



Version 3.1

ADDENDUM

- Introduction
- Enhancements
- Requirements
- Installation
- Duplicate Database
- Serial Numbers
- Enhanced Queries
- Improved Go Slip
- Improved Alpha Tabs
- Single Record Views
- Prefs Reorganized
- Column Alignment
- Print Current Record
- Enhanced About Slip
- Time Fields
- Assimilate
- Other Enhancements
- Bug Fixes
- Limitations

Introduction

This document provides information on Leverage version 3.1. It is an addendum to the Leverage Reference Manual, which is the primary reference for Leverage. Version 3.1 is a fairly significant enhancement to Leverage providing many new features and improvements to existing features. Leverage 3.1 will run on any Newton OS 2.0 or later device.

Enhancements

Leverage version 3.1 introduces numerous enhancements to the program. Here is a partial listing:

- duplicate database
- improved control over the placement of databases (i.e., on internal memory or card)
- no more “field defs are corrupted” message
- auto-increment or serial number field type
- multiple conditions on single fields when defining queries, useful for range tests (e.g., date \geq 1/1/98 and date \leq 12/31/98)
- more flexible, “functional” query definition (in addition to the original query by example)
- time fields (i.e., date/times without the date part)
- improved Go slip
- improved navigation within databases using new alpha tabs
- single record views (collections of fields in single record view)
- prefs extended and reorganized
- set type size for single record view, list view, and text fields
- column alignment in list view
- print only current record option added to single record print slip
- more robust handling of index failure

- character conversion on import and export (i.e., proper handling of 8-bit characters as found in München and Montréal)
- remember network connections
- enhanced “About this db” slip
- edit text fields in list view
- report sync overwrites
- tab between fields on NOS 2.1 machines
- easier control over exception logging
- offer to erase all Leverage dbs on scrub
- generally better error handling
- several bug fixes

Requirements

Leverage version 3.1 requires version 2.0 of the Newton operating system or later. It will run on the MessagePad 120 with Newton OS 2.0, the MessagePad 130, 2000 or 2100, and the eMate 300. It makes use of whatever size screen is available. The application itself requires about 460k of storage. Additional storage requirements depend on the size of user databases.

Installation

There is not a great deal to the installation of Leverage’s support software on your desktop machine. Just copy it onto your hard drive. If you have a Mac you may want to copy the items from the folder named “put contents in Extensions” into your System folder (the Finder will route them to the Extensions folder for you), particularly the four tools (“Text Tool”, “Serial Tool”, etc.) It is best to copy them one at a time to be sure you aren’t overwriting newer versions (i.e., if the Finder says there is a newer version already there and asks whether you want to replace it, say no). These tools or their equivalent

are necessary to make use of the FileMaker connectivity scripts. (AppleScript is necessary too, but you probably already have that.)

If you are using a Windows machine there is no system software to install, so when you've copied the diskette to your hard drive you're done.

As for installing Leverage itself onto your Newton device, a fresh install is just like installing an update, but without the trouble of deleting the earlier version, so refer to the next section and ignore the parts about removing the previous copy of Leverage.

Updating a Previous Version

The right way to install Leverage 3.1 over an earlier version or one of the pre-release copies if you want to save previous databases and data and to use it in the same configuration as it is (e.g., the current version is installed on a card and you want the new version there as well) is to:

1. Backup your Newton device, including a card if present
2. Delete the earlier version from the Extras drawer (scrub it or select it and choose delete), leaving cards in place
3. Use the Card app (in the Extras Drawer) to set the preference to where you want the Leverage application (e.g., it might be that you were keeping Leverage in internal memory but new data was being placed on a card, in which case the preference would need to be changed for the actual installation), leaving cards in place
4. Install Leverage with the Package Installer, Connection Utilities, Backup Utility or whatever you use to install packages, leaving cards in place.

The reason for leaving cards in place is to ensure that the new version of Leverage can see all the database definitions from the previous version and thus won't create any unnecessary (and potentially annoying) duplicates. This is less of an issue than it used to be, since Leverage now checks and recreates its own internal databases if necessary when installed (or when the machine is reset)

Nevertheless, it's still not a good idea to do anything like the following sequence: pull card, install Leverage (possibly scrubbing the previous version first), put card back in.

If you don't want to leave the previous data intact (i.e., you'd like to start fresh, with no databases), just follow the procedure detailed above, but first erase your existing databases by starting Leverage 3.0, going into its preferences, and tapping the "Erase Databases" button.

Duplicate Database

Version 3.1 of Leverage adds the ability to duplicate databases. You can copy or move all or a portion of the records in a database to a new or existing database. This could be used to archive old information (you would probably move a selection of the records in this case) or to use an existing database as a starting place for a new one (you might choose to copy only the definitions to a new database and then make changes to them). Note that if you copy or move records to an existing database, the fields must match.

To duplicate the current database (or copy or move a selection of records) choose "Duplicate" from the database tab in the upper left of Leverage's screen.

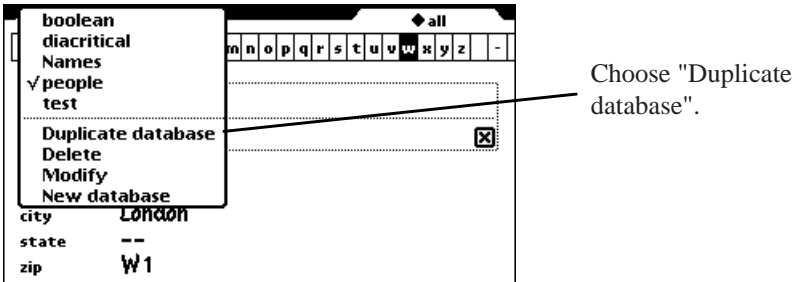


Figure 1 — Duplicate Database

Enter a new name to create a new database or choose an existing database as the target.

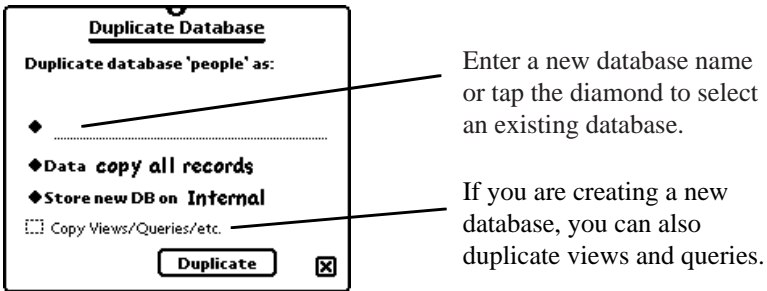
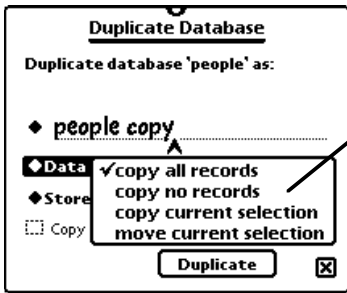


Figure 2 — Name the target database

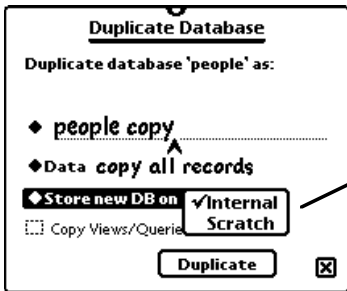
Next choose what to do with the records. You can copy the entire database, copy only the current selection, copy no records (i.e., copy only the definition), or move the current selection. (For what it's worth, this is somewhat inconsistent with usual Leverage practice, which would be to provide copy current selection and move current selection and leave it to the user to ensure that the current selection was all records or no records or whatever they wanted copied or moved.)



Choose what is to be copied.

Figure 3 — What to copy

Next select which store on which to place the moved or copied records and definitions. If your target is an existing database, the store where it is currently located will be selected, but you can change this selection (although that is probably not a good idea, since it will fragment the database — see the section of this addendum on control of database placement for a further discussion of this issue).



Choose where to store moved or copied records.

Figure 4 — Where to store

If you are creating a new database, you can also choose to duplicate supporting information by checking “Copy Views/Queries/etc.” as shown in figure 2 above.

When you are satisfied with the target database, the target store, and the disposition of existing records, tap the “Dupli-

cate” or “Move” button in the lower right of the slip (the label on the button changes depending on whether you are copying or moving the data).

Control over Database Placement

Leverage version 3.1 adds control over the placement of databases. You can file a database on internal memory or on a card. Then, when new records are added they will be placed in the same location. The field definitions and also any query or view definitions are placed in the same location as the data, so all the information specific to the database is kept in the same location.

Note that this control does not apply to database creation during import or synchronization, although it could be argued that it should. The details of this new feature can be found below in the section “Enhanced ‘About this database’ slip”.

Serial Numbers

Leverage 3.1 adds support for fields that automatically increment as new records are added. For example, if you are taking surveys you may want each survey to have a unique number. Define a serial number field for the survey database and as each survey record is added it will be given a field value one greater than the previous survey added.

This feature is controlled through the default value for the field (which in turn is set in the Create/Modify database slip).

Recall that the default value of a field tells Leverage what to place in the field when creating a new record. If you give a field a default value of 1, each time you tap the “New” button the new record Leverage creates will have a value of 1 in that field. This is old news. Now for the new feature.

If you give a field a default value of “++1”, then the first time you tap the New button, Leverage will give you a new record with 1 in the field. The next time you tap the New button Leverage will give you a record with a 2 in the field. The next time a 3, and so on. The “1” told Leverage to put a 1 in the next record created. The “++” prefixing it told Leverage to increment the default value each time a new record is created. (So if at this point, after creating three records, you go into the Modify database slip, the field in question will show a default value of ++4. The 4 is the value that will be given to the next new record. The ++ tells Leverage to continue incrementing the default value each time a new record is added.)

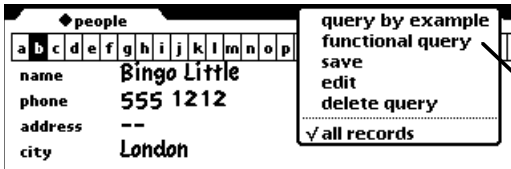
Returning to our survey example, if you want the surveys to start with 1 and go on from there (i.e., 1, 2, 3, ...), give the appropriate field a default value of “++1”. If you’re creating work tickets and you want them to start with 1001, indicate a default value of “++1001”. The first you create will be 1001, the next 1002, and so on.

Note that the field must be a numeric field (integer or real) and that the increment is always 1

Functional Queries

Version 3.1 adds support for functional queries — queries specified by defining a function that is applied to each record rather than by filling in a sample record. Functional queries are not such a trivial matter to define as QBE (query by example) queries, but they are more powerful (at least, they are more powerful than the QBE that Leverage supports). In particular users can define queries with OR-d conditions (e.g., `smokes = true or bloodPressure > 140`) and queries where the condition depends on comparing or combining two (or more) fields (e.g., `salePrice - cost > 100`).

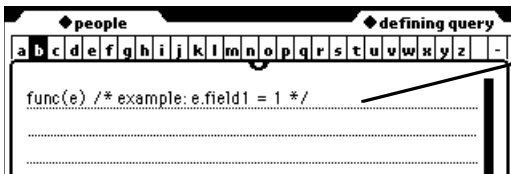
Define a functional query by choosing “functional query” from the Query pop-up.



The usual Leverage query is begun by choosing "query by example". To define a functional query, choose "functional query".

Figure 5— Starting a functional query

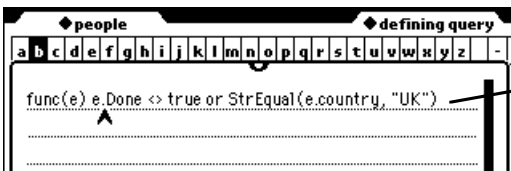
Leverage presents the same window you use to define code with a skeleton of a function already in place.



Leverage presents the code window with a skeleton of a function already defined.

Figure 6— Leverage presents an example function

Replace it with a NewtonScript function to be applied to each record to test whether it should be selected.



Replace the skeleton function with your own.

Figure 7— Replace the example function with your own

The function takes a single parameter: the record being tested. Field values are referred to via this parameter. For example, to select records for which the age field is 35 define a function

like:

```
func(e) e.age = 35
```

You wouldn't actually use a functional query for such a simple example. We're just starting with something simple.

To select records for which the state field is "TX" define a function like:

```
func(e) StrEqual(e.state, "TX")
```

See the Leverage Code Reference for more information about comparing strings and the StrEqual function. Both of the above examples would more appropriately be handled with a QBE query. But to select all records for which the state is "TX" or the age is 35, you'd need to use a functional query, defining a function like:

```
func(e) e.age = 35 or StrEqual(e.state, "TX")
```

As a final example, you might have an inventory database with salesPrice and cost fields, among others. To select records for which the salesPrice is greater than the cost by 100 or more (which might correspond to the high-margin items), define a function like:

```
func(e) e.salesPrice - e.cost >= 100
```

Multiple Conditions on Single Fields in Queries

Leverage has always supported multiple conditions in queries. For example, if you want to find all companies located in Germany with more than 50 employees, you can define a query as Country = "Germany" and employees > 50. However, in previous versions of Leverage, since the query was defined by filling in a blank record, only one condition could be placed on

each field. This made it impossible to find, for example, those companies with more than 50 employees and fewer than 100. With version 3.1 typical queries are still defined by filling in a record, but multiple conditions can be placed on a single field by tapping the “New” button while defining the query. This gives you a new blank record to work with. Thus you could look for companies with more than 50 employees and fewer than 100 by first entering employees > 50, then tapping “New”, then entering employees < 100. You can move from condition to condition with the up and down arrows, just as you move from record to record when looking at data.

Suppose we want to find all the people in a database whose names begin with “B” or “C” (i.e., name >= “B” and < “D”). Start by choosing “query by example” from the query pop-up (as noted above, with the introduction of functional queries there are now two choices for beginning queries on the query pop-up).

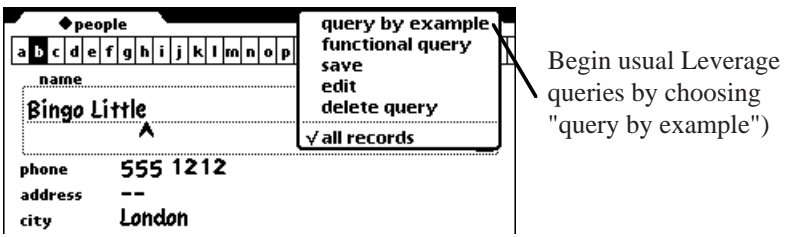


Figure 8— Starting a query

Enter the first condition.

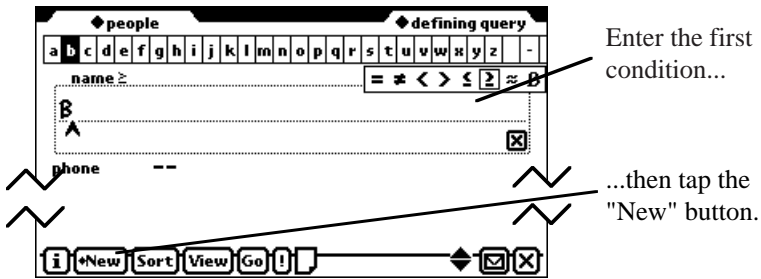


Figure 9— Condition one

Then tap the “New” button to present a new blank sample record, and enter the second condition.

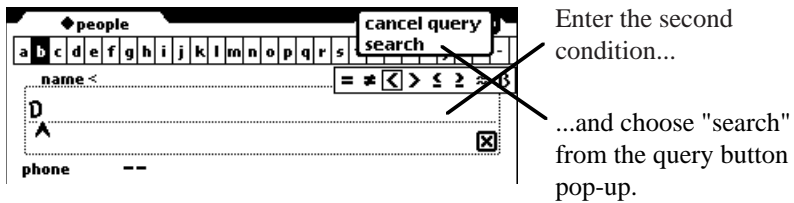
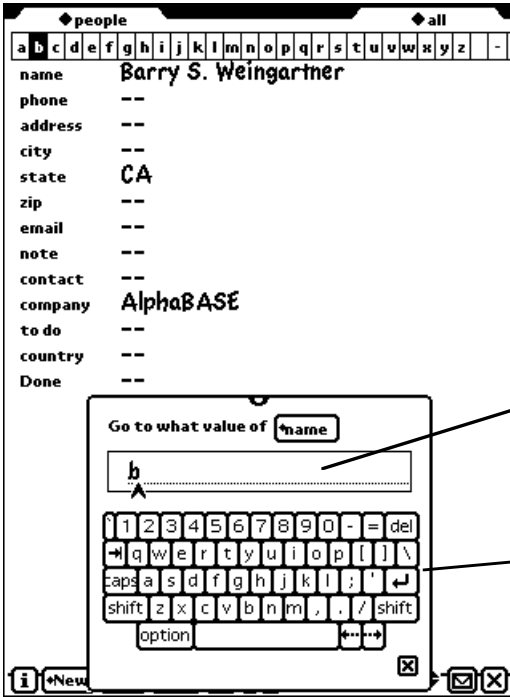


Figure 10— Condition two

Tap the query button and choose “search” as usual to complete the query.

Improved Go Slip

The Go slip used for moving quickly from one part of a database to another has been improved in two ways for Leverage 3.1. First, integral on screen keyboards reasonably appropriate to the current sort field have been added to the slip. Second, Leverage responds immediately as characters are entered if the sort field is a string field. Enter “B” and Leverage takes you to the beginning of the B-s.

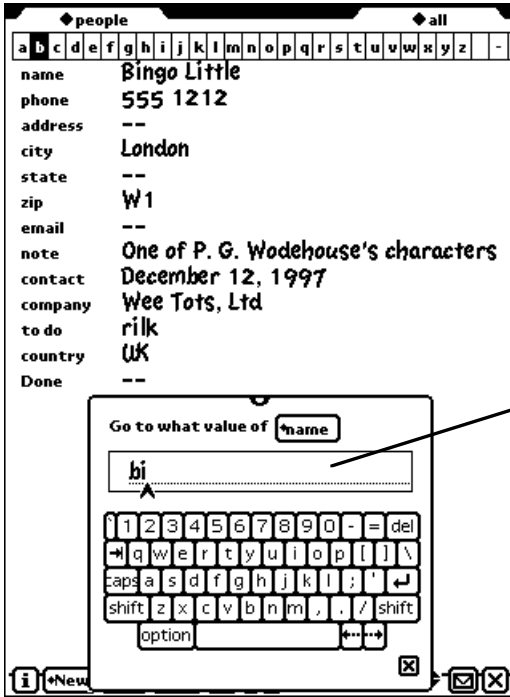


Enter "b" to go to the beginning of the B-s.

Use the integral keyboard if you like.

Figure 11— Go Slip

Then enter “i” and Leverage takes you to the first record beginning with “Bi” (or “BI” or “bi” — the case doesn’t matter). If you have thousands of records, you can just enter enough to get to the record you’re interested in and then dismiss the Go slip.



Enter another letter ("i" in this case) to move further along the current sort field.

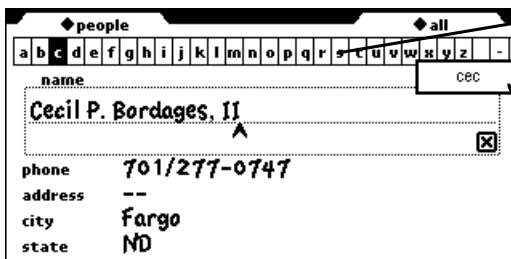
Figure 12— Enter as many letters as you need

Note that this technique (immediately moving through the database in response to partial entry) is not used for date or numeric fields. In the case of date fields the most significant part of the date (i.e., the year) is entered last in most locales. In the case of numeric fields, even though the most significant part is entered first (i.e., the first digit represents the largest part of the number) it's not possible to tell how large the number is going to be until it's finished (e.g., 30 and 300 both start with 3). For these field types the Go slip still has a "Go" button. Just enter the value you want to go to and tap "Go".

Improved Alpha Tabs

The alpha tabs accomplish essentially the same thing as the Go slip for alphabetic fields, but with a different interface. They have always responded immediately to taps, but in previous versions of Leverage it was impossible to specify more than the first letter of the record you wanted. If you had thousands of records and wanted to get to one beginning with, say, “Tr”, tapping the “T” tab (actually, tapping the “ST” tab and then tapping it again to indicate that you really wanted “T” rather than “S”) probably wouldn’t get you very close.

With 3.1 the alpha tabs have been changed to one tab per alphabetic letter. Leverage immediately moves to the nearest record as you tap letters: tapping ‘c’, ‘e’, ‘c’, ‘i’ might get you close to “Cecil” in a database sorted by name. Letters are added to previous letters if they happen soon enough, and a small slip appears for a moment after each tap on the right of the screen just below the tabs, showing what you’ve tapped so far. After 10 seconds has passed or if you move to another record (using the arrows) the letters are reset. Thus, tapping ‘c’, ‘e’, then pausing 12 seconds, then tapping ‘c’ takes you back to the first record starting with “C”, not to “Cec”. You can clear the letters without waiting ten seconds by tapping the ‘-’ tab at the right of the tabs.



When using the new Alpha tabs, tap several letters to move to the record you want.

A small window will appear temporarily to help you keep track of what letters you've tapped so far.

Figure 13— New alpha tabs

The tabs take a little getting used to and occasionally don't do what you expect (usually because previously tapped letters haven't "expired" yet), but it's very handy and takes up very little screen space. You can actually control the time it takes a tap to "expire" by a preference setting (i.e., 10 seconds is just the default). By setting the expiration time very short you could recover the behavior of the version 3.0 tabs.

Single Record Views

You've always been able to select which fields you wanted to see in list view — to name collections and arrangements of fields and columns for future use. With version 3.1 you can do the same with single record views: choose a selection of fields in a particular order, save it under a name, and recover it later. This will be particularly useful if you have databases with very many fields. You can group fields by the situations in which they are used or referred to and save the groups under an appropriate name. A customer database could have one set of fields related to sales contacts, another set for service, and another for accounting.

You define single record views by selecting "Columns" from

the “View” button, selecting which fields to display, and then specifying a view type of “Single” rather than “List” before tapping the “Apply” button.

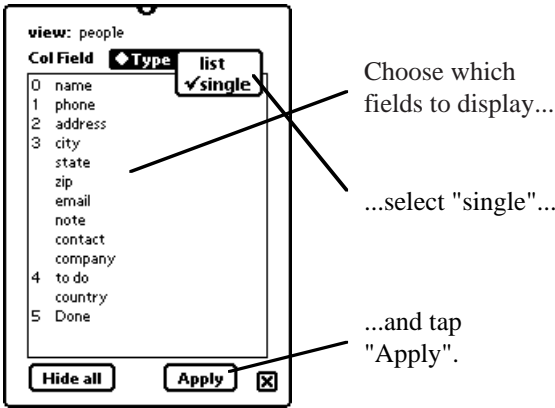


Figure 14— Defining a single record view

As with list views, save the view by name if you’ll want to reuse it. The saved view will appear on the View button with an icon alongside it that is intended to suggest single record view.

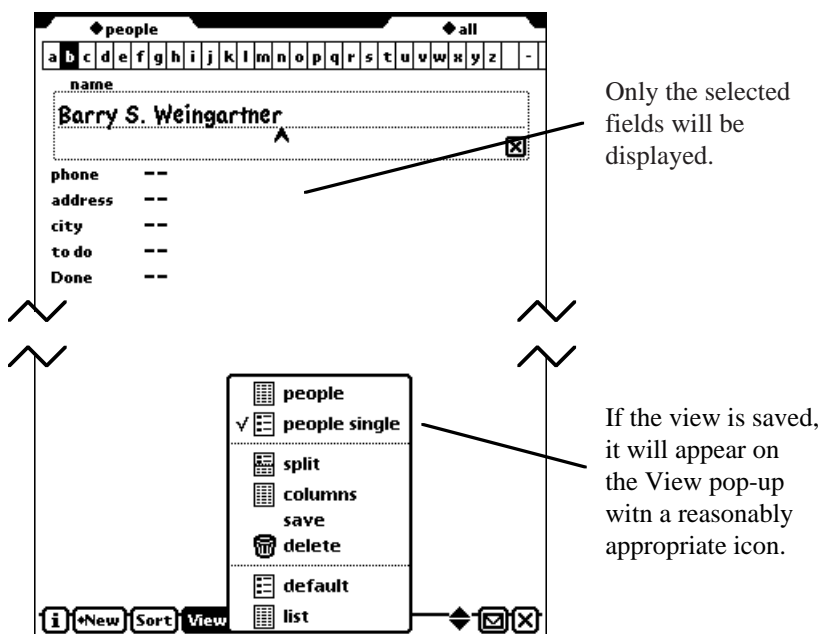


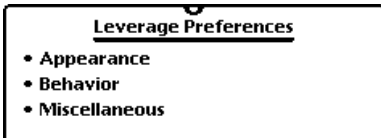
Figure 15— Saved single record view

Prefs Reorganized

For version 3.1 Leverage's preferences have been extensively reorganized. The most obvious change is the grouping of preferences into three categories: appearance, behavior, and miscellaneous. As the categories are intended to suggest, preferences are grouped into those that affect how Leverage looks, those that affect how it behaves, and those that don't fit well in these two categories. This grouping of preferences was necessary because there are quite a few more preferences in Leverage 3.1 than in previous versions.

Tap on a category to expand it and list all the preferences in it. You can also scroll through the preferences with the arrows and

expand and collapse them with the overview button.

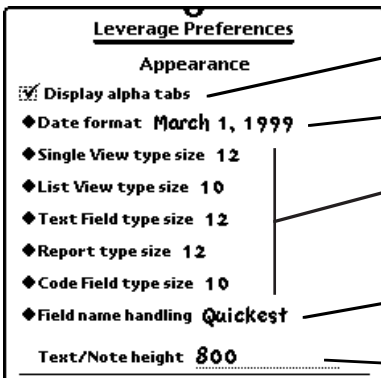


For 3.1 Leverage's preferences have been divided into categories. Tap on a category to see the preferences in it.

Figure 16— Preferences

Appearance Prefs

Appearance prefs include those that control the type size Leverage uses to present data. There are more of these than there were in version 3.0.



Does Leverage display the alpha tabs?

How are dates formatted?

What type size does Leverage use in various places?

How are field names capitalized?

How tall is the scrollable area for text and note fields?

Figure 17— Appearance preferences

In addition to controlling type sizes in various parts of Leverage, appearance preferences control whether the alpha tabs are displayed (note that you can no longer set this preference from the Go slip), what lame scheme to use to control the capitalization of field names (given that Leverage doesn't remember how you actually want it capitalized letter by letter), how large the

underlying scrollable area is in text, note, and code fields, and what format to use when displaying dates. The only one of these that is new for version 3.1 (aside from the additional control of type sizes, which is fairly self-explanatory) is the control over date format. The three choices are:

- short (something like 12/31/99, which is how the last day of 1999 is represented in some countries),
- short with four digit years (e.g., 12/31/1999),
- kinda long (e.g., December 31, 1999)

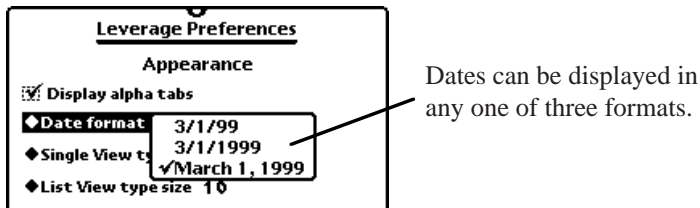


Figure 18— Date formats

Of course, the dates are localized, so if your locale displays the day of the month before the month and you choose the second format (or one of the others) it will display the parts in the right order (31/12/1999 or whatever). You are gently advised that it might be a good idea to use the short or kinda long format, as the short with four digit years format is frankly a hack.

Note that none of this really has anything to do with the Y2K problem, since the Newton OS, NewtonScript, and Leverage maintain dates in an internal format that has no trouble with the year 2000. It is necessary only if you are entering dates in the early 1900s (birthdates, for example), so you can tell whether 1/3/02 refers to 1902 or 2002 (By default it refers to 2002, but that isn't obvious). If all your dates (the dates of events you are recording in Leverage as they happen, say) are within the past 50 through next 50 years or so you can just leave the date

display in the default short format.

Behavior Prefs

Behavior prefs include those that control how Leverage responds to taps and which field opens as one changes records. Three behavior prefs are new for Leverage 3.1.

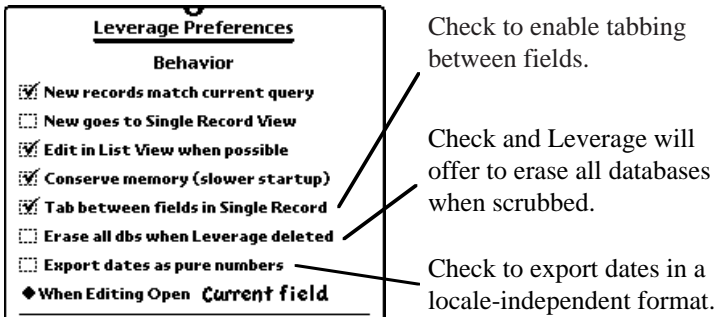


Figure 19— New behavior prefs

Check “Tab between fields in Single Record” to enable moving from field to field with the tab key in single record view. This happened automatically in older versions of the Newton OS and older versions of Leverage, but Newton OS 2.1 made changes to the way keyboards were handled that broke the old behavior, hence this pref. If for some reason you want to actually enter tab characters in fields (not generally a good idea, particularly if you’re going to be exporting the data to a desktop machine), or never use a keyboard (so it isn’t relevant) uncheck this pref.

Check “Erase all dbs when Leverage deleted” and Leverage will offer to delete all its databases when it is deleted from the Extras drawer.



Scrub Leverage from the Extras Drawer and the Newton OS will ask you to confirm that you want to delete it. Tap OK to confirm.

Figure 20— Confirm deleting Leverage

Once you've confirmed that you really want to delete Leverage, Leverage will ask whether you also want to delete all its databases (if you've checked "Erase all dbs when Leverage deleted").



If you confirm that you want to delete Leverage, the Newton OS begins the deletion...

...then Leverage asks whether you want to delete all Leverage databases as well.

Tap "Preserve" to delete Leverage but keep the databases.

Tap "Erase" to delete all Leverage databases along with Leverage.

Figure 21— Erase all databases?

Because of the potentially destructive nature of this preference, there are two additional safeguards beyond the confirmation dialog shown above. First, the preference has no effect unless you are in expert mode. Second, the preference is cleared every time Leverage is run. Thus, to erase all Leverage data-

bases:

- start Leverage and go into Prefs
- turn on expert mode and check “Erase all dbs when Leverage deleted”
- quit Leverage and delete it from the Extras Drawer
- confirm that you really want to delete Leverage
- confirm that you want to permanently erase all Leverage databases as well.

This is the recommended procedure for completely removing Leverage from your Newton. With the introduction of this preference the “Erase databases” button has been removed from Leverage’s Prefs. If you need to remove Leverage databases without removing Leverage itself you can use LevRename, found in the Tools folder on the Leverage distribution diskettes.

Check “Export dates as pure numbers” to export dates using Leverage’s internal format of number of minutes since midnight on January 1, 1904. The purpose of this preference is to avoid problems with varying or inconsistent locale dependent date formats. For example, if the desktop program expects dates to look like “13 Dicembre 1999” it may not know what to do with “12/13/99”. Of course, the desktop program will need to be able to handle “50464800”, which is the locale independent form of that date. LevConnect, LevLink for Windows, and the LevToFM and FMLevSync scripts all understand the format.

Miscellaneous Prefs

Examples of miscellaneous preferences include those that control Leverage’s state (e.g., Expert mode) and also preferences that are really appearance or behavior preferences, but

that are thought to be uncommon, obscure, or primarily of interest to expert users. Three miscellaneous prefs are new for Leverage 3.1.

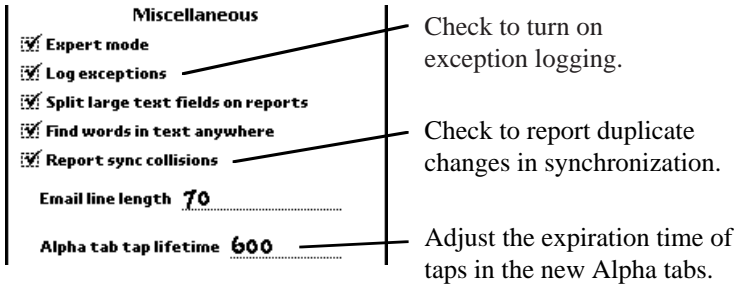


Figure 22— New miscellaneous prefs

It is no longer necessary to manually create the `_exceptions` database (or import the sample) to turn on exception logging. Simply check or uncheck the “Log exceptions” preference to enable or disable exception logging. Exception logging itself is primarily useful for tracking down problems with code, formulae, New scripts, and other user added NewtonScript.

Check “Report sync collisions” to receive notification when synchronizing that some modified Leverage records have been overwritten by modified records from the desktop database (i.e., the same record has been changed on both machines, hence the term “collision”). This isn’t as helpful as it might be, since you’re told after the fact that the records have been overwritten, but it can at least alert you to potential data loss and the possibility of procedural problems.

As noted above in the section on the new Alpha tabs, successive taps build a target key value, rather than simply changing to a new first letter (e.g., tapping “s” and then “t” takes you to the “st”s, not to the “s”s and then to the “t”s). Whether the

second tap is interpreted as an additional letter or a new key value depends on how much time has passed. The “Alpha tab tap lifetime” pref controls the allowable time between taps. It is in sixtieths of a second, so the default value of 600 means that a second tap within ten seconds of the first is appended to the previous letter.

Column Alignment

It is now possible to specify column alignment (left, right, or centered) in list view. The default behavior is to align text to the left and numbers to the right and to center booleans. If you find this causes information in one column to run into information in a neighboring column, you can change it. Tap the column header to open the column characteristics dialog.

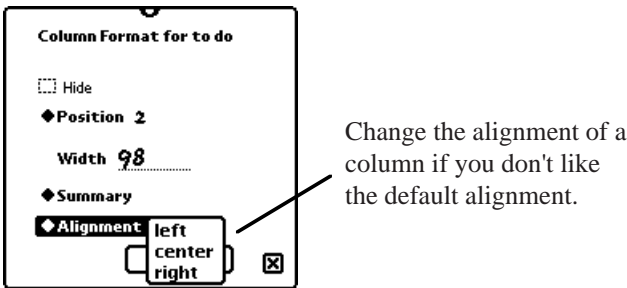


Figure 23— Setting column alignment

Column alignment is saved when a view is saved under a name.

Print Current Record Option

When printing using the Single Record format there is a new option to print only the current record. When you only want to print one record there's no longer any need to concoct a query

that selects only this record before printing.

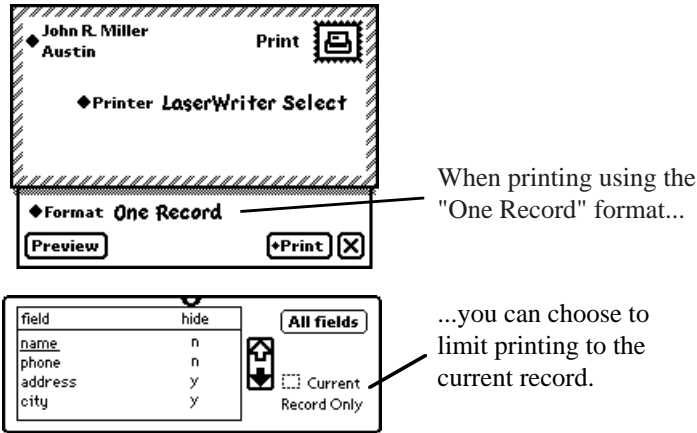



Figure 24— Print current record

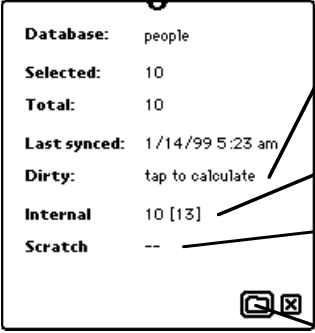
Enhanced “About this database” Slip

The info button (the button in the lower left of the Leverage screen with an “i” on it: ) includes a selection that presents information on the current database. Exactly what the selection says depends on the name of the current database. If you’re in a database named “Contacts”, the choice will be “About Contacts”.

For Leverage 3.1 this slip has been enhanced to present additional information and to give you greater control over database location.

The information displayed includes the number of records in the database and the number of records currently selected as well as the number of records on each currently active store (card or internal memory) and the number and location of the field definitions for the database. Also displayed is the date

and time of last synchronization (if ever) and the number of records that have been changed since the last synchronization. Referred to as “dirty” records, these are the records in Leverage that would be sent to the desktop database if the two were synchronized. Of course, immediately after successful synchronization, the number of dirty records will be zero.



The screenshot shows a database information slip for a database named 'people'. The slip contains the following information:

Database:	people
Selected:	10
Total:	10
Last synced:	1/14/99 5:23 am
Dirty:	tap to calculate
Internal	10 [13]
Scratch	--

At the bottom of the slip, there are two icons: a folder icon and a close (X) icon.

Annotations with arrows point to the following elements:

- Annotation 1: Points to the 'Dirty' field. Text: "Tap here to display the number of 'dirty' records."
- Annotation 2: Points to the 'Internal' field. Text: "For this database, 10 records and 13 field definitions are in internal memory..."
- Annotation 3: Points to the 'Scratch' field. Text: "...and no records or field definitions are on the memory card named 'Scratch'."
- Annotation 4: Points to the folder icon. Text: "Tap the filing button to control the location of the database."

Figure 25— New About this database slip

The information on record and field definition location appears at the bottom of the slip. For each store (internal memory or card) the number of database records on that store is shown, followed by the number of field definitions (displayed in brackets).

Generally you’ll want to keep all records and all field definitions in a given database on one store. Usually this will be the same store where Leverage itself is located, although it need not be. If the About this database slip indicates that the records or definitions are split across stores, it is a good idea to file the database on a single store. Even if a database is collected on a single store, you may want to move it to another store.

Whether you are consolidating a database or moving it, begin by tapping the File button in the lower right of the About this database slip. Leverage will present a filing slip with radio buttons representing the available stores. Choose the store where you wish the database to be and tap on the File button.

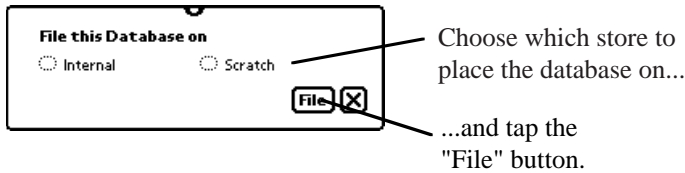


Figure 26— File slip

Leverage will move all records to the selected store (after checking that the store has enough room).

Time fields

Leverage 3.1 adds support for time fields, i.e., fields intended to contain a time of day. For many applications it is more appropriate to use a single date/time field (which Leverage has always supported) rather than separate date and time fields. In particular, it is much easier to query for records that precede a particular moment if the date and time fields are combined. If, say, we have an inspections database with a field or fields indicating when the inspection occurred (either a “day” date field and a “time” time field or just a “when” date/time field) and want all inspections that occurred after noon on 3 March 1998, it’s simple to query for “when” \geq 12:01 3 Mar 98. The corresponding query in the case of separate date and time fields would be something like (“day” $>$ 3 Mar 98 or (“day” = 3 Mar 98 and “time” \geq 12:01)). On the other hand, there are circumstances where separate date and time fields are appropriate. To extend our inspections database example, if we wanted to select those inspections occurring at any time on 3 Mar

1998, separate date and time fields would make the query easier. Another case that suggests separate date and time fields is a database originating in FileMaker, since FileMaker doesn't support a combined date/time field.

If you start with separate fields and decide to combine them later (or vice versa), it's a fairly simple matter to define an apply script that initializes the new date/time field from the separate date and time fields (or vice versa).

Assimilate

Ok. At last. That ominous looking selection on the Action button is documented. The purpose of the Assimilate choice is to make data from other applications available in Leverage — available for viewing, editing, reporting, querying, even importing and exporting and synchronizing. The main external data source that it is expected it will be applied to is the Newton's built-in Names app. Assimilate Names and you'll be able to perform queries and produce reports that were previously impossible. You'll even be able to keep your Names in sync with a FileMaker or Access database. Sort of.

Be warned, this is not as simple or perhaps even as useful as you and we would like. The data in Names (and, for that matter, in other Newton applications) is not organized as strictly as that in a Leverage database (or, for that matter, a FileMaker database). For example, Names entries can have an arbitrary number of phone numbers or addresses. This flexibility can be convenient, but it makes the data difficult to map into a Leverage database.

Begin the assimilation process by choosing Assimilate from the Action button.

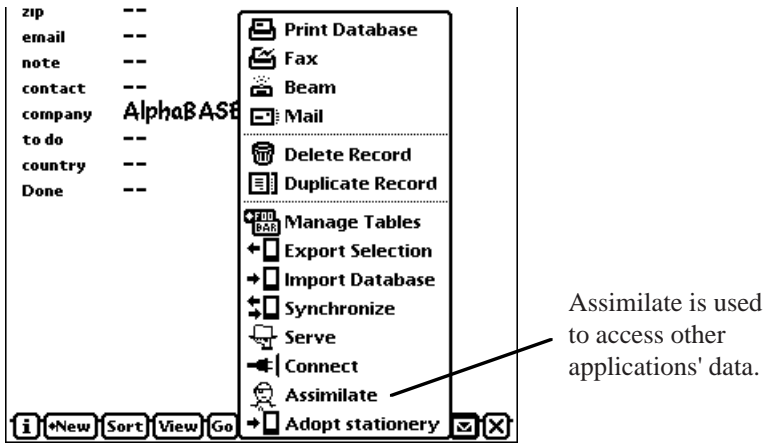


Figure 27— Assimilate

Leverage will open the Assimilate slip.

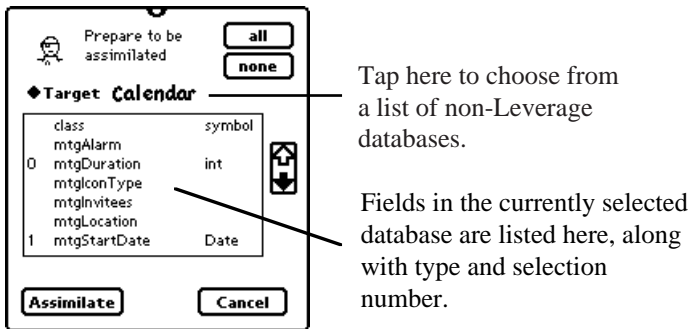


Figure 28— The Assimilate slip

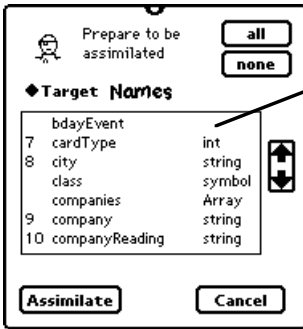
The pop-up list at the top lists potential data sources that could be assimilated. (They're called targets on the slip. They're both targets and sources, depending on how you look at them. If you'll forgive a somewhat grisly example, the target for an arrowing Prairie Falcon quickly becomes a source of protein. Come to think of it, it's even assimilated.) Most potential data

sources are likely to be even less appropriate for assimilation than Names. It is strongly advised that you do not simply try various sources (soups) to see what happens. You might not like what happens — particularly if you assimilate and modify. Simply assimilating and viewing is probably safe.

Choose a soup from the Assimilate slip’s pop-up list (for this extended example we’ll assume you picked “Names”) and Leverage will present a list of potential fields in the soup. Leverage constructs this list by reading through all the records in the selected soup. Leverage will keep you apprised of its progress as it reads the records, and you can cancel if it’s taking too long, but if you do cancel you run the risk of missing fields that are present only in later records.

The list of fields actually has three columns: the field name in the middle (actually, it’s not the field name — it’s the trailing part of the full field path, which isn’t quite the same thing), the field number on the left, and the field type on the right. The field number is present only for those fields that have been chosen for assimilation into Leverage, and is used to control the order in which the fields will be presented in Leverage’s default single record view. Leverage starts by assuming you’ll want to assimilate all the fields it identifies (i.e., the ones that Leverage can determine to be of a type it can handle), but you can control whether to assimilate a field by tapping on its field number. If you choose to assimilate a field that Leverage didn’t pick, you’ll need to tell Leverage what type to give it. Be warned that if you give a field the wrong type, it could cause trouble for the application that owns the soup.

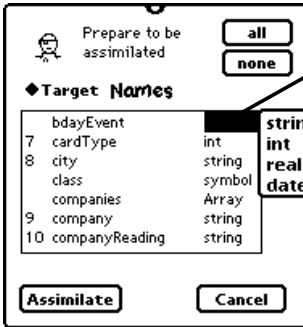
Generally you add or remove fields from the list being assimilated by tapping on the field name or field number (i.e., the first or second column). However, you can’t add a field that has no type or a type Leverage doesn’t recognize.



To include a field that Leverage hasn't divined the type of...

Figure 29— Assimilating a field of unknown type

To assimilate such a field, tap in the third column and choose the appropriate field type. To reiterate: if you choose an inappropriate type (and what is appropriate is completely determined by the application that owns the data) the owning app may have trouble with the data.



...you must first tap in the blank space where the type should be and select the correct type from the resulting pop-up.

Figure 30— Assigning a field type

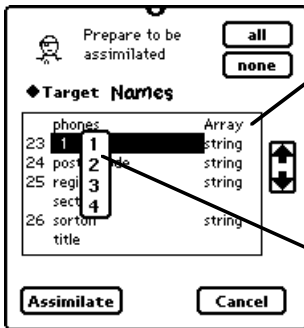
You can also tap in column three to change a field type Leverage has already assigned, but that is even more likely to cause trouble for the owning app (An exception would be changing a “string” field to a “text” field, but “text” isn’t on the list of field types in the assimilate slip anyway. If you want to make that change you can do it in the usual Create/Modify DB slip after

the data source has been assimilated.)

The field type is determined by actual inspection of the records, and includes the usual Leverage field types plus a few additional types used to show the structure of the records. Fields that have subfields are shown as type “frame”. An example (from the Names database — I’m going to stop pointing out that that’s our example) is the Name field, which has subfields “first” (i.e., first name), “last”, and a few others. Fields that occur multiple times (usually an arbitrary number of times, depending on how the data has been entered) are shown as type “array”. Examples of these are phone number fields and address fields.

When Leverage assimilates the fields, it constructs a field name from the full path to the information, not just the final part that it uses in the list of fields in the assimilate slip (since the final part isn’t unique and field names have to be unique). The field name is constructed by concatenating the parts of the full path. Thus, the first name is assimilated as namefirst and the first phone number is assimilated as phoneNumber0. You can change the name Leverage gives the field if you don’t like it (later, after the database has been assimilated, by choosing Modify from the database button, as you would with any database). This won’t affect the actual data nor will it affect the owning application.

If you choose to assimilate an array (e.g., the phone number fields), by default Leverage just creates field definitions for the first element of the array (which, as noted below in the Pitfalls section, could be any of the phone types, unless you’re consistent in how you enter them into Names). To assimilate more than just the first element, tap on the “1” that appears below the array line and specify another number to assimilate.



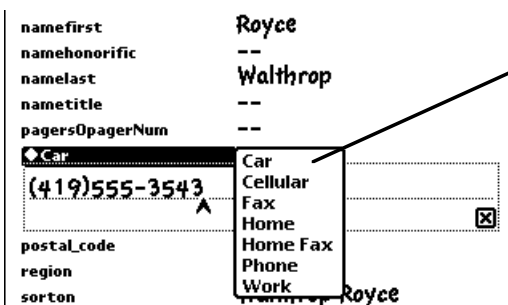
Phones is an "array" field since you can enter multiple phone numbers for a person in the built-in Names app.

To include more than just the first phone number, tap on the number and select more (sorry -- limit 4 per customer).

Figure 31— Array fields

Tagged fields

Phone numbers and pager numbers in Names include what is sometimes called out of band data. The numbers are tagged (home, work, fax, etc.) in a way that's similar to the subfield arrangement of names (i.e., names are divided into first, last, etc. — similarly, phone numbers can be thought of as divided into number and type). In single record view Leverage presents this information similarly to the Names app itself: the tag value is used as the label rather than the field name.

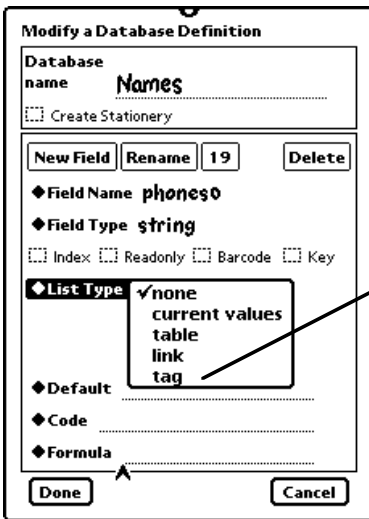


Tagged fields such as assimilated phone number fields present the tag value as the label, and choosing a different tag value from the pop-up list of tag values changes the label, not the field value.

Figure 32— A tagged field

The pop-up list for the such a “tagged” field is the list of tag values, and changing the tag changes the field label, but not the content of the field.

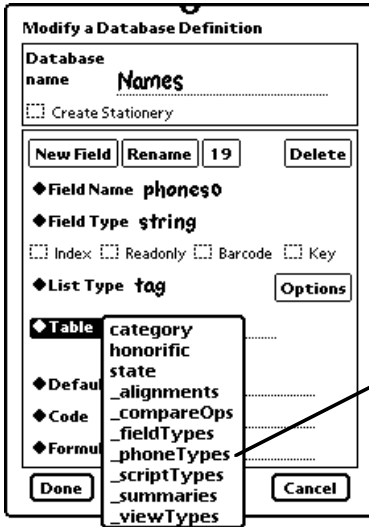
Although Leverage arguably should recognize the phone number fields in Names as tagged, it doesn't, so if you assimilate any phone number fields, you'll need to go into the Create/Modify DB slip after assimilating Names and set the List Type to “tag” for any phone number fields you included.



Leverage doesn't automatically recognize phone number fields as tagged, so you'll need to choose a List Type of "tag"...

Figure 33— Setting phone number fields to “tagged” pt.1

Then set the Table (which determines what tag values appear in the pop-up list) to “_phoneTypes” (Leverage creates this list of values for you).



...and then set the Table to "_phoneTypes".

Figure 34— Setting phone number fields to “tagged” pt. 2

Note that in list view, since there are potentially many records visible with many different tag values, the column heading is invariably the field name, never the tag value.

Note also that since you can enter phone numbers in any order, in one record the home number may be first and in the next record the work number may be first and the home number may be third. This makes it impossible, for example, to find all people with a home number (say) in a particular area code using a normal “by example” query. (It would be possible with a functional query, but you’d have to be conversant in NewtonScript to craft the query.)

This field attribute (i.e., tagged) is of course available to your own Leverage databases (under the List Type pop-up in the Create/Modify DB slip as shown in figure 33 above), and you’ll usually want to create your own table of tag values to

use. However, realize some of the limitations of the attribute. It is not possible to search on the tag value (again, except using functional queries). In fact, when defining a “by example” query there is no pop-up list associated with the field. It is difficult to express formulae that depend on the tag value. “Set...” from the bang button can’t be used to set the tag value. Tag values are not exported or imported, since normal databases and tab or comma-separated files have no place for this information (this is probably the most serious limitation). One last note about tagged fields and phone fields: phone fields in Names are treated specially by Leverage. The tag values displayed and the tag values stored are actually different (because this is how the Names app does it — don’t change the values in the `_phoneTypes` table). For any tagged fields you create yourself the two (the tags stored and those displayed) will be the same.

Limitations of Assimilate

You can’t assimilate notes type fields. Sorry.

Setting the order of the fields by choosing to assimilate them in the order you want is probably pretty useless unless you’re only assimilating a very few fields, the more so since Leverage starts with all assimilable fields selected. You’re probably better off just picking which fields you want and then adjusting the order in the usual manner (in Modify Database — of course, this method could be easier than it is, too).

Pitfalls when Assimilating

Aside from the aforementioned possibility of giving a field the wrong type and confusing the owning application, there are other things to look out for when assimilating foreign soups.

First, adding and deleting fields is not a problem (this is not a pitfall, of course, but it used to be, so it's explained in this section). If you add a field it's kept in a special Leverage place in each record so as not to affect the owning application. If you delete a field that you originally added (i.e., one that was not part of the application before it was assimilated — one that was added in Leverage) its definition and the data will be removed as usual for a Leverage database. If you remove a field that you didn't add, the definition will be removed (so it will no longer be accessible from within Leverage), but the data will be left intact (and still accessible within the owning application).

Second, don't change the type of fields that are already part of the foreign soup. This will almost certainly cause trouble for the owning application. One exception is that any string field could be changed to text (and back) since this doesn't change the way the data is stored. It's also possible to add tables or links to a field definition, but this isn't changing the field type. Leverage should of course tell you if you're doing something bad, and perhaps it will someday (This isn't really very likely).

Most other troubles are related to the previously mentioned tendency of foreign soups to be less strictly organized than Leverage databases. For example, the address fields in the Names database are actually of different types. If there is only one address it is simply the first line of the street address. If there are other addresses they are subsumed under an array of frames. Each element in the array is a full address, including a street address, city, region, postal code, etc. So the default name (i.e., the one Leverage assigns to the field when you assimilate it) for the first street address is simply "address". The default name for the second one is "address0address" (the address field of the first or zeroth element of the array of address fields). This can be confusing. Worse, it makes it

difficult to find records by address, since the sought address could be in either field. It's also a bad idea to add a, say, third address to a record that doesn't already have a first and second address (or a third phone number to a record that doesn't have a first and second). Your indication that the earlier addresses are not present would be that they are blank (in Names they'd simply not be there, but in Leverage the fields are present but empty). I'm not sure what would happen if you tried this (i.e., added an address or phone number out of turn), but the best it could be would be confusing, and I'd expect it to be worse.

Code, formulae, and NewScripts involving assimilated soups can be problematic since the code will need to use the actual path to refer to the fields, and this can be quite different from the name (in fact, it can be arbitrarily different, since you can change the field name without affecting the path).

Other Enhancements

Leverage no longer complains when its internal definitions are duplicated (usually caused by Leverage being installed first on a card, and then in internal memory with the card absent, after which the card is reinserted). Instead it quietly removes the duplicates, leaving the records that reside on the same store as Leverage itself.

Leverage now supports full editing of text fields in list view.

Leverage attempts to reduce database fragmentation by adding new records to the same store as existing records, rather than using the global preference from the Newton OS Card app to decide where to place records.

When making a network connection (for import or export, say), Leverage now remembers and lists previous connections in

much the manner of the Newton OS Dock application. Note that the FileMaker connectivity scripts present themselves as “LevLink”. When in doubt, you can still select “choose” from the “Connect to” pop-up to select from the Network chooser as before. If you do select a previous connection (handy if you’re always connecting to the same machine and program) the best sequence is probably to tell the desktop program to connect, and then tap “Connect” in Leverage (having already selected the correct item from the “Connect to” pop-up). It’s not critical, since Leverage will wait for the desktop program for a little, but the desktop program is probably more patient than Leverage.

Bug Fixes

Leverage now handles various index failures more robustly, and in general does a better job of handling data related failures (e.g., finding data in a record that is the wrong type for the database definition).

Diacritical marks are now translated correctly on import and export.

Limitations

Most of the new limitations are described in the appropriate section above.

Leverage still doesn’t do a perfect job of eliminating fragmentation. There are still circumstances which will force Leverage to resort to the global Card app preference to determine where to place a new record (e.g., if no records match the current query).