# CoexEdit 1.2

# User Manual

November 22, 2005

Genii Software Ltd.
http://www.GeniiSoft.com

# Table of Contents

# 1

# Overview

*This chapter will discuss what* CoexEdit *is, what business problems it is designed to solve, and how it goes about solving them.*

## Description

CoexEdit is a specialized software extension that runs on the IBM Lotus Domino server, and optionally on the IBM Lotus Notes client, using IBM's "extension manager" technology to seamless editing integration between the IBM Lotus Notes client and various third party rich text editors which run in a web browser. No scripts or agents are required.

## Purpose

CoexEdit was developed to allow users to edit rich text fields in both the Notes client and in a web browser without significant loss of fidelity when switching between them for editing or viewing.

CoexEdit works by dynamically converting between Notes rich text and HTML/XHTML on demand, with specific attention to storage of shared elements so that loss in formatting and content is minimized. Graphics, tables, attachments and links may be edited on either the Notes client or on the web, depending on which third party web editor is used. CoexEdit simply makes sure the conversion is as seamless as possible and minimizes the loss usually encountered in such a conversion.

## Compatibility

CoexEdit 1.2 on the server supports all point releases of IBM Lotus Domino R5, ND6, ND6.5 and ND7 on Windows and AIX, with Linux supported planned for the near future. Other platforms will be supported if demand warrants their addition.

CoexEdit 1.2 on the client is available for IBM Lotus Notes R5, ND6, ND6.5 and ND7 on Windows directly and on the Mac indirectly through the server. A direct Mac client version is under investigation.

CoexEdit 1.2 in the web browser supports a variety of third party rich text editors, and can be easily adapted to support others based on a set of configurable characteristics. Browser support is mostly dependent on the support necessary for

the rich text editor, but generally requires Internet Explorer 5.5+, Firefox 1.1+, Mozilla 1.3+ and Netscape 7+.

## Just in time conversion

To limit the inevitable small losses created by converting between two formats, CoexEdit only converts from rich text to HTML and back when the conversion is needed. If a document is mostly edited and accessed through the Notes client, the conversion to HTML does not need to happen each time, but will be done just before the web browser needs to display the HTML.

## Detection & Optimization

CoexEdit will detect whether a field should be converted based on trigger conditions. This allows CoexEdit to identify changes made and replicated into the database as well as those made on the client. It also allows a variety of configurations depending on the needs of an organization, with more or less of the effort placed on the server or the client, for example.

## Security

CoexEdit fully respects and follows the existing Lotus Notes/Domino security model. CoexEdit works with the privileges and security level of the Lotus Domino server ID or Lotus Notes user ID it is running on.

# 2

# Installation

*This chapter will discuss exactly how to install and configure CoexEdit to run on your Domino server. It includes examples of the most common configuration entries for your NOTES.INI file. Additional information on configuration options can be found in Chapter 3 – Configuration Options. Information on monitoring the activity of CoexEdit, including debugging and tracing options, can be found in Chapter 4 – Monitoring and Debugging.*

## Preparing for Installation

In order to install and configure CoexEdit, you will need both the software and the software license. The CoexEdit software will be available from the Genii Software website from the CoexEdit Download page at http://www.GeniiSoft.com/showcase.nsf/CoexEditDownloads, while the CoexEdit license, whether an evaluation license or the production license, must be obtained from Genii Software or a reseller. Be sure that you have the appropriate software for your Domino server's operating system and Domino release.

## 1. Stop the Domino server

You cannot modify the Notes.INI information on a Domino server without first shutting down the server or stopping the Domino service. The same rule applies for upgrades to CoexEdit.

You will need to edit your Notes.INI file to add and modify some options. Open the Notes.INI file and proceed to the next step.

## 2. Copy the software and license into the program directory

Copy the CoexEdit software, named **nCoexEd.dll** for Windows systems, **libcoexed.a** for AIX, and eventually **libcoexed.so** for Linux, into the same directory as the Domino server software. The license file, named **coexedit.lic** for all operating systems, should be copied into the same directory as the software.

You will need to edit your Notes.INI file to add and modify some options. Open the Notes.INI file and proceed to the next step. Steps for modifying the Notes.INI information on different server platforms can be found in the appropriate Domino server documentation.

## 3. Modify the "extmgr_addins=" parameter

Search the Notes.INI file for the **extmgr_addins=** parameter. If it does not exist, add it at the end of the Notes.INI file, but *always make sure there is a blank line after the last entry in the Notes.INI file*. Without this blank line, the final entry may be ignored. The line added should be this on Windows:

**extmgr_addins=CoexEd**

or this on AIX,

**extmgr_addins=coexed**

or possibly, on a Linux system,

**extmgr_addins=coexed.so**

You may find there is already an **extmgr_addins=** parameter, as other custom add-ins also use this technology. In this case, simply add CoexEd to the parameter as a comma delimited list, so that

**extmgr_addins=app1,app2**

becomes

**extmgr_addins=app1,app2,CoexEd**

There can be any number of applications included in this list. While it is not required that CoexEd be last, it is generally recommended.

## 4. Add additional CoexEdit parameters

See the Configuration Options section for a list of additional CoexEdit parameters which can be set in the Notes.INI.

## 5. Restart the Domino server

When you have finished making changes to the Notes.INI information on the Domino server, and made sure they are saved, you can safely restart the Domino server or Domino service. If the CoexEdit software is properly loaded, you will see a few messages along the lines of:

```
11/03/2005 11:06:47 AM  CoexEdit (server): Loaded into 'SERVER' process
11/03/2005 11:06:48 AM  CoexEdit Version 1.2
11/03/2005 11:06:48 AM  Copyright (c) 2005 Genii Software Ltd., All Rights Reserved
```

# 3

# Configuration Options

*This chapter will describe each standard configuration option for CoexEdit, what values it may contain, and the impact of each value. The monitoring and debugging options are covered separately in Chapter 3.*

## NOTES.INI options

The only required change to the NOTES.INI file is the setting of the **extmgr_addins** line described in section 2. There are a small number of additional configuration options available.

**CoexEditEditor=editor** (FCKEditor, TinyMCE, Xinha or EditLive in Version 1.2)

Setting the **CoexEditEditor=Xinha** (for example) is the equivalent of setting the **$CoexEditProperties** field to "CoexEditor='Xinha' ", but it allows you to leave that field off the form entirely. This parameter is useful if only a single rich text editor is to be used in your environment. There should not be quotes around the editor name.

**CoexEditProperties=properties**

Setting the **CoexEditProperties** value is the equivalent of setting the entire **$CoexEditProperties** field, which may be useful if you are using a non-standard web editor (i.e., not FCKEditor, TinyMCE, Xinha or EditLive for Version 1.2) and have set individual properties directly. Again, it is useful if you have uniform requirements on your server and want to leave the **$CoexEditProperties** field off the form or subform.

**CoexEditServerTasks=tasks** (HTTP or SERVER, both default to on)

By default, both the HTTP process and the SERVER process use CoexEdit, which means that updates done through the web are handled by the HTTP process while updates to a server based database done from the Notes client are handled through the SERVER process. In some circumstances where performance is a consideration, it may be desired to set this to **CoexEditServerTasks=HTTP** so that web updates will be handled by the HTTP process, but the client based CoexEdit process will handle all updates to both local and server based databases from the Notes client. This is not normally advisable, but may be necessary on very performance intensive or underpowered servers.

# 4

# Implementation

*This chapter will describe each standard configuration option for CoexEdit, what values it may contain, and the impact of each value. The monitoring and debugging options are covered separately in Chapter 3.*

## Option 1 - Subforms

The most popular way to implement CoexEdit is by adding subforms that separate out the Notes client use and the web use. These subforms can then contain the standard CoexEdit fields defined below, as well as additional fields and events which implement the specific web editor. For example, in the samples we ship with CoexEdit 1.2, the subforms contain a computed field called HTMLBodyContent which can then be referred to in the HTMLBodyContent formula of the form itself. In addition, the JavaScript Header for the subform can be used to supply the Javascript used to launch or configure most web editors.

The subforms are usually paired so that there is one version for Notes and the other for the web for each form. In the FCKEditor with CoexEdit sample, there are four subforms used in this way. The pair **FCKEditorSubform** and **NotesClientSubform** are used by the **Example Form**, which only has a single rich text field. The pair **MultiFCKEditorSubform** and **MultiNotesClientSubform** are used by the **Multi-Field Example Form**, which has three separate rich text fields that are handled by separate editor instances. If different forms were used which use the same field names, these could be used repeatedly by the different forms (e.g., a **Body** and **BodyWeb** field might be used by a discussion database in both the **Main Topic** and **Response** forms).

## Option 2 - Forms

The second most popular way to implement CoexEdit is by simply creating two forms that separate out the Notes client use and the web use. The forms themselves then contain the standard CoexEdit fields defined below, as well as additional fields and events which implement the specific web editor. The best reasons for this approach is to simplify the implementation, since the HTMLBodyContent formula and the JavaScript Header can be used directly, and there is no need for additional design elements. This is often a good approach if it is only desired to use CoexEdit with a single form and the least impact on the database is required.

## Standard CoexEdit fields

The **$CoexEdit** field should always compute to "Web" when the document is saved on the web, and to "Notes" when the document is saved from the Notes client. In Notes 6 and above, the formula can be set to @ClientType, but in R5 it would need to check the roles to determine whether "Web" or "Notes" should be computed.

The **$CoexProperties** field is set to "CoexEditor='FCKEditor' " or "CoexEditor='TinyMCE' " or "CoexEditor='Xinha' ", but the field could also be left out completely if CoexEditEditor=FCKEditor or CoexEditEditor=TinyMCE or CoexEditEditor=Xinha was added to the Notes.INI file for the server. There are additional properties, but they should only be set under consultation with Genii Software support.

The **$CoexEditFlds** field is a multi-value field set to the names of the rich text fields which should be editable with the Notes client but which have matching web rich text fields in the **$CoexEditFldsH** field. If this field does not exist, it will default to a value of "Body".

The **$CoexEditFldsH** field is a multi-value field set to the names of the rich text fields which should be editable with the FCKEditor and which have matching Notes rich text fields in the **$CoexEditFlds** field. If this field does not exist, it will default to a value of "BodyWeb".

## Additional Fields used in samples

The **HTMLBodyContent** field is computed to include the FCKEditor references to style sheets and other resources used by the editor. It is not used directly by CoexEdit, but is simply a convenient way of storing the content used by the HTML Body Content formula on the main form.

# 5

# Monitoring and Debugging

*This chapter will discuss a variety of utilities that we have included for you. They will provide an easy way to get started, as well as an easy way to test your own code. Mainly, these utilities will provide easy ways to send or receive messages.*

## When should you use monitoring and debugging?

Only when a specific problem arises. Most of the time, CoexEdit runs quietly in the background on your server, and there is no need to turn on any monitoring, tracing or debugging. On occasion, usually in consultation with the support people at Genii Software, you may turn on debugging to track a specific problem. Since messages from debugging accumulate quickly, only turn on debugging when necessary, and start with a CoexEditDebug level of 1 first. If you need additional information, turn on CoexEditDebug to a level of 2, but monitor the growth of the Notes Log when in this mode. When you are finished diagnosing a particular problem, remember to either remove the CoexEditDebug option or set it to 0. Besides the log file size, debugging can inhibit performance.

## CoexEditDebug

Determines the debugging level. Normally, this option should be left out or set to 0, which implies no debugging. Debugging information is to the log.nsf. As debugging can inhibit performance, it should only be used when specific information is sought.

| | |
|---|---|
| CoexEditDebug=0 | (all debugging disabled – same as leaving option out) |
| CoexEditDebug=1 | (minimal debugging enabled) |
| CoexEditDebug=2 | (full debugging enabled) |
| CoexEditDebug=3 | (invasive debugging enabled) |

# 6

# Troubleshooting

*This chapter will discuss different potential problems and the possible causes and cures.*

## Editor toolbar does not appear, but textarea does

The exact cause of this may depend on the editor and implementation, but one common cause is an OnLoad JavaScript event that interferes with the window.onload code which implements the editor. If you have code in your OnLoad event, it should be moved down into the editor's subform, usually into the window.onload code in the subform's JavaScript header.

## Modifications made from the web are not saved

There are a few different causes for this, but a couple of things to watch for are the **Generate HTML for all fields** property and WebQuerySave agents. The **Generate HTML for all fields** setting will cause problems with implementations that use innerHTML to get the content of the rich text field since the rich text field will show up twice on the document as a named element. Switching to an editor implementation which uses textarea replacement is recommended if these setting is necessary. If you do switch from an innerHTML to textarea implementation, do not forget to switch the rich text field on the web subform to an editable field from a computed field. You may also need to set the id of the rich text field.

A WebQuerySave agent which modifies the rich text field

# Appendix A - FCKEditor

*Integration details for the "FCKEditor" web editor.*

## What's in a name?

FCKEditor is named for **Frederico Caldeira Knabben,** the original creator and the leader of the team that supports and enhances the product (see team page).

## General Information

FCKEditor is an HTML text editor with many of the functionalities provided in desktop editors, including spell checking, source formatting, table manipulation, anchor insertion, link creation and image insertion. FCKEditor is an open source effort, and is available for free under the GNU Lesser General Public License. Both source code and distribution packages are available at the FCKEditor website. CoexEdit has been integrated and tested using FCKEditor 2.0 RC2, which we make available from our website, but it is likely that subsequent

## Genii Warranty

Genii Software does not guarantee the availability, viability or stability of any third party product. FCKEditor is used as a convenient, free HTML web editor, but our use in samples and documentation does not constitute an endorsement or warranty about the performance of the FCKEditor software. Use at your own risk.

## Implementation

The implementation we have provided in the **FCKEditor with CoexEdit** sample database (FCKEditorCXE.nsf) assumes that the FCKEditor source code has been installed on your Domino server in the data/domino/html/FCKEditor directory, starting with the Domino data directory.

This implementation uses the subform approach (See Section 4 – Implementation), with an **HTMLBodyContent** field computed to include the FCKEditor references to style sheets and other resources used by the editor.

FCKEditor has three additional features which will be incorporated in other editors soon. These are the ability to import local images, the ability to create file attachments (but not two of the same name in this version), and the ability to create doclinks and linkhotspots if an appropriate form and view are defined in your database.

## Limitations

There are very few limitations with FCKEditor, which is currently our preferred open source web editor. One is that file attachments made from FCKEditor must not have the same name, which will be addressed in a subsequent version. A second is that doclinks and link hotspots must be made from the current database. That will also be addressed in a seubsequent version of CoexEdit.

B

# Appendix B - TinyMCE

*Integration details for the "TinyMCE" web editor.*

## What's in a name?

TinyMCE is named for MoxieCode (Tiny MoxieCode Editor)**,** the creators and main developers of the product (see MoxieCode page)..

## General Information

TinyMCE is a cross platform JavaScript based HTML text editor with a very low footprint that works well converting any HTML TEXTAREA into a mini-WYSIWYG editor. TinyMCE is an open source effort, and is available for free under the GNU Lesser General Public License. Both source code and distribution packages are available at the MoxieCode website. CoexEdit has been integrated and tested using TinyMCE 1.44, which we make available from our website, but it is likely that subsequent versions will also be compatible.

## Genii Warranty

Genii Software does not guarantee the availability, viability or stability of any third party product. TinyMCE is used as a convenient, free HTML web editor, but our use in samples and documentation does not constitute an endorsement or warranty about the performance of the TinyMCE software. Use at your own risk.

## Implementation

The implementation we have provided in the **TinyMCE with CoexEdit** sample database (TinyMCECXE.nsf) assumes that the TinyMCE source code has been installed on your Domino server in the data/domino/html/TinyMCE directory, starting with the Domino data directory. We have a second **Integrated TinyMCE with CoexEdit** sample database (IntegratedTinyMCECXE.nsf) in development which does not require any installation on your server, but instead uses design elements inside the database itself.

This implementation uses the subform approach (See Section 4 – Implementation), with an **HTMLBodyContent** field computed to include the TinyMCE references to style sheets and other resources used by the editor. The JavaScript Header is very important and is set directly in the subform.

## Limitations

TinyMCE is very lightweight and easy to use, but it has fewer editor functions than FCKEditor. The refresh of the screen when the HTML switches to WYSIWYG is somewhat annoying as well. We have not encountered many problems when using Firefox as a browser, but occasionally have JavaScript errors when running with Internet Explorer.

The implementation of TinyMCE works with HTML TEXTAREA's, so it may be necessary to restrict use of multi-value text fields that also show as TEXTAREA's.

C

# Appendix C - Xinha

*Integration details for the "Xinha" web editor.*

## What's in a name?

Xinha (pronounced like Xena, Warrior Princess) stands for "Xinha is not htmlArea", which refers to the fact that it is a branch off from the popular, but not being continued, htmlArea 3 web editor.

## General Information

Xinha is a cross platform JavaScript based HTML text editor with a robust combination of features and ease of use. It can convert any named HTML TEXTAREA into a mini-WYSIWYG editor. Xinha is an open source effort, and is available for free under a specialized htmlArea license based on the BSD license. Both source code and distribution packages are available at the Xinha website. CoexEdit has been integrated and tested using a specific build of Xinha that we make available from our website, but it is likely that subsequent versions will also be compatible.

## Genii Warranty

Genii Software does not guarantee the availability, viability or stability of any third party product. Xinha is used as a convenient, free HTML web editor, but our use in samples and documentation does not constitute an endorsement or warranty about the performance of the Xinha software. Use at your own risk.

## Implementation

The implementation we have provided in the **Xinha with CoexEdit** sample database (XinhaCXE.nsf) assumes that the Xinha source code has been installed on your Domino server in the data/domino/html/Xinha directory, starting with the Domino data directory.

This implementation uses the subform approach (See Section 4 – Implementation), with an **HTMLBodyContent** field computed to include the Xinha references to JavaScript and other resources used by the editor. The JavaScript Header is very important and is set directly in the subform.

## Limitations

Xinha is a wonderful editor in many ways, but it seems to suffer from not being at a specific stable build yet. Like TinyMCE, we have not encountered many problems when using Firefox as a browser, but have had issues with JavaScript errors when running with Internet Explorer.

The implementation of Xinha works with named HTML TEXTAREA's, so it is important to set the names of the fields correctly. It does not share TinyMCE's issue with multi-value text fields that also show up as TEXTAREA's for this reason.

D

# Appendix D – EditLive! for Java

*Integration details for the "EditLive for Java" web editor.*

## What's in a name?

EditLive for Java is a commercial product created by Ephox, and it is a successor to other EditLive products created for HTML/XML. This is simply a continuation of the brand name.

## General Information

EditLive for Java is a cross platform Java applet based HTML text editor with a very powerful set of features and high ease of use. It can convert any named HTML TEXTAREA into a mini-WYSIWYG applet editor. EditLive for Java is a commercial product, and not open source, but it has a highly extensible API to allow modifications and enhancements and customizations. EditLive for Java is available from Ephox and all distributions should come from them.

## Genii Warranty

Genii Software does not guarantee the availability, viability or stability of any third party product. EditLive for Java is used as a powerful, commercial grade product that may be well suited to Enterprise use, but our use in samples and documentation does not constitute an endorsement or warranty about the performance of the EditLive for Java software. Use at your own risk.

## Implementation

An implementation database is not currently available for EditLive for Java, as we are working with Ephox to provide a state of the art implementation. Please contact Genii Software if you would like to work with CoexEdit and EditLIve for Java and would like to participate in a beta of this implementation.

## Limitations

EditLive for Java is implemented as a Java applet, which requires allowing use of Java and which makes sizing of the editor area a little trickier than with some of the JavaScript implementations. This is more than made up for by the cross platform consistency allowed by the Java applet approach.