LINKBACK PROJECT
Developers Guide

March 29, 2005

ABOUT LINKBACK

LinkBack is an open-source technology that helps applications on Mac OS X cooperate so users can use them together to do their work. You can add LinkBack to your application so your users can:

- Paste content from other applications in your documents and then later edit or update that content with just a double-click or picking an item on the menu.
- Edit content they created in your application and pasted into another application document. Your application will automatically open and display the content for editing and then send any changes back to the cooperating application document.
- Automatically update content created in your application and pasted into another application document. Your application will automatically launch, update the content and send the edits back to the cooperating application document.

This developer's guide takes you step by step through deciding how you can use LinkBack in your own application and then actually integrating LinkBack into your own software. Read the remainder of this chapter to learn about the general architecture of LinkBack, how it can be used, and guidelines on how you can use LinkBack in your own application. The remaining chapters dive into the specifics of how you can incorporate LinkBack in your application based on how you decide to use it.

The LinkBack Architecture

The purpose of LinkBack is to provide a standard way for applications to coordinate sending content created in one application but pasted into another back to the original application so it can be edited or updated. It's API is designed to make it as easy as possible to integrate the software into your own application. If your application is a standard Cocoa application using the NSDocument architecture, this should be especially easy but you can easily integrate LinkBack no matter how your application is structured.

LinkBack works between two applications at one time. The application that holds the data and requests that it be edited or updated is called the **client application**. The application that creates the data in the first place and later is responsible for editing or updating the data is called the **server application**. Both the client and server applications include LinkBack as a private framework in their own application bundles. Both applications must support LinkBack though they may use different versions of the framework.

Usually, a user will create some content in the server application, copy it onto the pasteboard and paste it into a client application document. Later the the client application that may contact the server application to request the content be modified. The server application either displays the content for the user to edit or it updates the content automatically. Either way, changes to the content are sent from the server application to the client, where it will replace the original content with the changes it received.

They way LinkBack facilitates these actions in three ways:

1. Whenever a user copies content in the server application, the server uses LinkBack functions to create a special LinkBackData object that will be included on the pasteboard along with all the other standard pasteboard types the server application would provide. Applications that are not LinkBack enabled will simply use the standard pasteboard types when the user pastes. Client applications, however, will notice the LinkBack data and will save that data along with the standard pasteboard types they understand.
2. When a client wants to modify some LinkBack content, it can pass the LinkBack data it saved to the LinkBack framework which will locate the server application, launch it if necessary, then send the server application the LinkBack data so it can edit or update it.
3. When a server application launches, it uses the LinkBack framework to listen for edit/update requests. When it has changes to send back to the client application, it uses the LinkBack framework to create an updated LinkBack data object and then the framework will send this updated content back to the client application.

The diagram below depicts a client and server application, their LinkBack frameworks and the actions performed by each component in this system.

Choosing How to Use LinkBack in Your Application

The first step in deciding how to use LinkBack in your own application is to decide if you want your application to act as a client, a server, or both. Use the guidelines below to decide what LinkBack features you should support in your application:

- Your application should be a **client** if your users will want to paste content from other applications and then edit or update that content later.
- Your application should be a **server** if your users will want to create content in your application, paste it into another application document and then edit or update that content later.
- Your application should be both a **client** and a **server** if your users may want to include content from other applications in your application and include content from your application in other applications.

When you make your application both a client and a server, you enabled LinkBack **chaining**. This feature means that a user can progressively build complex content by creating content in one application, pasting it into another and adding new content then pasting all that content into yet another application, and so on. If you properly design your application, users can build complex content in this manor and then edit all it all the way back to the original content they began with.

Once you have decided how you want to use LinkBack in your application, read the appropriate chapters in this book to learn what to do with the LinkBack API in your application:

- Everyone should read the next chapter, "*Adding LinkBack To Your Application*".
- If you want to support LinkBack as a server application, read the chapter "*Using LinkBack*

for Server Applications."

- If you want to support LinkBack as a client application, read the chapter "*Using LinkBack for Client Applications*."
- If you want to support LinkBack as a client and a server application, read both chapters for client and server applications and also read the chapter "*Special Considerations for Chaining Applications*."

A primary goal of LinkBack is to make it easy for applications to coordinate while keeping the time and effort required for you the developer to incorporate LinkBack into your application to a minimum. If you have any suggestions on ways we can improve the LinkBack framework, please let us know by e-mailing your suggestions to linkback@nisus.com.

ADDING LINKBACK TO YOUR APPLICATION

Whether you are adding LinkBack as a client or a server, you will need to incorporate the LinkBack framework into your application. This chapter describes how to do just that. It assumes that you are using the most recent version of Xcode for development with the new Xcode-style targets (not those upgraded from ProjectBuilder.) If your setup is different, your specific instructions may be a little different but the basic steps are the same.

To add LinkBack to your application, you will need to complete the following actions:

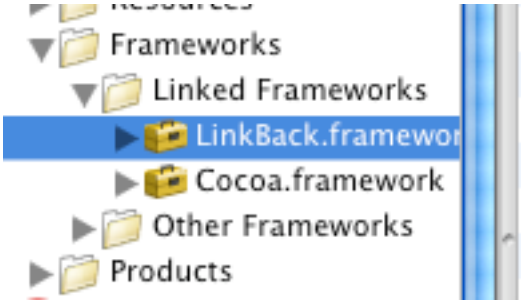
1. Add the LinkBack framework to your project.
2. Add a Copy Phase to include LinkBack in your built application.

Adding the LinkBack Framework to Your Project

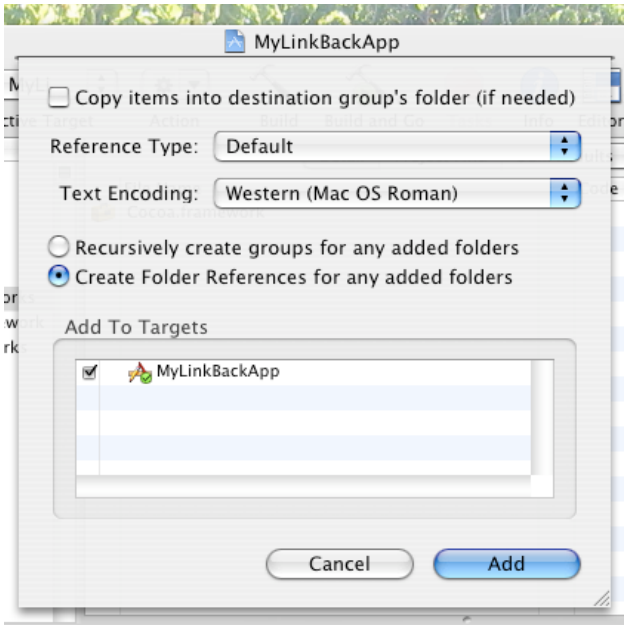
To add the LinkBack framework to your project, you will first need a binary copy of the LinkBack framework. You can either use the built version that is included in the LinkBack distribution or you can build a copy yourself using the LinkBack project file. Either way, you should store this binary copy of the LinkBack framework in some standard location so your Xcode will always be able to find it.

To update the LinkBack framework in your application in the future, you will simply replace the framework at this standard location with an updated copy and rebuild your project. Xcode will do the rest for you.

Once you have a binary copy of the LinkBack framework, open your Xcode project and drag the framework into “Linked Frameworks” group in your project.



When you do this, a sheet will appear with several options about including the framework. You can leave everything as its default, however you must make sure that your application target is checked in the list of targets shown at the bottom.

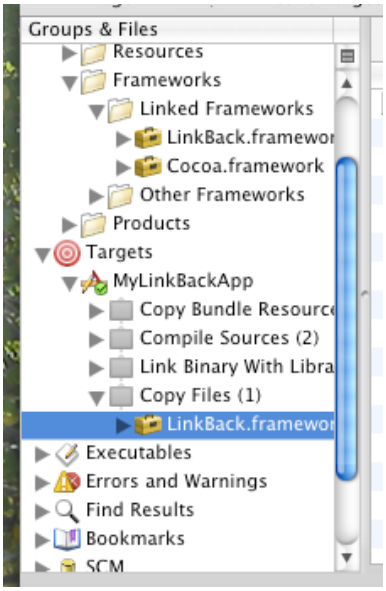


Now the framework is added to your project and you are ready to add a copy phase.

Adding a Copy Phase to Your Application Target

You need your LinkBack framework to be copied into your built application or else it will not run when you launch it. To do this, you will add a copy phase to your application target. Do the following steps:

1. Select your application target in the "Targets" section of Xcode.
2. Choose Project->New Build Phase->New Copy Files Build Phase
3. A "Copy Phase" info panel will display. Under the "General" tab choose the pop-up menu labeled "Destination" and pick the item "Frameworks".
4. Close the Copy Phase info panel.
5. If you toggle the disclosure triangle next to your target in Xcode, you will now find a new phase labeled "Copy Files". Find the icon for the LinkBack framework where you added it to the project under "Linked Frameworks" and drag it to your Copy Files phase. This should add the LinkBack framework underneath the copy files phase as shown below.



Moving On

That's it! Try to build and run your application just to make sure you added everything properly. You should check your application bundle to make sure a frameworks directory is created with the LinkBack.framework bundle inside of it. Once you have finished this, you can move on to actually integrating LinkBack into your application.

A Word on Testing

The LinkBack distribution includes two samples applications: a server application called LiveSketch and a client application called TextEdit+LinkBack. You can use these two applications to test your application as you add functionality to it. Note that neither application currently saves LinkBack content.

USING LINKBACK FOR SERVER APPLICATIONS

Adding support for LinkBack to your application as a server is fairly straight forward. You will need to plan a few things first and then make some changes to your code.

Planning

Before you being making any changes, you will need to plan two things about your design:

Pick Your Server Name

Your server name will be used to identify your application when it is running so client applications can contact it requesting an edit. You can choose any name you like; it only needs to be unique to your own application.

> **NOTE:** It is possible for your application to respond to several different server names. Though you will probably never need this, you can use this to support different types of LinkBack data if needed.

Choose Your Application Data

Whenever a user copies some of your application's content to the pasteboard, the application will include LinkBack Data along with its other standard pasteboard types. This data contains several pieces of information, but primarily it includes application data that you provide. This data will be sent back to you when clients request an edit. This application data should contain whatever you need to edit or update the data when clients send it back to you.

Your application data can contain anything but it should meet the following guidelines.

1. The data must be supported on the pasteboard. This means it can contain only array, dictionary, string, number, or data objects.
2. The data may be included in documents saved for years to come. Be sure to include any necessary versioning information so future versions of your application will be able to interpret the data.
3. You must decide if your LinkBack data will just be a reference to the original data or a completely separate copy of the data. If it is a reference, your application data only needs to contain a link to the original data. If the data is a copy, it you can create an archive of your application data to restore later.

The differences between including a reference or a copy of your application data can be significant for the user experience. For example, if you have a graphics application, you should probably include an editable copy of your selected content. This would allow a user to receive a document containing your graphic and then they could edit that graphic without needing the original file. On the other hand, a database or mapping application might include only a reference to the content so that when the user opens a document containing your data, you can update the content with the latest information from the database or mapping information.

Modifying Your Code

With your design decisions made, you will need to make the following changes to your application:

1. Add LinkBack items to your Info.plist file.
2. Register to listen for LinkBack edit requests on application launch.
3. Modify your copy and cut code to include LinkBack data with other pasteboard types.
4. Handle LinkBack edit requests when you receive them.
5. Send edited or updated content back to the client application.
6. Handle the link closure and abnormal termination conditions that may occur with the client.

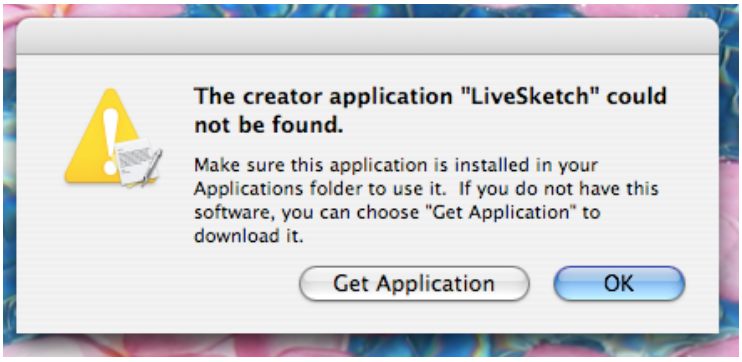
Adding LinkBack Items to Your Info.plist File

Each application target has an Info.plist file. You can find this file by choosing your target in Xcode, then choose "Get Info" and click on the "Properties" pane. Click on the button at the bottom of this pane "Open Info.plist as File." If you know where this file is in your project, you can also open it in the Property List Editor for a little nicer interface.

If your Xcode files uses an "old style" application target (the icon for your application target looks like a target rather than an application) you can access these settings by double-clicking on the target icon. Choose "Expert View" under the "Info.plist Entries" group in the window that appears.

No matter how to access this Info.plist, you will need to add two keys to the Info.plist:

1. **LinkBackServer** -- The value of this key contains a string that identifies the server names you will use to listen for edit requests from clients. If LinkBack cannot find the application that originally created some LinkBack data, it will use these values to find an application on disk that will respond to the data. (If your application supports multiple server names, this can also be an array of strings.)
2. **LinkBackApplicationURL** -- this is actually only needed if your application is going to be a server. The value of this key should be a string with the URL where the user can find a copy of your application. If the user tries to edit some content that belongs to your application but they do not have your application, LinkBack will display a dialog offering to send them to the URL you provide so they can get your application.



Listening for LinkBack Edit Requests

The first code you will need to add tells LinkBack that you want to respond to edit requests coming from client applications.

To do this you need to choose an object that will respond to LinkBack edit requests in your application. This should be an object, such as your `NSApplication` delegate object, that will exist throughout the entire life of the application. You will need to make it conform to the `LinkBackServerDelegate` protocol declared in LinkBack.h. You may want to stub this code in for now; it will be filled in later on in this chapter.

Next, publish your interest in receiving edit notifications when your application launches. You do this by adding the line:

```
[LinkBack publishServerWithName: YOUR_SERVER_NAME
delegate: YOUR_DELEGATE_OBJECT]
```

You need to send this message somewhere right after your application launches and before it starts receiving events. A good place to add this code is in the `-applicationWillFinishLaunching:` method of your application delegate if you have one.

> **NOTE:** You can also stop listening for LinkBack edit requests at anytime by using `LinkBack`'s `+retractServerWithName:` method. This is not necessary if you plan on listening for edit requests until your application terminates.

Including LinkBack Data For Pasteboard Operations

Whenever the user places any of your content on the pasteboard, either via Cut or Copy operations or even using drag and drop, you need to include a special LinkBack data type along with your other standard data types on the pasteboard. LinkBack-enabled applications will notice this data and save it along with your norma pasteboard data. Other applications will use the normal data and ignore the LinkBack data.

> **NOTE:** Be sure you include these modifications anywhere you write content you would want the user to edit or update later to the pasteboard. This can include places you respond to the Cut or Copy commands, or where the user starts a drag that may go to

another application.

This phase requires two modifications to your code:

1. Include the LinkBack data type when you declare the data types you are putting on the pasteboard. Wherever you call NSPasteboards `-declareTypes:owner:` add `LinkBackPboardType` to the list of types you support.
2. Whenever you actually write data to the pasteboard, create the LinkBack data and provide it for the `LinkBackPboardType`. Cocoa allows you to provide pasteboard data immediately or lazily. You can use either method to provide the LinkBack data as well.

You create the LinkBack data to place on the pasteboard using one of several methods defined in the `LinkBackData` category on `NSDictionary` defined in LinkBack.h. The simplest method you can use is:

```
[NSDictionary linkBackDataWithServerName:
YOUR_SERVER_NAME appData: YOUR_APP_DATA] ;
```

The server name is the one you have chosen for your application and the same name you used to publish your interest easier. The application data is the data you create from your current selection, as described in the planning section of this chapter.

Other methods allow you to provide several additional items:

- **actionName:** This is a string that describes the action you will perform when the client requests an edit. The action names are used by clients to build a menu item for the user to invoke on your LinkBack content. Two standard action names are defined:
    - `LinkBackEditActionName` is used when your application will display the content for the user to edit. If you do not specify an action name, this is the one that will be used.
    - `LinkBackRefreshActionName` is used when your application will update the content and return it automatically to the client.
    - You can include your own action name if you want, however the above two standard names will be localized while your custom name will not.
- **suggestedRefreshRate:** This is a time interval that suggests how often the client should automatically request an edit for your LinkBack data. This is appropriate only if you support a refresh action rather than an edit. Client applications may not respect this refresh rate (see the chapter on client applications for more on how this is handled) but if the application does support it, this rate will trigger automatic updates. If you do not require or support automatic refreshes, set the refresh rate to 0.

Handling LinkBack Edit Requests

When a client requests an edit of your application data, your server delegate will receive the `-linkBackClientDidRequestEdit:` message. The object passed in is an instance of the LinkBack class. This represents a single connection between the client and the server application. A LinkBack instance is created in both the client and the server each time the client

requests and edit from the server. The instances are connected to each other and relate information back and forth. At any time, the link between these two objects may be closed. Once a link is closed it cannot be reopened; the client will need to request a new edit instead.

You should respond to this message by either opening the data for editing or by automatically refreshing the data.

Opening Data for Editing

For NSDocument-based applications, the most straight forward way to handle this is to modify your NSDocument subclass so that it can be created from from a LinkBack in addition to its current methods that can create a new empty document or load from a file on disk. Your document object should do the following with the LinkBack object:

1. Retain the object for future use. You will use the LinkBack object to send edits back to the client application.
2. Obtain your application data from the LinkBack object and use that to populate your NSDocument object. You can obtain your application data with the following message:

```
id myAppData = [[[aLinkBack pasteboard]
propertyListForType: LinkBackPboardType]
linkBackAppData] ;
```

The window you display is showing content from another application, not from a file. It should not show any icon in the title bar and the window name should be something like:

Graphics from My File (Nisus Writer Express)

The type, like "Graphics", is something you should choose based on the type of content you are displaying. The name "My File" can be obtained using the `-sourceName` method on the LinkBack object. The application name can be obtained using `LinkBack`'s `-sourceApplicationName`.

Refreshing Data Only

When you refresh data only, you may not need to show any user interface at all. You could just retrieve the data from the pasteboard using the same method described in the previous section, update it, and send it back.

If refreshing the data will take more than a few seconds, you should consider showing a progress panel that will tell the user what you are doing. The panel should include a Cancel button so the user can stop your refresh operation if it is long.

If the user cancel's an update, you simply need to close the link to the client application using the LinkBack `-closeLink` method.

Sending Edited Data Back to the Client Application

You should send your edited data back to the client application as soon as it is available during a refresh. If you show the content so the user can edit it, you should send data back to the client

application whenever the user chooses the “Save” menu item.

To send edited data back to the client application:

1. Generate new LinkBack data for all your edited content.
2. Place the LinkBack data along with any other standard pasteboard formats you would normally place on the pasteboard during a Copy operation onto the special pasteboard provided by the LinkBack object. You can obtain this pasteboard object using the `-pasteboard` method.

   > **NOTE:** The pasteboard provided by each LinkBack object has been created just for that one link between the client application and your server application. Changing the contents of this pasteboard will not interfere with copy/paste or drag and drop operations.

1. Notify the client that new data is available by invoking the `-sendEdit` method on the LinkBack object.

If you are refreshing the data only, you may now be finished with the edit, so you can also close the link by sending the LinkBack object the `-closeLink` message.

Handling Link Closure and Application Termination

For various reasons, the client may choose to close its link with your server application before the application sends edits back to it. This may happen if the user closes the client document or if the client application terminates.

Your application server delegate will receive the message -linkBackDidClose: whenever the client application closes the link. You need to add code to this method to close any edits or updates you might have in process for that link. If you have a window open for the user to edit content, you should close that window.

USING LINKBACK FOR CLIENT APPLICATIONS

Adding LinkBack support to your application as a client is slightly more complicated than server support, but it is still fairly straightforward.

Planning

Before you start changing code, you will need to plan two things about your design:

Choose Your Saving Format

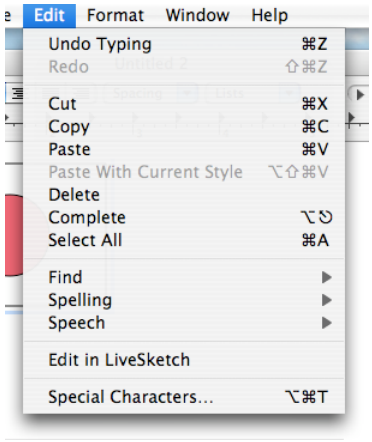
To support LinkBack, you will store the LinkBack data you find on the pasteboard along with any other normal content when a user drops or pastes content into your application. You will also need to save the entire LinkBack data object in your data files. When deciding how to save this data object, keep the following guidelines in mind.

1. The LinkBack data object is an NSDictionary object that can be archived using standard Cocoa methods. All objects will be dictionaries, arrays, strings, numbers, and NSData instances.
2. You should not depend on the LinkBack data object containing any specific key/value pairs. Depending on the version of LinkBack used in the server application, the contents of the dictionary may differ.

Design Your User Interface

When you have a document with LinkBack content, you need to provide some feedback that the user can edit or refresh the content. You also need to provide a way for the user to initiate an edit or refresh. As you design this user interface, keep in mind the following guidelines.

1. A standard LinkBack badge is provided with the LinkBack distribution. You can show this badge in the lower right hand corner of graphics or drawing to indicate LinkBack is available. The badge should only display on screen and when the document is editable.
2. All LinkBack clients should have a standard menu to initiate LinkBack edits. This menu appears below the “Find”, “Spelling” and “Speech” menu items in the “Edit” menu with a separator above it. LinkBack provides two helper functions and a method to automatically generate the menu name. You should update the name each time your menu is validated. See the section on modifying your user interface below for more instructions.



- Users typically can also initiate a LinkBack edit by double-clicking on the item in your document. While this is not required, especially if you already using double-clicking for other things, many users will do this instinctively from years of training in other applications so you should support it if it makes sense for your user interface.

Modifying Your Code

To add LinkBack support for your document you will need to modify your application in the following ways:

1. Add support for storing LinkBack data in your documents.
2. Modify your pasteboard reading code to extra LinkBack data.
3. Modify your user interface to initiate edit requests.
4. Respond to updates sent by server applications.
5. Add support for link closing and terminating conditions
6. Add support for saving and and reloading LinkBack data.

Storing LinkBack Data In Your Documents

You will need to store any LinkBack data you encounter in your application document. You should store the entire LinkBack data object you will get off the pasteboard. It should be attached to the regular content you get from the pasteboard at the same time.

Modifying Your Pasteboard Reading Code For LinkBack

You will need to modify any place in your code where you read content off the pasteboard to look for LinkBack data. Your modifications should:

- Perform its pasteboard reading as normal, using any standard formats it understands.
- Look for the presence of the `LinkBackPboardType`.
- If it is exists, get the link back data using something like:

```
id linkBackData = [pboard propertyListForType:
```

```
LinkBackPboardType];
```

- Store the link back data with the normal data you also retrieved from the pasteboard.

Modifying Your User Interface To Initiate Edit Requests

You should add support in your user interface for initiating edit requests according to the user interface changes you planned at the beginning of this chapter.

When adding support for your menu item, you should have the menu item trigger your standard edit action and add menu validator code to update its title and enabled state based on the current selected content. LinkBack provides several helper methods and functions to generate the menu title for you. Use the table below to give your validator code the proper behavior.

LinkBack items in selection	Enabled?	Use for menu title
0	NO	`LinkBackEditNoneMenuTitle()`
1	YES	`[linkBackData linkBackEditMenuTitle]`
> 1	YES	`LinkBackEditMultipleMenuTitle()`

.

When the user chooses to initiate an edit request, your application should initiate an edit for each selected LinkBack item using the following method:

```
LinkBack* link = [LinkBack editLinkBackData: YOUR_DATA
sourceName: SOURCE_NAME delegate: DELEGATE itemKey: KEY] ;
```

- YOUR_DATA is the link back data you want to update.
- SOURCE_NAME is the name of the document the data comes from. This should not be your application name since that is provided separately. This will be used to display a title in the server application.
- DELEGATE is the object you want to be notified when updates or sent from the server or when the link closes. Usually this will be the object responsible for keeping track of the LinkBack object as well, such as your document object. This object must conform to the `LinkBackClientDelegate` protocol.
- KEY is a key you can pass to uniquely identify the item in your document this LinkBack is updating. Since you will be replacing the LinkBack data object with a new one sent by the server. This key can be anything you want, but if you just need to make sure the key is unique, you can use the function `LinkBackUniqueItemKey()` to generate one.

The returned value of this method is a LinkBack object. This object represents a single connection between your client application and the server. You will need to store this LinkBack object somewhere you can find it later if you need to close the link. (For example, when the user

closes the document containing the LinkBack data being edited, you will use the LinkBack objects to close any open links.)

If your application is NSDocument based, a good place to store the set of open LinkBack objects is in your document object. You do not need to save these objects.

Responding To Updates Sent by Servers

When a server edits or refreshes some data, it will send an edit back to the client application. When this happens, your application will be notified with a `-linkBackServerDidSendEdit:` message to the LinkBack's delegate object.

Your implementation of this method should basically do two things:

1. Read the standard and LinkBack content off of the LinkBack pasteboard (use LinkBack's `-pasteboard` method to retrieve this pasteboard.)
2. Replace the LinkBack data and the content it is attached to in your document with the new content.

This process is very similar to the user selecting the content in your document and then pasting the contents of this special pasteboard. You should be able to reuse your existing pasteboard code accordingly.

Be sure you update your display after you change this content. The user will be looking for the appearance of their document to reflect the changes as feedback that their edit or refresh succeeded.

You should also mark your document as being dirty at this point since its contents have changed.

Supporting Link Closure and Termination Conditions

At various times the server application may choose to close the link with your application. This may happen because the user has closed the edit window in the server application or because it has completed its refresh, or maybe because the application quit or crashed. When this happens, the delegate object for the LinkBack object will receive the `-linkBackDidClose:` message.

When you receive this message, you will need to remove the LinkBack object from your list of LinkBacks that you are maintaining and release any other resources you were holding on to for the LinkBack session. The LinkBack object will no longer function when you receive this message, so you can just release it.

If the users quits your client application or closes a document with open LinkBacks, you should also close any open links. Do this by sending the `-closeLink` message to every open LinkBack object for the documents you are closing. This will clean up everything on the server side as well and close all windows.

Saving and Reloading LinkBack Data

Finally, you will need to save and reload LinkBack data in your document. Implement these changes according to the design developed earlier in the chapter.

Your done!

USING LINKBACK IN BOTH CHAINING APPLICATIONS

If your application is both a client and a server, congratulations! You will help your users reach the highest level of LinkBack nirvana.

You need to implement both the client and server changes described above. You should also take into account a few other items:

1. When pasting content that was copied from your own application, you probably do not want to embed your own LinkBack data. You can find out if the data was created by your own application by calling the method `-linkBackDataBelongsToActiveApplication` on the LinkBack data object.
2. Be sure to test for the case where a users inserts some LinkBack-enabled content into your application, adds a few things, then copies all of that from your application and pastes it into yet another LinkBack enabled application. The user should be able to reopen the data for editing in your application and then in turn reopen data in your document in that original application for editing also.