Technical Notes on FieldWorks Send/Receive

September 8, 2015

Contents

1 Send/Receive Introduction

FieldWorks Language Explorer (FLEx) provides two ways to collaborate with colleagues working on the same project. The first approach (starting in FieldWorks 7.3) includes the entire FieldWorks project: the full lexicon, grammar, interlinear texts, data notebook, scripture, lists, pictures, sound files, writing systems, and some configuration information. This is the approach that should be used whenever you are collaborating with other FLEx users. The second approach is via the Lexicon Interchange Format (LIFT). This approach should only be used if you are collaborating with WeSay users or some future program that can only use LIFT. LIFT only includes the lexicon, parts of grammar, writing systems, pictures, and sound files. The LIFT format does not cover as much detail in the lexicon as the FLEx project format, so the results are less satisfactory when using features of FLEx that WeSay does not use. There are also some extra issues with custom fields and lists.

With either approach, collaboration can be through any combination of the following

- Internet server (e.g., http://languagedepot.org) which requires some initial setup on the server that needs to be requested ahead of time.

- USB drive (no setup needed) that is passed around to users.
- Local Network using ChorusHub (requires setting up an instance of ChorusHub on one machine on a network).

If you are just collaborating between FLEx users, you would only use a FLEx repository (repo).

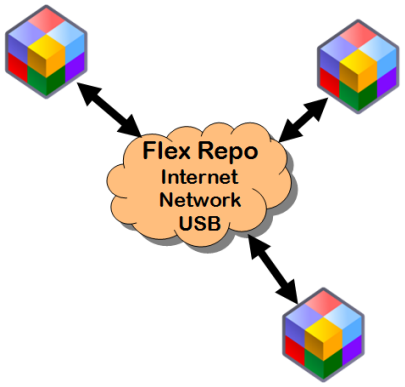


Fig. 1 Flex Collaboration

If you are just collaborating between WeSay users and one optional FLEx user, you would only use a LIFT repo.

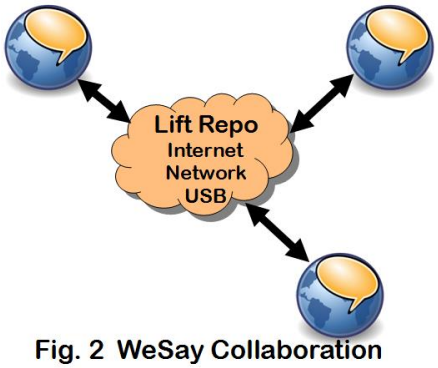


Fig. 2 WeSay Collaboration

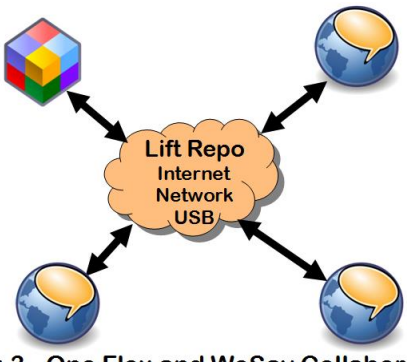
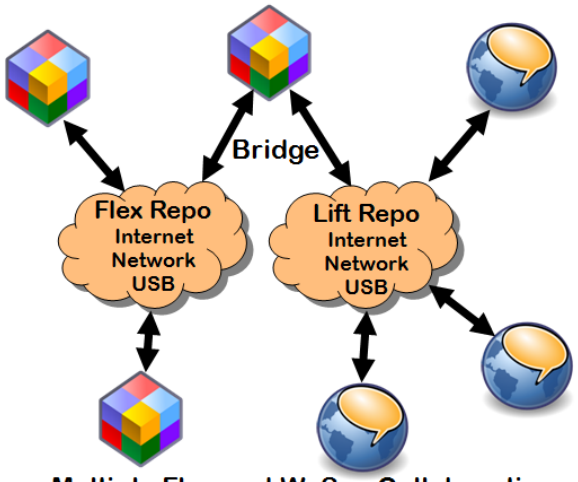


Fig 3. One Flex and WeSay Collaboration

If you are collaborating with multiple FLEx users and one or more WeSay users, then you will need a separate FLEx repo for collaboration between FLEx users, and a LIFT repo for collaborating between WeSay users and a FLEx user. It's critical (explained below) that only one FLEx user act as a 'bridge' between the two repos. This user would periodically sync with the LIFT repo and the FLEx repo. Other FLEx users would just sync with the FLEx repo, and WeSay users can only sync with the LIFT repo.



Multiple Flex and WeSay Collaboration

The FLEx Send/Receive menu has changed in different versions since FW7.3. This document was written for FLEx 8.0.5. Here is a brief summary of how the Send/Receive process works between two or more FLEx users.

One user starts the collaboration process from a master copy of the project. This user does an initial Send/Receive…Send this Project for the first time to store a copy of their project in a Mercurial repository (repo). Each colleague then gets a copy of the project from the repo using Send/Receive...Get Project from Colleague. After that, all colleagues periodically do Send/Receive…Send/Receive Project (with other FLEx users) (S/R) to keep their projects synchronized with their colleagues.

If colleagues are working in different parts of the project, the S/R will merge their changes without any conflicts or loss of data. If colleagues change the same piece of data (e.g., the definition of the same sense), during the S/R process the program will merge the changes the best it can (e.g., picking one of the two definitions), then add a conflict report warning the users about a change that was made that would be wise to review. If an undesirable change was made in a merge, the user will need to fix the appropriate data manually.

The current S/R process does not support different privileges for different users. Any user doing S/R can modify any part of the data. For an Internet server, a user can be given read-only permission to a repository, which allows them to get a copy of the project, but nothing they do will get back into the repo.

With each S/R operation, changes and new information are appended to the repo with a timestamp. Merges use a 3-way process that can generally tell whether the change was an addition by one user, a deletion by one user, or a modification by both users. Each user has a local (hidden) copy of the repo as it was the last time they did a S/R, so this helps the program to know what changed since their last S/R. FLEx does not provide a way to see what changed

during a S/R—it only shows merge conflicts that occurred. Some special programs outside of FLEx let you see the history of changes in the repo, and it is possible to go back in time to an earlier version of the project, but this takes more advanced skills.

When using S/R with a project, you can still back up your project using FieldWorks backup, but you need to be very careful about using a FieldWorks restore, as this will likely cause all colleagues to lose their work since the backup was made.

2 Getting started

2.1 Starting up Project (FLEx) Send/Receive

Before your first S/R you should make sure that you have a single project that can be considered your master project. If two users created their own FieldWorks project and added entries independently, the merge collaboration will not work. Also, if two users started with the same FieldWorks project, but have made independent changes, the merge collaboration will not work. This is discussed in more detail later. If you have multiple projects, you should find some way to merge these before doing your first S/R. Once you do the initial S/R of your project, all other collaborators need to delete or move (not just rename) any copies of the old project from their FieldWorks Projects directory and then start over by getting the master project from the new repo.

If you plan to use Chorus Hub or Internet options, you first need to set these up. See the following sections for details.

The first step is for one user to do an initial S/R from their master project. To do this, follow these steps.

1. Go to Send/Receive...Send this Project for the first time.
2. Optionally type a label in the Send/Receive Project dialog
3. a) Click USB Flash Drive or Chorus Hub if you are not using the Internet button. This will start the Send/Receive process to the specified device.
   b) For Internet
   1. Click the Settings box in the lower right to bring up the Send/Receive Settings dialog.
   2. Make sure Server is set to LanguageDepot.org
   3. Fill in your LanguageDepot project id, user login and user password
   4. Click OK to close the Send/Receive Settings dialog.
   5. Now the Internet button should be enabled, so click that to start the Send/Receive process.

Every other user that wants to collaborate with this project then needs to do a one-time setup to get the project on their machine. To do this, follow these steps.

1. Make sure you have the same version of FieldWorks installed on your machine that was used for the master project.
2. Go to Send/Receive…Check for FLEx Bridge Updates to see if there is a later version of FLExBridge available. If there is, Click Install update and follow the default installation.
3. If you already have an older copy of this FLEx project on your machine use File…Project Management…Delete Project if you can do this from another open project. Otherwise, delete the project folder in C:\ProgramData\SIL\FieldWorks\Projects. On XP this is in

c:\Documents and Settings\All Users\Application Data\SIL\FieldWorks\Projects. The default settings for Windows Explorer will not show this directory. You can still get to it by typing at least part of the path into the edit box at the top.

4. Start FLEx. If the startup screen comes up, choose "Get project from a colleague". Otherwise, go to Send/Receive…Get project from colleague. Either option will bring up a Receive project dialog.
5. Choose the location of the repo holding your project
   a) Click USB Flash Drive or Chorus Hub, or
   b) Click Internet. You'll need to specify some additional information in the Get Project From Internet dialog before it will start.
      1. Make sure Server is set to LanguageDepot.org
      2. Fill in your LanguageDepot project id, user login and user password
      3. Type the project folder name you want to use on your computer (not the full path). The project folder will be created in your FieldWorks Projects directory.
      4. Click Download to start the download.

This process will create a new project folder in your FieldWorks Projects directory which will contain a copy of what was in the repo at that point.

Another way users can get started after the first user did the initial S/R is to copy the entire FieldWorks project directory from the first machine to the other machines. If you do this, the first time each user does a S/R, they should click the Settings link in the Send/Receive Project dialog and change the "Name to show in change history:" to the current user.

Once each user has a current copy of the project on their computer, they can periodically do Send/Receive...Project (with other FLEx users) and click the desired location for synchronization. You can do another S/R to a different location if you want to keep several repos in sync.

2.2 Starting up Lexicon (LIFT) Send/Receive in FLEx.

The master project for starting S/R can be either a FLEx or a WeSay project. Note there are a few compatibility issues between FLEx and WeSay. See Section 4.2 for more details.

If a FLEx user is setting up the initial LIFT repo, use Send/Receive…Send this Lexicon for the first time (to WeSay) and then select the desired destination.

If a LIFT repo has already been set up by WeSay, and there is no FLEx project for this language, then the FLEx user needs to connect to the existing repo using Send/Receive…Get Project from Colleague. FLEx will realize that the project is a LIFT repo, so it will create a new FLEx project based on the LIFT repo.

It's possible that you have been collaborating between FLEx and WeSay in the past, but the connection may get broken. For example, if your FLEx project directory gets deleted and you restore from a current FLEx backup. (Note restoring a backup when using Send/Receive is usually dangerous. See Section 4.3 for more details.) To get reconnected to an existing LIFT repo, you need to use Send/Receive...Get Lexicon (WeSay) and Merge with this Project. This will do the initial sync to the LIFT repo without losing other information in your FLEx project, such as interlinear texts. Before doing this command, inside your FLEx language project folder, look for an OtherRepositories folder. If there is anything inside this folder, delete it first.

Once the initial connection has been established, then use Send/Receive…Lexicon (WeSay) whenever you want to sync with the LIFT repo.

2.3 Starting up Send/Receive in WeSay.

The master project for starting S/R can be either a FLEx or a WeSay project. Note there are a few compatibility issues between FLEx and WeSay. See Section 4.2 for more details.

If a WeSay user is setting up the initial LIFT repo, In the WeSay Home tab, click the Send/Receive button, and then choose the destination for the repo.

If a LIFT repo has already been set up by FLEx or another WeSay user, and there is no WeSay project for this language on your machine, then you need to use the WeSay Configuration tool to get connected. In the WeSay Configuration Tool opening dialog, choose "Get from USB drive" or "Get from Internet", choose the desired repo, then click Copy To Computer. This will create a WeSay project on your machine that will be in sync with the repo.

Once the initial connection has been established, then use Send/Receive in the WeSay Home tab whenever you want to sync with the LIFT repo.

2.4 Chorus Hub startup

Older versions of Chorus Hub used a process on the server computer. Recent versions use a service on the server computer. These two versions are described in the next two sections. It should be possible to use a mixture of versions of FLEx Bridge and Chorus Hub without causing problems. But it is recommended that collaborators use current versions of software.

FLEx Bridge 2.2 – Chorus Hub process

In this version of FLEx Bridge, the Chorus Hub 2.4.266 program is installed in the FLEx Bridge directory, and can be started on a single machine on the network. It is not a service, so it is stopped any time the user that started it logs off, or the machine goes into sleep mode. Only one instance of Chorus Hub can be running at one time on a network. If you try to start Chorus Hub when there is already an active Chorus Hub on the network, it will let you know that it can't be started because there is already a Chorus Hub running on the network. It uses a directory (c:\ChorusHub) on the machine that is running Chorus Hub to store repos for all users on the network. Anyone on the network has access to Chorus Hub whether it is running on your local machine or on another computer on the network. Chorus Hub supports repos from several different programs including FieldWorks, WeSay, and Bloom. Each program limits access to repos it can use.

From FW8.0.6 through FW8.1.4, chorushub.exe was installed in the FLEx Bridge directory and could be started from there. C:\Program Files (x86)\SIL\FieldWorks 8\Installers\ChorusHubInstaller.msi was also copied to your machine during installation. This installer can be run on any machine, even without FieldWorks, to run the Chorus Hub 2.4.7 program on that machine. After installing the program, you can launch it from: c:\Program Files (x86)\SIL\Chorus Hub\ChorusHub.exe (omit " (x86)" for 32-bit Windows)

When Chorus Hub is running, a Chorus Hub dialog is present that gives information about the program. Other users that start a Send/Receive operation on the network will have access to the Chorus Hub button in the Send/Receive dialog as long as some machine on the network is

running Chorus Hub. You'll have to wait a second or two for this button to be activated. Before you can close the Chorus Hub dialog, you must click the "Stop Chorus Hub" link at the bottom. The project directories will remain in the ChorusHub directory and can be used again by restarting Chorus Hub.

Initial startup of Chorus Hub may ask to permit hg.exe through the firewall. This should be granted. It may also need ChorusHub.exe clearance through the firewall. This seems to happen automatically with Windows Firewall, but with other firewalls you may need to allow these permissions before it works.

Chorus Hub 2.4 will give continuous warning messages, "this machine has more than one IP address", and may or may not work if it detects more than one IP address on your machine. This will happen if you have more than one network card, if you have wireless as well as wired connections, or if you are running virtual machines. In general, you should pick a machine to run Chorus Hub that has a single IP port. See Section 4.1 for further information on virtual machines.

FLEx Bridge 2.3 – Chorus Hub service

Starting in FW8.2.0, FLEx Bridge 2.3.5 or later is installed as part of the installation. This version uses a newer version of Mercurial (3.3.2) to solve some difficult bugs occasionally encountered with the older version. ChorusHub.exe is no longer installed in the FLEx Bridge folder. A new ChorusHubInstaller.msi file is copied to c:\Program Files (x86)\SIL\FieldWorks 8\Installers. This is Chorus Hub 2.5.20 or later. This version installs a service called Chorus Hub Sharing Service on the computer instead of running a process. As a service, it will be available on the machine whenever it is running and is not dependent on a user being logged in.

Firewall: At least on some machines it is important to add c:\Program Files (x86)\SIL\Chorus Hub\mercurial\hg.exe to the inbound rules in Windows Firewall for Chorus Hub to work. If you get a S/R message, "abort: error: A connection attempt failed because the connected party did not properly respond after a period of time, or established connection failed because connected host has failed to respond" it probably indicates your firewall is blocking hg.exe.

When Chorus Hub is installed as a service, it can't easily warn you about another Chorus Hub already running on the network since services do not use dialogs. If there is more than one machine running Chorus Hub service on a network, as could happen in the past, FLEx will pick one, seemingly the last one that was started. In the S/R dialogs Chorus Hub will identify the machine on which it is running. If for some reason you want to have Chorus Hub server installed on more than one machine, you should stop all services except the one you want to use. During installation, the service will be set to Automatic which means it starts whenever the machine is booted. To start or stop the service, you can type Services in the Start dialog, then click Services at the top of the menu. In this dialog, find Chorus Hub Sharing Service in the right pane. You can right-click this and choose to Start or Stop the service. With more than one installation, you should go to the Properties on this service and set the Startup type to Manual on all but your primary machine which should be set to Automatic so it will start up automatically when you reboot.

When you uninstall Chorus Hub through Programs and Features, it will automatically stop and uninstall the Chorus Hub service.

The Chorus Hub service will send messages to the Event Viewer when there are problems. To see these, type Event Viewer in the Start dialog, then click Event Viewer at the top of the menu

to open the event dialog. In the left of this dialog, open the Windows Logs node and click the Application node. Any messages from Chorus Hub will show up in the right pane.

If you install Chorus Hub service on a machine when another machine on the network is already running Chorus Hub, The installation takes place, but when it tries to start the service, it fails and gives a red error icon in the Application log. When you click this you'll see a message, "Only one ChorusHub can be run on a network but there is already one running on …" The message will list the machine that is currently running Chorus Hub. The service will remain installed and set to Automatic. Thus when the computer is restarted it will try to start again and will succeed if the other Chorus Hub is no longer running. Otherwise it remains off.

Chorus Hub 2.5 service has the same limitations with multiple IP addresses on the Chorus Hub machine. It currently doesn't communicate this via the event log, but will cause a crash when someone tries to access Chorus Hub with an error report similar to this:

Msg: There was an error on the Chorus Hub Server, which was transmitted to the client.
…..
**Inner Exception:
Msg: Could not connect to net.tcp://192.168.196.1:5912/. The connection attempt lasted for a time span of 00:00:20.9952009. TCP error code 10060: A connection attempt failed because the connected party did not properly respond after a period of time, or established connection failed because connected host has failed to respond 192.168.196.1:5912

See Section 4.1 for further information on how to solve this.

2.4 LanguageDepot Internet startup

LanguageDepot is an Internet server (http://languagedepot.org) that hosts repos for various programs including FieldWorks. As of FieldWorks 8.0.6, in order to use the Internet option, a repo needs to be set up on the server prior to your use.

The first step is to have user accounts set up for each user that will be working on your project. You can do this by going to the site and clicking the Register link at the top right. You'll need to fill in a Login (preferably firstName_lastName) and enter a Password. Do not use a password you use for other secure operations since the password is not encrypted as it is used. You will need to use this information when you set up Internet Send/Receive from your FieldWorks project. Fill out the other required fields and then click Submit.

Next, you need to e-mail admin@languagedepot.org to have the repo created. Let the administrator know whether you need a FLEx repo and/or a LIFT repo for your work, and give the vernacular language and ISO-639 code (see www.ethnologue.com) used for your project. Give the user Login name of the manager for this project. The administrator will then send back a project id that you'll need to use in Send/Receive setup.

A project can have one or more managers. A manager can sign in to http://languagedepot.org using their user login and password. They can then type in their project name in the search box and click on their project link to access information on their project repo. From the Settings tab and the nested Members tab, any manager can add other users as Manager, Contributor, or Observer. A manager or contributor can send and receive changes in the repo. An observer can get the project, but cannot make changes. If you click the outer Repository tab for your project, you can see information on each S/R to your repo. Clicking the number at the left will list files

that were modified for that S/R. Clicking a file link will attempt to show changes that were made to that file, although this may be very slow due to the size of many files and speed of the Internet. There are better ways to get detail on the repo if you need it. The Tortoise Hg Workbench will be described later.

3 How it works, or why it doesn't

For the most part, changes from all colleagues will be merged successfully. However, remember the program has definite limitations when merging results between users. If you don't consider these limitations, you may get results that are baffling and disappointing.

When users change different things in the project, the changes will merge without problem. However, when users modify the same thing, this results in a conflict that needs to be resolved. To keep the S/R process functioning quickly, we don't want to stop and ask the user about each conflict before finishing the S/R. So instead, the process picks one of the two conflicting changes and then adds a merge conflict report describing the conflict and what the program did to resolve it. Most common conflict descriptions will be clear to the user. However, there are some things that can change that are internal to the FieldWorks model, and there isn't a simple way to explain this to the user, so it may just show some underlying XML in the conflict report in these cases. Sometime after the S/R, the user can investigate the conflict reports and decide whether the program's guess was good or not, and fix any that were incorrect. At this point it requires a manual edit to fix any conflicts that were resolved incorrectly. The conflict report has a link that will try to take you to the spot that changed in FLEx. Technically, if there is a conflict, the person doing the merge will win (except for deletions described later), but because it's not easy to determine which user will actually initiate the merge for a given object, it's best to assume that the winner is random. A conflict report is displayed after a S/R that results in conflicts. You can also see this report using Send/Receive…View Project Messages (for a FLEx repo) or Send/Receive...View Lexicon Messages (for a LIFT repo).

Conflict information is stored in special XML files (*.ChorusNotes) that are merged along with the data. The user can mark conflicts as resolved so they don't normally show in the conflict dialog, but there isn't any way in the program to actually delete the conflicts. So if you get a huge number of conflicts, these conflict files can get very large.

3.1 Lexicon examples

Merging entries is not based on the headword, but on an internal unique identifier (id) that is assigned to an entry when it is created. The reason for this id is that headwords do not uniquely define entries since you can have homographs, and homographs are not unique because they can be renumbered or adjusted when switching to a different writing system, etc. Entries can also have the same headword but have a different morpheme type (prefix, suffix, root). Headwords can also be misspelled, but when corrected, we don't want the entry to be considered a different entry, especially when lexical relations and interlinear text depend on it. The computer needs something that is created once when an entry is created and it will never change for the life of the entry. The technical name for this is a Globally Unique Identifier (guid). Anything inside FLEx that references this entry uses that id.

If I create an entry for 'house', and my colleague also creates an entry for 'house', each one will have a different id, so when we merge our projects using S/R, the result will have two entries for

'house'. Duplicate entries can be merged one at a time using the Tools...Merge with entry option, but it doesn't happen automatically. If two users take a list of words and both add them to the lexicon, after they are merged, as far as humans are concerned, there will be duplicates of every word. The computer knows they have different ids, so it thinks it did a great job of merging the entries. On the other hand, if one user adds words from A-M and another adds words from N-Z, and they merge their projects, everything will be great since the human and computer agree that these are different entries.

Likewise, the computer assigns a unique id for every sense that is created within an entry. Again, there isn't any other unique way a computer can identify a sense because some senses may have the same or missing gloss, definition, part of speech, etc. If I added a new sense for 'house' with a gloss 'political body', and my colleague did the same on his machine on the same entry, when we merge the projects, 'house' is going to have two identical senses from a human perspective, but the computer considers them different because they have different ids. These duplicate senses can be merged one at a time using the Merge Sense into menu option on a sense, but it is not automatic.

During the Send/Receive merge process, once the computer realizes two people changed the same entry or sense based on the id, then it will do the best it can at merging the changes within that 'object'. At least it knows it is working with the same entry or sense. If I add example sentences to senses, and my colleague adds semantic domains to senses, when we merge, the result will be exactly what we want. Likewise if I add Spanish glosses and my colleague adds French glosses, the merge will again be flawless.

If I change a gloss on a sense and my colleague doesn't do anything with that gloss, then when we merge, my change will be made in both projects. However, if I change a gloss for a sense, and my colleague changes the same gloss to something else, now when we merge we have a conflict, so the program will pick one change and reject the other, then display a merge conflict report after the merge.

3.2 General concepts

In FLEx, every 'object' has a unique id. Objects are defined in the underlying model of the data. Some examples of objects are lexical entries, senses, example sentences, etymology, pronunciations, allomorphs, grammatical info, lexical relations, reversal entries, interlinear texts, wordforms, wordform analyses, word glosses, parts of speech, notebook records, list items, etc. In WeSay only entries and senses have unique ids. The merge process basically adds any new objects, and merges the contents of existing objects with identical ids.

Now suppose I delete an entry or sense. As long as the other user does not modify this entry or sense, when we merge, the deletion I made will remain deleted. However, if my colleague changed something on that same entry or sense, the merge process doesn't want to potentially lose some important work, so the object I deleted will remain after the merge with the change my colleague made, and a merge conflict report will be added. So after the merge I may think the S/R messed up because the thing I deleted came back again.

When two or more people are working on a project, the longer the interval between S/R, the bigger the chance of having many merge conflicts. Naturally, if multiple users make a lot of changes to the same fields of the same entries or senses, a lot of merge conflicts will also result. So if you are working in the same areas, more frequent Send/Receives will be helpful. If you

plan to do extensive bulk editing or orthography changes, you can avoid a lot of conflicts if everyone does a S/R and then stops work until the extensive changes are made, then they all do another S/R before continuing their work to get the bulk changes.

3.3 Interlinear examples

Interlinear texts and charts are complex objects in FLEx with many sub-objects with ids and references pointing into the text via offsets. Even pasting text into the baseline will generate a whole new set of wordforms with different ids if the wordforms are new to the project. This is normally not a problem, but when using Send/Receive it presents major difficulties with merging if another user was working on the same text and will likely result in duplications of text and/or problems with interlinearization. So in order to avoid these potential merge problems, coordinate with your colleagues so that only one person works on a given text in any round of S/R operations, especially if one member makes any change to the baseline. Even adding or removing a segment break character (period, question mark, exclamation point, section sign) in the baseline will alter the underlying segment objects so that interlinearization of the affected segments from a colleague in the same S/R cycle may be lost.

Even when working in different texts and not modifying the baseline, two users doing interlinearization can create duplicate objects when working in the same S/R cycle. For example, if user A analyzes the word 'cat' for the first time, and user B also analyzes the word 'cat' in the same S/R cycle, when they merge, there will be two 'cat' entries in the lexicon as homographs, even though they have identical content. So one user will need to use the 'Merge with entry' option to combine these entries, then the 'Merge sense into' option to merge the two senses in the merged entry. Also, under Word Analyses, there will be two duplicate analyses for the wordform 'cat'. So one user will need to delete the duplicate analysis that isn't being used, or possibly use the 'Assign Analysis' option to move any analyses from one analysis to the other analysis.

Merges and deletions of analyses or word glosses in Word Analyses affect all interlinear texts that use the modified wordform. If another user makes use of the modified analysis or word gloss in the same S/R cycle, their analyses involving the modified wordform will likely be lost.

In summary, when working on interlinear texts, only one user should modify a given baseline in one S/R cycle. Any interlinearization that adds new entries, wordform analyses, or word glosses will produce duplicates if another user does similar interlinearization on the same wordform in the same S/R cycle. Any merges or deletions in the Word Analyses area will cause loss of analyses that were made to the modified wordforms in the same S/R cycle. Analyses that use existing entries, senses, and wordform analyses and glosses should survive S/R operations. Frequent use of S/R will minimize the chances of encountering these problems.

3.4 WeSay/LIFT collaboration

When collaborating between multiple FLEx users and one or more WeSay users, it's critical that only one FLEx user sync with the LIFT repo. This is because the logic for merging gets fuzzy when dealing with lexical relations, variants, complex forms, example sentences, etc. since these are not simple strings, but objects similar to entries or senses. In FLEx these each have unique ids, but the LIFT file used by WeSay does not record the unique id for these. Every time a S/R happens between a LIFT repo and FLEx, for any new objects other than entries and senses, FLEx

will give them a unique id. If multiple FLEx users sync with the same LIFT repo, each FLEx will have different ids for the same object, which would result in duplicate objects when the FLEx users merge via the FLEx repo.

Any program that uses LIFT files is supposed to read and modify what it understands, and leave the rest of the information intact for other programs that need it. FLEx uses a hidden LiftResidue field on several objects in the lexicon to store information that isn't in the FLEx model, so that it can be exported back to the LIFT file without being lost. Any time you import a LIFT file, every entry and sense will have a LiftResidue field added to hold this information. If you are only using FLEx to FLEx collaboration, the LiftResidue fields are excess clutter that can be deleted.

3.5 FLEx/ParaTExt collaboration

If a ParaTExt (PT) project is associated with a FLEx project, and you are collaborating with other users via PT Send/Receive (S/R), and the other users also have FLEx installed and are collaborating using FLEx S/R Project, and FLEx users are doing anything in the Texts & Words area, there are some cautions that need to be taken to avoid corrupting FLEx data.

A PT administrator is the only one that can associate a shared PT project with a FLEx project. Once associated, PT can access lexical data from FLEx, and FLEx can access current PT data in the FLEx Texts & Words area. The association is stored in the PT project settings file with the name of the associated FLEx project. These settings will go to colleagues during PT S/R. If the colleagues have a FLEx project with the same project name, they will automatically be connected with the PT project on their machine. If these users also use FLEx S/R Project to keep the FLEx projects in sync, there is potential for problems if they also use the Texts & Words area in FLEx.

Important! When you are using S/R with FLEx and PT, one user should load all of the data from PT, then go to Word Analyses. After doing this, he should S/R Project in FLEx. Then all other FLEx collaborators should do S/R Project to get all of this new data before they open the Choose Texts dialog in Texts & Words...Interlinear Texts. After this initial S/R cycle, it will work best if collaborators sync FLEx and PT projects at the same time. When new books are added to PT or significant changes are made, again, you should go through this same process with a single FLEx user initially getting the changes and then passing them around via S/R.

As with any interlinear text in FLEx where users are collaborating with S/R, you should avoid more than one user making changes in a given book during a single S/R cycle. For Scripture books, each section is treated as an interlinear text, so this caution would apply to each section in a scripture book.

Here is some technical detail to understand what is happening and what can go wrong.

When scripture is first loaded from PT by opening the Choose Texts dialog in FLEx and selecting scripture, for the books that are checked, it creates ScrBooks, ScrSections, ScrFootnotes, StText (for titles and headings), ScrTxtParas with ParseIsCurrent set to False, and Segments without Analyses. As part of the import, a checksum is made of the PT file and stored in ImportedCheckSum of ScrBook. Once loaded, when a user opens the Choose Texts dialog, if the stored checksum matches the current PT checksum, then FLEx will not reimport that book. If the book is modified in PT, and a portion of that book is included in the current Text & Words area, when the user goes to the Choose Texts dialog, FLEx will recognize that the PT book has

changed, so it reimports the book, trying to maintain any interlinearization that has been done on that book. These checksums are included in FLEx S/R.

When you click a scripture text in the Texts pane (actually a ScrSection, ScrFootnote, or StText for titles and headings) that text is parsed creating WfiWordforms and PunctuationForms as needed, setting default Analyses on Segments, and setting ParseIsCurrent on ScrTxtParas to True.

If you go to the Word Analyses (or Concordance) tool, it will parse all texts and scripture that is currently imported, creating WfiWordforms and PunctuationForms as needed, setting default Analyses on all Segments used in those texts, and setting ParseIsCurrent on all ScrTxtParas to True.

All of the objects above, except Analyses which are initially pointers to WfiWordforms or PunctuationForms, have unique IDs. So if two FLEx users import the same books in the same S/R cycle, there will be a lot of duplications that would be hard to clean up after a S/R merge. FLEx does not provide a way to delete duplicate scripture books or sections that get imported in this way. Although wordforms have unique IDs, they will actually merge during S/R if the form is the same, so that helps some.

The duplicated portions can result in the load process failing. This will continue until the book is cleaned up enough for the import process to work. With Translation Editor (TE) you could run a verse chapter check on a bad book which will usually show serious problems in the data. If you clean these up enough in TE, you should get to a point where the import will work again.

If something is messed up too badly, it may be necessary to delete a book entirely. This is best done using FDOBrowser. This is a low-level utility program that is in c:\Program Files (x86)\SIL\FieldWorks 8 directory. You should not have FLEx or TE open when using this program. To delete a book of scripture, you can type FDOBrowser at the Windows start command and it should find the program.

Once FDOBrowser is open, go to File…Open Language Project. In the open dialog, you'll have to navigate to your project *.fwdata file. Project directories are normally under c:\ProgramData\SIL\FieldWorks\Projects\. After the project is open, you'll see two other tabs. Click the LangProj tab then scroll down to TranslatedScriptureOA and click the + to the left to expand the node. Under the TranslatedScriptureOA node, open the ScriptureBooksOS node. You'll see a node for each book that has been imported from PT. To delete a book, right-click the book and choose Delete. When done, do File…Save Current Language Project and then close FDOBrowser. FDOBrowser does autosaves the same as FLEx.

Various things can go wrong when users sync FLEx and PT at different times and the user goes to the Choose Texts dialog. For example, if user A imports scripture and then does FLEx S/R, his import and checksum will go to user B. But if PT is not in sync with users A and B and user B does FLEx S/R and opens the Choose Texts dialog, they will reimport the scripture from B's version of PT data, and conflicts will likely happen the next time they do FLEx S/R with user A. Currently, the FLEx merge conflicts for scripture typically give information that is difficult even for technical people to understand. Also, the merge conflict hotlinks into FLEx do not work for scripture at this point. So it's best if colleagues work in a way that reduces or eliminates scripture merge conflicts.

By following the Important paragraph above, most conflicts will be avoided. If one user imports books from PT and then does FLEx S/R and PT S/R, and another colleague does FLEx S/R and PT S/R, their FLEx and PT projects will normally be in sync so that nothing will be reloaded by going to the Choose Texts dialog. However, if multiple users are importing scripture in one S/R cycle, the FLEx S/R merging capability may have trouble sorting everything out.

Interlinearization on Scripture is stored with each paragraph in a section. The entire section is loaded into FLEs as an interlinear text. When FLEx imports from PT, it will update the baseline text and then try to maintain the interlinearization of the revised text. If you make a lot of changes to paragraphs, and add and remove sections heads in PT, there is a good possibility that FLEx will not be able to keep up with the changes which means interlinearization may be lost. Also, if you delete a scripture book as described above to get past a messed up book, you'll lose all interlinearization in that book.

When we talk about interlinearization being lost, it doesn't mean everything is lost. There are 3 places that are affected by interlinearization. First, you may create a wordform in the wordform inventory and add analyses to the wordform. Second, these analyses are connected to entries and senses in the lexicon, which may involve creating entries or senses in the lexicon. Third, in the interlinear text itself, for each wordform in a paragraph you are linking to one of the analyses in the wordform inventory. When we talk about losing interlinearization, what we are actually losing is the pointers to specific analysis in wordforms which means it will default to the wordform. So the actual work in creating wordforms, analyses, entries, and senses are not lost and are still available when doing future interlinearization. So although you do lose work by deleting a scripture book (or interlinear text), recovering should be much faster because everything you previously created in the lexicon and wordform inventory is still there and you simply have to link the wordforms again. FLEx will propose analyses for the entire text, so the proposals can either be approved immediately, or adjusted to get the correct analysis.

The process of associating a PT project with a FLEx project was designed for one FLEx project being associated with one PT project. If you happen to have multiple PT projects for a single FLEx project, you should stick to associating a single PT project to the FLEx project. If you switch the associations around between different PT projects, it will likely confuse the process resulting in corrupted Scripture and loss of interlinearization.

3.6 Linked files

Pictures, sound files, and other linked files are included in the S/R process as long as they are stored under the default LinkedFiles directory inside the FieldWorks project directory. FLEx allows you to use an external directory for linked files which is advantageous if you have multiple projects that refer to a master set of pictures and sound files. But if you have chosen this approach, they will not be included in S/R. Also, because repo size and time for S/R, especially to the Internet, can become too great if you use high resolution pictures, sound files, movies, etc., the S/R process currently limits files to 1 Mb, and only accepts certain file extensions. For images, it accepts these extensions: bmp, jpg, jpeg, gif, png, tif, tiff, ico, wmf, pcx, and cgm. For audio, it accepts these extensions: wav, snd, au, aif, aifc, aiff, wma, and mp3. Anything that doesn't meet these requirements is skipped during S/R. A warning message will be given in the S/R log if files are greater than 1 Mb. Files with doc and txt extensions are included, but mp4 files are not.

3.7 FieldWorks version

When using S/R for a FLEx project, all collaborators should use the same FieldWorks data version. FieldWorks projects have a data model version number that is stored in the repo. Several versions of FLEx may use the same data model. S/R users can work with any version of FLEx that uses the same data model. If one user upgrades to a newer version of FieldWorks that has a newer data model version, and then does a S/R, their data will be stored in a new version track in the repo. Any other users with an older version will remain on the old version track in the repo. At this point, people on the older track will only see changes being made by others with the older version. People on the newer track will only see changes made by others with the newer version. Each person that does a S/R in this state will see a blue message in the S/R log that warns them there are different unmerged versions in the repo. Data on the older track will only be merged to the new track when one of the users on the older track upgrade to the new version of FLEx. Then the remaining users of old versions will need to upgrade as well so that everyone can keep in sync. Thus when you are using S/R FLEx collaboration, it's best for participants to upgrade to a new version together.

3.8 FieldWorks project name

Repos also have a unique id that is assigned when it is first used. The id stays with the repo when you do a S/R to different locations, or copy the repo. This is used to ensure that you are really working with the correct repo. During a S/R the program uses this repo id to find the project in the FieldWorks Projects directory. If one user changes the name of their FieldWorks project, they can still S/R with the same repo because the repo id hasn't changed. For this reason, if you want to receive a new version of the repo, you can't just rename the old folder or give it a different name in the Receive dialog. In order to do this, you need to actually move the old project folder outside the Projects folder, or move it inside a subfolder inside the Projects folder so that it won't cause a conflict. This also means that different colleagues can specify their own name to a project when it is received the first time, although this could be rather confusing.

4 Technical details

4.1 Chorus Hub and virtual machines

Chorus Hub 2.4 will give continuous warning messages, "this machine has more than one IP address", and may or may not work if it detects more than one IP address on your machine. Chorus Hub 2.5 is unable to display these warning messages, but has the same limitations. Instead, when a user attempts to access Chorus Hub 2.5 under these conditions, they will get a crash with a message similar to the following:

Msg: There was an error on the Chorus Hub Server, which was transmitted to the client.
…..
**Inner Exception:
Msg: Could not connect to net.tcp://192.168.196.1:5912/. The connection attempt lasted for a time span of 00:00:20.9952009. TCP error code 10060: A connection attempt failed because the connected party did not properly respond after a period of time, or established connection failed because connected host has failed to respond 192.168.196.1:5912

If the problem is due to a virtual machine, and you really need to use Chorus Hub on the virtual machine or on the machine that is running a virtual machine, there are ways to make this possible. Here are some notes for using Oracle VM VirtualBox on Windows 7. There are probably similar things that can be done with other virtual machines.

Virtual Box defaults to creating an IP address for the virtual machine. The default attachment is NAT which is a separate IP address that is not available outside the virtual machine. If you need to run Chorus Hub in the virtual machine, you can change this setting to Bridged Adapter. In this mode it uses the same IP as your main machine. So when you run Chorus Hub inside the virtual machine, users outside the virtual machine will be able to use it.

If you want to run Chorus Hub on a machine that is using a virtual machine, you'll need to disable the IP address for the virtual machine. You can do this using Control Panel, Networking and Sharing. Click "Change adapter settings", then right-click the Virtual Box Host-Only Network and choose disable. Now Chorus Hub will run on the main machine without complaining about extra IP addresses. You can still access the Chorus Hub running on your main machine from within the virtual machine.

4.2 Using FieldWorks backups and Send/Receive

Making FieldWorks backups is not as critical when using Send/Receive Project because the S/R process is storing a type of backup on the repo. If your machine should fail or something happens that makes your version of FLEx unusable, you can delete your FLEx project directory and start over again by using Get Project from colleague. This will restore your project to the current state of the repo.

If you are only using S/R Lexicon, then the repo only stores the lexical data, but none of the other data that may be in your FLEx project. So in this case, you should definitely do regular FieldWorks backups to prevent data loss in case of some emergency.

While making FieldWorks backups is perfectly safe when using S/R, you normally should not do a restore from a backup when using S/R. For one thing, if you restore your project from a backup and then do S/R, it will basically set all other users back to your restored state, thus cancelling all work they have done since the state of the project when the backup was made.

Also, a FieldWorks backup does not include the LIFT or FLEx repo that is normally inside your project directory. So if your project directory gets deleted, when you restore from a backup, you are now disconnected with both the LIFT and FLEx repos.

If you are using S/R Project, instead of restoring from a FieldWorks backup, you should delete your directory and use Get Project from colleague to restore your project. Then it would be in sync with the repo. If you have made a FLEx backup since your last S/R, the next section describes how you can restore from this.

If you are using S/R Lexicon, and you need to restore your project from a FieldWorks backup, then you should use Get Lexicon (WeSay) and Merge with this Project. But keep in mind that this will probably cause a loss of the WeSay work since the time the FieldWorks backup was made.

4.3 Restoring a FieldWorks backup

Restoring from a backup is usually not a good thing when using S/R, so to prevent accidentally losing a lot of data, the program currently refuses to let you restore to the project, and forces you to restore to a different name. Say your original project is QQQ and you restore to QQQ-01. There isn't a direct way to get QQQ-01 hooked back up to S/R. The only way you can get back to S/R with the restored data is to copy the QQQ-01.fwdata file from the QQQ-01 project to the QQQ project and rename it QQQ.fwdata, replacing the bad QQQ.fwdata and then continue S/R from the QQQ project. This will be safe to do IF the backup from which you restored was made since your last S/R. Otherwise this would likely cause loss of data for everyone on the project.

The normal way to recover if you mess up the project in a way that you don't want to pass on is to delete your QQQ project and then do Get Project From Colleague to get back to a normal S/R project. Note, if you have QQQ-01 in your Projects directory, you will not be able to do Get Project From Colleague because it searches all projects to see if any have the same unique project id stored in the fwdata file. This project id will be the same if you restore a project from a backup or rename the project. So you need to make sure there is no project in the projects directory that was made at some point from the project you want to download.

If you are going to do something potentially dangerous to a project that is connected with S/R, one approach is to first make a copy of the fwdata file in the project directory using Windows Explorer. If you need to restore, just delete the fwdata file and rename the copy back to the original name. But again, this should only be done if you haven't done S/R since the time you made the backup.

4.4 To switch a WeSay bridge user to another FLEx user

Remember that only one FLEx user should provide a S/R bridge between FLEx and WeSay users. Assume FLEx users A and B are using S/R Project to keep in sync, while user A is also doing S/R Lexicon to sync with WeSay users. Now they would like to switch the WeSay bridge operation to user B instead of A. It's probably best for other FLEx and WeSay users to avoid doing S/R during the following process.

1. User A does S/R Lexicon to get the latest changes from WeSay users.
2. User A does S/R Project to send these merged changes to the FLEx repo.
3. User B does S/R Project to get the latest from the FLEx repo.
4. User B does Send/Receive...Get Lexicon and merge with this project to get started as the bridge.

At this point user B would be the only one that does S/R Lexicon to function as the bridge between WeSay and FLEx users.

When done, user A should delete the *_LIFT directory inside their FieldWorks project folder under the OtherRepositories folder to make sure they don't accidentally do S/R Lexicon.

4.5 FLEx and WeSay compatibility issues

There are some compatibility problems with writing systems between WeSay and FLEx for non-standard ISO codes. Lists are maintained in different ways between the two systems and are not included in S/R. Custom fields are also stored in different ways.

One issue is an issue between meaning, gloss, and definitions. WeSay displays definitions in the Meaning field. However, if an entry from FLEx only has a gloss, WeSay will show the gloss in the Meaning field. But if you edit the meaning in WeSay, it is not actually editing the gloss, but adding a definition based on your edits. When you add a new sense in WeSay, it is not possible to enter just a gloss. Even though you have the Gloss field enabled, WeSay will not open fields on a sense until you type into the Meaning field, which fills in the definition. Once you do that, then it enables other fields including gloss. It won't even show the Gloss field in a new sense without first clicking Hide Uncommon Fields, and then Show Uncommon Fields. These limitations make it difficult if you only want to enter glosses in entries. At this point it would be best to do all of your work in FLEx and WeSay in the definition fields, and then at some point bulk edit these to glosses.

There are similar problems between word, lexeme form, and citation form. WeSay displays the lexeme form in the Meaning field. Although you can create a new word with only the lexeme form, the list of entries only uses the lexeme form.

THIS NEEDS MORE WORK!

CAUTION! UNDER CONSTRUCTION!

****

Describe how to reset S/R for FLEx or WeSay

Describe how to make sure everyone gets the same data with multiple FLEx/WeSay users.

Describe connections with PT and how there may be damaging conflicts when more than one person uses the interlinear chooser while doing S/R both in FLEx and OT.

Describe new Chorus Hub service.

Describe how to connect an existing FLEx to a WeSay project, if not already covered.

****

> Please e-mail this to flex_errors@sil.org Message (not an exception):

> Syncing...

> Executing: recover

> standerr: *** failed to import extension fixutf8 from C:\Program Files

> (x86)\SIL\FLEx Bridge\MercurialExtensions\fixutf8\fixutf8.py: [Errno

> 2] No such file or directory no interrupted transaction available

Problem and solution:

Each repo has a hgrc file inside the .hg hidden directory. This file contains a string like this:

fixutf8 = C:\Program Files (x86)\SIL\FLEx Bridge\MercurialExtensions\fixutf8\fixutf8.py

The path has to be a valid path to fixutf8.py on your computer.

When a repo is cloned (e.g., Get project from colleague), this line is initialized in the hgrc file to the correct path to this file. Once set, this line normally isn't changed.

I assume you either copied your project folder from your Windows machine, or you might be using the same folder you used on Windows in Linux?

In either case, the repos were cloned while in Windows, so this line is set to the Windows path in this file. However, that path is different on your Linux machine. So the solution is to edit these two files:

/home/andrewcarson/.local/share/fieldworks/Projects/Tampuan/.hg/hgrc

/home/andrewcarson/.local/share/fieldworks/Projects/Tampuan/OtherRepositories/Tampuan_LIFT/.hg/hgrc

Replace the line with this (make sure this path and file exist on your machine)

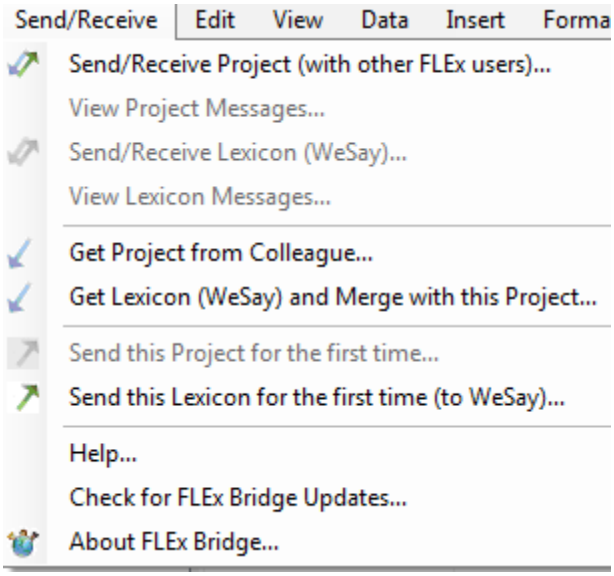
fixutf8 = /usr/lib/flexbridge/MercurialExtensions/fixutf8/fixutf8.py

Need to describe what happens when users have different versions of FLEx and use S/R. They can still do S/R, but their changes stay in a different track until the older person upgrades to the same version, then the dual tracks will be merged. When a user makes a change and that change gets to the other user, the log will have a blue message warning of the different versions. The warning will not happen again unless the other user makes a change.

We do not support sharing a common FLEx project directory in Windows and Linux if using S/R.

(********** Finish updating the menu commands below. Also mention that FLEx-WeSay can be done on one machine, but FLEx to FLEx must be on separate machines. Also explain how to choose writing systems when doing FLEx/WeSay collaboration**********)

8.0.5 S/R menu:



4.? Unfinished notes

Explain difficulties of Meaning in WeSay vs Gloss in FLEx.

The FLEx Send/Receive process uses a Mercurial repository (repo) for storing data. A local copy is stored in a hidden Mercurial .hg directory inside your FieldWorks project directory and contains a split version of the fwdata file as well as files from your LinkedFiles, WritingSystemStore, and most of your ConfigurationSettings directories. The fwdata file is split to reduce the size of files so that they work more efficiently in the Mercurial repository.

Here is the basic process that happens during a FLEx S/R:

1. The fwdata file is split into various XML files and subdirectories in the project folder
2. All of these split files are committed to the local repo in one change set. It only stores changes that occurred since the last time.
3. It pulls any new change revisions from the remote repo (USB drive, Chorus Hub, or Internet).
4. Each change set in the repo records the immediate change set it comes from, plus a change set that was common to

only some picture/audio files are included.

There is another unique id when a repository (repo) is created. Send/Receive will only allow merges to take place when two repos have identical ids. This is why when two users start with the Send/Receive process, only one user must initialize a repo, and then additional users connect to that existing repo. Instead, if I initialize a Send/Receive repo on my computer, and another user with a copy of my project creates a Send/Receive repo on his computer and we try to merge via Send/Receive, the process will refuse to merge because the repo ids are not the same.

FieldWorks backup and restore are very important since you could lose all your work if some catastrophic problem occurs in the program, or power fails at the wrong time, or if the computer

fails in some way. When this happens, you can restore the project to the last backed up version and lose a minimum of work. A FLEx S/R includes the entire FieldWorks project. Since a LIFT S/R only uses the lexicon, it will not provide backups for anything other than the lexicon. Be very cautious about restoring a FieldWorks backup when you are using Send/Receive with colleagues. If you restore an older FieldWorks backup and then do Send/Receive, all of the changes since that backup will likely be made on your colleague's machines during their next Send/Receive.

In FieldWorks 7.2, once a Send/Receive has been initialized, either by starting a new repo or connecting to an existing repo, each computer stores a mapping between a unique FieldWorks project id and a repo id. Even if the FieldWorks project is renamed, the internal id is unaffected, so the mapping of the renamed project will still be to the original repo. Special measures must be taken outside of FieldWorks to break this mapping, if needed.

There is also a unique id for a FieldWorks project. If I create a FieldWorks project for Spanish and create some entries, and my colleague also creates a FieldWorks project for Spanish and adds some entries, these cannot be merged via Send/Receive because the projects have different unique ids.

Repositories used with Send/Receive also have unique ids that are assigned when they are created. So even if my colleague has a backup of my FieldWorks project, if we independently create a repository using Send/Receive, we will not be able to use Send/Receive to merge our data. To avoid this problem, when two users start with the Send/Receive process, only one user must create a project and initialize a repo, and then additional users connect to that existing repo. In this way the project and the repo used by both users will have the same unique ids, so Send/Receive can be used to merge data between the projects.

FieldWorks backups are very important since you could lose all your work if some catastrophic problem occurs in the program, or power fails at the wrong time, or if the computer fails at the wrong time. When this happens, you can restore the project to the last backed up version and lose a minimum of work. Be very cautious about restoring a FieldWorks backup when you are using Send/Receive with colleagues. If you restore an older FieldWorks backup and then do Send/Receive, all of the changes since that backup will likely be made on your colleague's machines during their next Send/Receive. FieldWorks backups store the current state of the project, but do not include all of the historical data in the repo.

A repository contains a history of sequential Send/Receive operations, so it also provides a type of backup of your data. With a repository, it's technically possible to restore a project to any point in this sequence. So far this is not possible within the FieldWorks program, but requires some technical assistance.

The FLExBridge Send/Receive process includes all of the data in your FieldWorks fwdata file and your writing systems. Pictures and sound files are included if they are stored in the LinkedFiles directory within your project folder and are not over 1Mb in size. At this point, all user settings are unique to each machine, so they are not included in the Send/Receive operation.

Note that FLExBridge will not work when you are using the shared project option that allows users to work on a single project within a network. If you are using shared mode, you'll need to

switch to unshared mode before using Send/Receive. Our goal is to make Send/Receive work well enough that using the shared mode is no longer necessary.

Why can't I rename a FLEx folder and have it work?

xxxx

using LiftTools to 'merge homographs' to merge senses. Can't use normal LIFT import UI methods to merge senses, but can do S/R LIFT to USB, then replace the lift file in the usb repo (using various hg commands and setting the username, then do another S/R which will force FLEx to match LIFT content.

Any time you do a LIFT import, you 'corrupt' the FLEx project with LiftResidue fields that are useless to FLEx S/R. At least one for every entry and sense. Most of the clutter is because LIFT uses a string for the entry/sense id which is typically a guid, but not always, so FLEx needs to keep track of this string to insert it back in the LIFT file on export. I have a CC table that strips these out.

If one user does an initial S/R and a second user already has the same FW project on their machine and they try to do a S/R without using the initial "Receive from Colleague" approach, the second S/R will fail with a message similar to, "The repository that you tried to synchronize with has the same name as yours, but it does not have the same heritage, so it cannot be synchronized."

Bug in FW8.0.2 that can damage a repo

Scenario 1
1. User A does initial S/R Project to create a FLEx repo
2. User B does get from repo and gets the FLEx project from A
3. User A does initial S/R Lexicon to create a LIFT repo
4. User B is confused, and instead of choosing S/R Get and Merge Lexicon to the LIFT repo, they choose S/R Lexicon. The program tells them they can't do this, but in the meantime, it actually creates a disconnected head in the LIFT repo and trashes it.

Scenario 2
1. User A does initial S/R Lexicon to create a LIFT repo
2. User B does get from repo and gets the LIFT project from A
3. User A does initial S/R Project to create a FLEx repo
4. User B now wants to use the FLEx repo, but is confused so does S/R Project. The program tells them they can't do this, but in the meantime, it actually creates a disconnected head in the FLEx repo and trashes it.

If you happen to get into this state, you'll need to request help to recover. These problems are written up in LT-14694 and should be fixed in FW8.0.3 (FW 8.0 Beta 4), or an updated version of FLExBridge.