value is true.

12.9.5. Unzip

This action can be used to unzip a file.

The following formatting must be used:

```
<FILE ...>
  <ACTION type="unzip">
    <LOCATION>%updatertemp%\MyZipFile.zip</LOCATION>
    <DESTINATION>%app%\MyZipFolder\</DESTINATION >
  </ACTION>
</FILE>
```

Object	Type	Possible values
location	string	Any string value
destination	string	Any string value

Object	Description
location	The location of the zip file that needs to be unzipped.
destination	The destination folder where the files will be unzipped to.

12.9.6. Set File Attributes

This action can be used to set the file attributes for a file or folder.

The following formatting must be used:

```
<FILE ...>
  <ACTION type="setfileattr">
    <LOCATION>%updatertemp%\MyZipFile.zip</LOCATION>
    <ATTRIBUTES>system readonly</ATTRIBUTES>
  </ACTION>
</FILE>
```

Object	Type	Possible values
location	string	Any string value
attributes [1]	File or folder attribute	Hidden Readonly System Archive Compressed

Object	Description
location	The location of the file or folder that will get the new attributes.
attributes	One or more attributes for the file or folder.

[1] *A combination of several attributes is also possible.*

12.9.7. Register

This action can be used to register a file. This action can be very useful when a DLL file or OCX file needs to be registered.

The following formatting must be used:

```
<FILE ...>
  <ACTION type="register">
    <LOCATION>%app%\MyControl.ocx</LOCATION>
    <ASKUSER>true</ASKUSER>
  </ACTION>
</FILE>
```

Object	Type	Possible values
location	string	Any string value
askuser	boolean	True False

Object	Description
location	The location of the file that needs to be registered.
askuser	If true, Updater will ask if the user wants to register the file. If not, Updater will register the file without asking the user for confirmation. Default, this value is true.

12.10. Registry

This part alters or creates registry entries.

When the registry must be altered, the following formatting must be used:

```
<?xml version="1.0" encoding="UTF-16"?>
<UPDATEINFO>

  <REGISTRY>
    <ITEM type="string">
      <KEY>HKEY_LOCAL_MACHINE\SOFTWARE\CD Database\Version</KEY>
      <VALUE>0.0.0.1</VALUE>
    </ITEM>
  </REGISTRY>

</UPDATEINFO>
```

Object	Type	Possible values
type	registry entry type	String Binary Dword
key	string	Registry key
value	string	Any string value

Object	Description
type	The type of the registry entry. Default, this value is string.
key	Complete key of the registry. Updater will split this value into separate parts itself
value	The value of the registry entry.

12.11. Ini

This part alters or creates ini-file entries.

When an ini-file must be altered, the following formatting must be used:

```
<?xml version="1.0" encoding="UTF-16"?>
<UPDATEINFO>

  <INI>
    <ITEM>
      <LOCATION>%app%\settings.ini</LOCATION>
      <GROUP>app_info</GROUP>
      <KEY>version</KEY>
      <VALUE>0.0.0.1</VALUE>
    </ITEM>
  </INI>

</UPDATEINFO>
```

The object names are name as following:

```
[GROUP]
Key=value
```

Object	Type	Possible values
location	string	Any string value
group	string	Any string value
key	string	Any string value
value	string	Any string value

Object	Description
location	The location of the ini-file that contains the entry that should be altered.
group	Group in the ini-file.
key	The key in the ini-file.
value	The value of the key in the ini-file.

12.12. Validation

It is possible to check if the software is updated successfully. This is very useful when an external installer is launched and Updater needs to check if the update is installed correctly.

When validation is required, the following formatting must be used:

```
<?xml version="1.0" encoding="UTF-16"?>
<UPDATEINFO>

  <VALIDATION>
    <FILE location="%app%\MyApp.exe"
          hash="8e2fa1048065bb493986bb8dd43e20f2" />
  </INI>

</UPDATEINFO>
GROUP]
Key=value
```

Object	Type	Possible values
location	string	Any string value
hash	string	File hash

Object	Description
location	The location of a file located on the client's pc.
hash	Hash of the file as it should be.

12.13. Events

Events are a very powerful extension to Updater. To enable events, use the following formatting:

```
<?xml version="1.0" encoding="UTF-16"?>
<UPDATEINFO>

  <EVENTS>
  </EVENTS>

</UPDATEINFO>
```

12.13.1. Supported events

These events are supported:

Name	Fired when / Description
OnNewVersion	Fired when a new version is found.
OnNoNewVersion	Fired when no new version is found.
BeforeMinimumRequirements	Fired before Updater checks the minimum requirements
AfterMinimumRequirements	Fired after Updater has checked the minimum requirements
BeforeHtmlMessage	Fired before the user sees the html message.
AfterHtmlMessage	Fired after the user leaves the html message.
BeforeLicense	Fired before the user accepts the license agreements.
AfterLicense	Fired after the user accepted the license agreements.
BeforeProtection	Fired before the software protection starts.
AfterProtection	Fired when the software protection is finished successful.
BeforeSelectUpdate	Fired before the user can select the sections to update.
AfterSelectUpdate	Fired after the user has selected the sections to update.
BeforeDownload	Fired before files are downloaded.
AfterDownload	Fired after files are downloaded.
BeforeInstall	Fired before files are installed.
AfterInstall	Fired after files are installed.
BeforeValidate	Fired before the validation of the update.
AfterValidate	Fired after the validation of the update.
BeforeRollback	Fired before changes are rolled back.
AfterRollback	Fired after changes are rolled back.
OnClose	Will be fired only after a successful update of the application. This way, it is possible to execute files which will overwrite the Updater executable. Keep in mind that if the user cancels this process, no rollback will be performed.

12.13.2. Supported event actions

There are several event actions supported in Updater. Each action is explained and each action will contain a small example.

Each event action must at least use the following formatting:

```
<?xml version="1.0" encoding="UTF-16"?>
<UPDATEINFO>

  <EVENTS>
    <ACTION type="actiontype"></ACTION>
  </EVENTS>

</UPDATEINFO>
```

Object	Type	Possible values
type	string	Event action type

Object	Description
type	The event action type. Each action explained in this chapter contains the action type to use for this field.

12.13.2.1. Close

This action will close Updater immediately. The user will not be asked for any confirmation.

The following formatting must be used:

```
<EVENTS>
  <ACTION type="close"></ACTION>
</EVENTS>
```

12.13.2.2. Skip next task

This action can be used to skip tasks. For example, the install step can be skipped when only files need to be downloaded and executed when Updater quits.

The following formatting must be used:

```
<EVENTS>
  <ACTION type="skipnexttask"></ACTION>
</EVENTS>
```

12.13.2.3. Set mode

This action switches the run mode of Updater at runtime.

The following formatting must be used:

```
<EVENTS>
  <ACTION type="setmode">
    <MODE></MODE>
  </ACTION>
</EVENTS>
```

Object	Type	Possible values
mode	Run mode	Full Silent Verysilent Hidden

Object	Description
mode	The mode where Updater will change to. See chapter modes for more information about the modes available in Updater.

12.13.2.4. Execute file

This action can be used to execute a file at an event. Updater will not wait until the application closes again, but will continue immediately.

The following formatting must be used:

```
<EVENTS>
  <ACTION type="run">
    <LOCATION>%updatertemp%\setup.exe</LOCATION>
    <PARAMETERS></PARAMETERS>
    <WAIT>true</WAIT>
  </ACTION>
</EVENTS>
```

Object	Type	Possible values
location	string	Any string value
parameters	string	Any string value
wait	boolean	True False

Object	Description
location	The location of the file that should be executed.
parameters	The parameters that should be used when executing the file. This value is not required.
wait	If true, Updater will wait until the execution is finished and then continue updating. If false, Updater will execute the file and continue updating immediately. Default, this value is false.

12.13.2.5. Start next task

This action can be used to start next tasks. This can be useful when the installation should start immediately after downloading the files. This way, full mode can be automated.

The following formatting must be used:

```
<EVENTS>
  <ACTION type="startnexttask"></ACTION>
</EVENTS>
```

Note: Only use this action in full mode since the other modes are already automatic.

12.13.2.6. Close application

This action can be used to close other applications, for example internet explorer windows. This action can also be very useful when an application consists of multiple executables. Since the restartapplication option can only close only one executable, this action can be very useful.

The following formatting must be used:

```
<EVENTS>
  <ACTION type="closeapplication">
    <TITLE>Internet Explorer</TITLE>
    <FILENAME>iexplore.exe</FILENAME>
    <ASKUSER>true</ASKUSER>
  </ACTION>
</EVENTS>
```

Object	Type	Possible values
title	string	Any string value
filename	string	Any string value
askuser	boolean	True False

Object	Description
title	The title that will be used to inform the user about the application that is going to be closed. For example, use Internet Explorer for iexplore.exe.
filename	The filename of the application that must be closed. This value must contain a full path, but some applications (such as iexplore.exe) can be closed without a path.
askuser	If the user should be asked for confirmation, this value must be true. If false, Updater will close the application immediately without asking. Default, this value is true.

12.13.2.7. Show notifier

This action can be used to show the custom notifies with custom text. The notifier will override the default Updater messages.

The following formatting must be used:

```
<EVENTS>
  <ACTION type="shownotifier">
    <TITLE>Custom notification</TITLE>
    <DESCRIPTION>This is the description</DESCRIPTION>
  </ACTION>
</EVENTS>
```

Object	Type	Possible values
title	string	Any string value
description	string	Any string value

Object	Description
title	The title that will be used for the notification. Keep it short and simple.
description	The description that will be used for the description. It is possible to use HTML tags inside this value to format the text inside the description.

12.14. Popup menu

When in silent mode, it is possible to create a popup menu. The user will be able to use the menu by right-clicking on the tray-icon when Updater is finished updating the application.

By default, the popup menu will only have two entries:

About Updater…

Exit

The popup menu can be customized by adding custom actions and icons. To customize the popup menu, the following formatting must be used:

```
<?xml version="1.0" encoding="UTF-16"?>
<UPDATEINFO>

  <POPUPMENU>
    <ITEM>
      <TEXT>Launch readme</TEXT>
      <LOCATION>c:\windows\notepad.exe</LOCATION>
      <PARAMETERS>%app%\Readme.txt</PARAMETERS>
      <BITMAP>http://www.catenalogic.com/bitmap1.bmp</BITMAP>
    <ITEM>
  </POPUPMENU>

</UPDATEINFO>
```

Object	Type	Possible values
text	string	Any string value
location	string	Any string value
parameters	string	Any string value
bitmap	string	Any string value

Object	Description
text	Text of the popup menu item. **Note:** Use a – when a separator must be added to the menu.
location	The location of the file that should be executed when the user clicks on the popup menu item.
parameters	The parameters that should be used when executing the file. This value is not required.
bitmap	The location of the bitmap. Updater will download this file and use it in the popup menu item.

	When this value is not used, Updater will not show any image before the popup menu item. The image should be 14 x 14 pixels (h x w).

12.15. HTML message

It is possible to show some advertisement to the user about new or existing products or events. The user will only see this HTML message when running in full mode.

When a HTML message should be enabled in Updater, the following formatting must be used:

```
<?xml version="1.0" encoding="UTF-16"?>
<UPDATEINFO>

  <HTMLMESSAGE>
    <URL>http://www.catenalogic.com</URL>
    <TIME>3</TIME>
  </HTMLMESSAGE>

</UPDATEINFO>
```

Object	Type	Possible values
url	string	Any string value
time	integer	0 to 60

Object	Description
url	The url where the HTML page is located which contains the HTML message. The width of the HTML page must be 391 pixels to make sure no horizontal scrolling will be enabled. Be sure to use *target="_blank"* when using links inside the HTML message so the user will be linked to a new window with more information.
time	The minimum time the user must wait before continuing the update process. The user will see a count down in the next button so he / she can see how long it takes before he / she can continue.

12.16. Shortcuts

Updater is able to create shortcuts on the client pc.

The following formatting must be used when creating shortcuts:

```
<?xml version="1.0" encoding="UTF-16"?>
<UPDATEINFO>

  <SHORTCUTS askuser="true">
    <SHORTCUT>
      <TARGETFILE>%app%\MyApplication.exe</TARGETFILE>
      <WORKINGDIRECTORY>%app%</WORKINGDIRECTORY>
      <PARAMETERS>-s</PARAMETERS>
      <LINKFILE>%desktopallusers%\MyApplication.lnk</LINKFILE>
      <ICONFILE>%app%\img\MyIcon.ico</ICONFILE>
      <ICONINDEX>1</ICONINDEX>
      <SHOWMODE>SW_SHOWNORMAL</SHOWMODE>
      <DESCRIPTION>MyApplication shortcut</DESCRIPTION>
    </SHORTCUT>
  </SHORTCUTS>

</UPDATEINFO>
```

Note: Don't forget to take a look at the constants which can be very useful when creating shortcuts.

Object	Type	Possible values
targetfile	string	Any string value
workingdirectory	string	Any string value
parameters	string	Any string value
linkfile	string	Any string value
iconfile	string	Any string value
iconindex	integer	≥ 0
showmode	Show mode value	SW_SHOWNORMAL SW_SHOWMAXIMIZED SW_SHOWMINNOACTIVE
description	string	Any string value

Object	Description
targetfile	Location of the file that should be executed when the user uses the shortcut.
workingdirectory	The working directory of the executable.
parameters	The parameters that should be used to execute the application when the user uses the shortcut.
linkfile	Location of the link file. This value defines where the shortcut will be located, for example the desktop or quick launch menu.
iconfile	The location of the file that contains the icon that must be used to show in the shortcut. Only use this value when the executable does not contain the right icon itself.
iconindex	The index of the icon that must be used. This is needed when a DLL or executable contains more than one icon.
showmode	The mode in which the application must be launched when the user uses the shortcut.
description	The description that will be shown to the user when the user hovers the shortcut.

12.17. Software protection

Software protection is a very important feature of Updater.

There are two ways of protection. They can be used separate, but also together:

1. License check

File hashing

To enable software protection, the following formatting must be used:

```
<?xml version="1.0" encoding="UTF-16"?>
<UPDATEINFO>

  <PROTECTION>
    <BUYLICENSE>
      <WEBSITE>http://www.catenalogic.com</WEBSITE>
    </BUYLICENSE>
  </PROTECTION>

</UPDATEINFO>
```

Object	Type	Possible values
website	string	Any string value

Object	Description
website	The website where the user can get a valid license when the software protection fails. This website is not shown when the software protection succeeds or when this value is empty.

12.17.1. License check

The license check can be used by creating a DLL with at least one function with header:

```
bool CheckLicense();
```

Updater will call this function. When the function returns true, Updater will consider the license as valid. Any code can be inserted in this DLL, so any license can be checked this way.

The DLL will be downloaded from a website to prevent that hackers will edit the file. The file will be deleted immediately after the check, and will not remain on the users system. This is a very secure way of license checking.

Add this code to the update file to enable the license check:

```
<PROTECTION>

    <REGISTRATION>
      <LOCATION> http://wwww.catenalogic.com/license.dll</LOCATION>
    </REGISTRATION>

</PROTECTION>
```

Note: There is an example of a protection DLL inside the Updater compiled project.

Object	Type	Possible values
location	string	Any string value

Object	Description
location	The location where the DLL is located on the internet. Updater will download the DLL from this location to check the license.

12.17.2. File hashing

File hashing can check if a specific file is replaced by a different file. Since this is a very common way of software cracking, Updater supports file hash checking.

To enable file hashing, the following formatting must be used:

```
<PROTECTION>

    <HASHING>
      <HASH file="%app%\MyApp.exe" hash="b03900cf3fbc1f2c... ">
    </HASHING>

</PROTECTION>
```

Note: There is no specified limit to the amount of file hashes.

Object	Type	Possible values
file	string	Any string value
hash	MD5 file hash	Any MD5 file hash

Object	Description
file	The filename that should be hash checked.
hash	The hash that the file should have. **Note:** Keep in mind the user will still have the old version of the file, so the hash of the old version file should be used here.

13. Section file

This file is located on the hard-disk of the client user. This file is used to store all the version information used by sections by that specific user.

If this file is not available on the client's computer, it will install all the updates the user selects and store the newest version in a new generated file.

This file must be named as sections.xml and must be located in the same folder as the executable for Updater.

The file is formatted as followed:

```
<?xml version="1.0" encoding="UTF-16"?>
<SECTIONS>
  <SECTION name="[sectionname]" version="[section version]" />
</SECTIONS>
```

14. Self update file

This file is located at a server of CatenaLogic. This file is used to check for a new version for Updater itself.

Updater will check if a new version is available when self-update is enabled in the update file. When a new version is available, Updater will download the new version.

Updater will not handle a self-update as a real update. This means Updater will download a new version of itself only when there is a new version available of the product that uses Updater too.

This file can, but only should, be hosted on a different location when a custom version of Updater is used. In that case, use the following formatting for the file:

```
<?xml version="1.0" encoding="UTF-16"?>
<SELFUPDATE>
  <VERSION>1.0.0.1</VERSION>
  <LOCATION>http://www.site.com/bin/updater/updater.exe</LOCATION>
  <LANGUAGEPACK>
    <VERSION>0.8.1.1</VERSION>
    <LOCATION>http://www.site.com/bin/updater/languages.zip</LOCATION>
  </LANGUAGEPACK>
</SELFUPDATE>
```

15. User interface

15.1. Customizing

This chapter will explain how to customize the user interface of Updater.

All modes are able to change the way hyperlinks are showed. The colors of the hyperlinks in Updater can be defined in the settings file.

15.1.1. Full mode

Full mode can be customized by using custom wizard images. There are two images used when running in full mode.

1) Large image

This size of the image must be 314 x 164 pixels (h x w). The default logo looks like this:

Figure 15-1 Default large image

2) Small image

This size of the image must be 58 x 435 pixels (h x w). The default logo looks like this:

Figure 15-2 Default small image

15.1.2. Silent mode

Silent mode can be customized by the custom notifies which can be defined in the settings file. There is a tool (UpdaterSettings) included in the Updater compiled package which can be useful when designing custom notifies.

15.2. Icon explanation

15.2.1.1. Download icon

This icon represents downloading.

15.2.1.2. Install icon

This icon represents installing.

15.2.1.3. Succeeded

This icon represents a successful update process.

15.2.1.4. Not succeeded

This icon represents an update process that is not succeeded.