LinkDB

USER GUIDE

Table of Contents

LinkDB

LINKDB - WHAT IT IS USED FOR

LinkDB is a stand-alone application, separate from LinkUP!. LinkDB is used to process ASCII text files - such as those generated as reports by database application programs. LinkDB converts an ASCII text file generated by a database into a file that LinkUP! can use as a data source in the usual way.

LinkDB allows you to deal with report files generated in ASCII text format by database programs. It also allows you to 'clean up' the data they hold during the conversion process, perhaps removing leading or trailing spaces and other 'rogue' characters that appear in fields but that you do not wish to appear on your XPress pages.

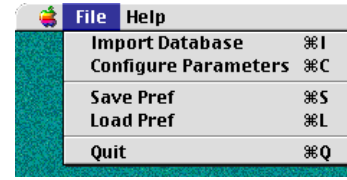
Remember, each time the source database changes and you are supplied with a new report file in ASCII text format, you will need to run LinkDB and generate a new LinkUP!-compatible file if you want to produce an update to the XPress document with the new data.

LINKDB - WHY YOU MIGHT NEED TO USE IT

LinkUP! can read database files directly from any ODBC-compliant database, but you may come across situations where you simply cannot gain access to an appropriate file. For instance, you may need to work with a report file that was generated by a remote mainframe computer as an ASCII text file - or you may find that you cannot gain access to an on-line database system and have to accept a report file generated for you by a somewhat reluctant IT department...

LINKDB - HOW YOU USE IT

LinkDB is simple to use - it has only one control menu: the FILE menu.



When you are working with a new ASCII text file for the first time in LinkDB, you need to Configure Parameters using the Import Text File Parameters Dialog and then Import Database in order to read the ASCII text file into LinkDB. The final step requires you to specify a name for the LinkUP!-compatible file to which LinkDB will write the data - bear in mind that this file will be of a similar size to the original ASCII text file. Make sure you have enough disk space available; an error will be generated if LinkDB doesn't find enough room to write the file it generates.

Once you have successfully imported, processed and saved out the contents of an ASCII text file in LinkDB, you may wish to save the Text File Import Parameters that you used to do the job as a LinkDB preferences file. LinkDB allows you to load and save preferences files using Load Pref and Save Pref so that you don't have to reconfigure the Text File Import Parameters by hand each time you need to process an ASCII text file from a particular database source.

ASCII TEXT FILES:

An abbreviation that has been on the lips of computer people for decades now is GIGO, and before your ask, Garbage In Garbage Out. Computers are still not capable of complete artificial intelligence, so on the whole, the data that users get out of a computer system is only as good as the data they put in.

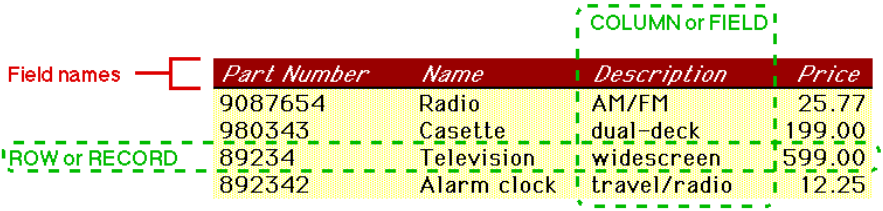
This philosophy applies to LinkUP!. In order to create a LinkDB file, there must be an accurate source of information for LinkDB to work with. This data will most likely be held in a database, but this isn t a requirement.

LinkDB works with what is known as ASCII data and therefore your data must be contained within an ASCII file. Don t worry though, this is simpler than it might sound. Virtually every type of database, spreadsheet and word processor will allow you to save your information as an ASCII text file, certainly all the mainstream software packages will.

Although ASCII is a universally accepted standard, there are many varieties depending on the application or computer system that has created the file. ASCII is also a very inefficient format to store database information in, as access to the data is extremely slow. LinkDB processes the ASCII file, standardising its contents and converting it to a LinkDB file, a far

more efficient format. This converted file is one that you can work with within your QuarkXPress document.

When a database program exports the contents of a database file or table to an ASCII text file it also needs to export control information alongside the data. Take a look at the following example database file or 'table':



ROW, OR RECORD

A horizontal row in the database table contains all the data that is associated with a single entity - a product, perhaps, in the case of a database file that will be used with LinkUP! to create a price list.

COLUMN OR FIELD

A vertical column in the database table contains data of a particular type or category - prices, perhaps, in the case of a database file that will be used with LinkUP! to create a price list.

FIELD NAMES

The first row in the database table may not contain data associated with a single entity; instead each cell in the row might be used to store a label that refers to the type or category of the data or information that is stored in the cells below. In effect, the cell at the top of each vertical column can be used to store a 'header' or label that can be used to identify that column. This label is called the Field Name.

In our illustration, PART NO, NAME, DESCRIPTION and PRICE are the four Field Names.

WHAT YOU MIGHT FIND IN ASCII TEXT FILES

For LinkDB to achieve a successful conversion from a report file generated in ASCII text format by a database application, that report has to be generated so that the start and end of each field can be determined. The end of every record also needs to be indicated and every record needs to be exported with the same number of fields. Generating a report that follows these rules should not be a difficult in any commercially-released database application.

If you need to generate an ASCII text file from a database, refer to this section and in particular to the Rules for Valid data before specifying the control characters that will be used to prefix, suffix and delimit fields and records.

Here's a run-down of the structures and control characters that you might need to configure LinkDB to deal with:

Field Prefix

This is a character or string of characters that precedes the data held in each field, but is not actually a part of the data in a field. In acts rather like an 'open bracket' to encapsulate the data that a field holds. Often it is the same character or string of characters as the Field Suffix.

A Field Prefix does not appear in all ASCII text files generated by databases. LinkDB discards the Field Prefix characters it encounters and does not store them with the file it generates from the ASCII text file you are processing.

When a Field Prefix is used, LinkDB will treat any characters it encounters after the Field Prefix as valid data belonging to the data in the field currently being processed rather than as control codes or delimiters - until it encounters a Field Suffix character, which flags the fact that all the data characters in the current data field have been processed.
An ASCII data file that uses Field Prefix and Field Suffix allows characters, such as a comma, to be used inside fields as data and outside fields as a control character.

Field Suffix

This is a character or string of characters that follows the data held in each field, but is not actually a part of the data in a field. In acts rather like an 'close bracket' to encapsulate the data that a field holds. Often it is the same character or string of characters as the Field Prefix.

A Field Suffix does not appear in all ASCII text files generated by databases. LinkDB discards the Field Suffix characters it encounters and does not store them with the file it generates from the ASCII text file you are processing.

Field Separator

This is a character or string of characters that is used to separate each field. Field Separators must appear in the ASCII text file you are seeking to process with LinkDB, because without them, LinkDB cannot reliably identify where one field ends and the next begins.

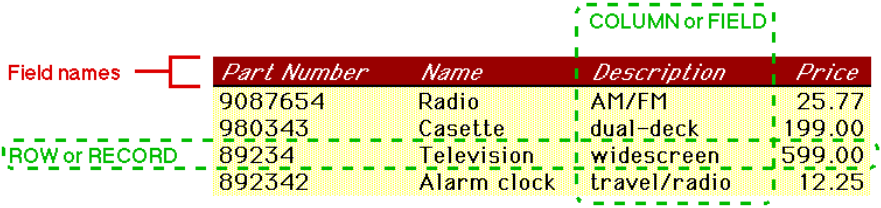
The Field Separator appears after each Field Suffix if Field Suffixes are present, or immediately after the data that forms the content of each field if Field Suffixes are not present - EXCEPT for the last Field Suffix or the last field in each record. Instead of a Field Separator after the last field in a record, there will be a Record Separator.

Record Separator

This is a character or string of characters that is used to separate each record. It appears after the Field Suffix that follows the last field in each record if Field Suffixes are present, or immediately after the data that forms the content of the last field in each record if Field Suffixes are not present

Mainframe style price breaks

Some mainframe database systems use a data structure under which records holding stock items contain one or more quantity fields, each immediately followed by a field containing the price for that quantity.
LinkDB needs to know the number of the column in which the first quantity/price pair is located so that it can unpack the data correctly when the ASCII text file is processed.



Here are a couple of examples of the content of ASCII text report files that could be generated from the data held in our example database table, as they are viewed when you import them to an XPress page with View Invisibles enabled:

In this version, there are no prefixes or suffixes to fields and a comma is used to separate fields. A Carriage Return + Line Feed is used to mark the end of a record.

```
Part Number,Name,Description,Price¶
9087654,Radio,AM/FM,25.77¶
980343,Casette,dual-deck,199.00¶
89234,Television,widescreen,599.00¶
892342,Alarm clock,travel/radio,12.25¶
```

This is how the data looks if it is exported from Excel in 'tab delimited' format. Fields are separated by the TAB character, and the end of a record is marked by TAB+Carriage Return+Line Feed

```
Part Number→Name→Description→ Price→ ¶
9087654→ Radio→AM/FM→ 25.77→¶
980343→ Casette→ dual-deck→ 199.00→ ¶
89234→Television→ widescreen→ 599.00→ ¶
892342→ Alarm clock→ travel/radio→ 12.25→¶
```

In this case, the fields all begin and end with a double quote and are separated by a comma - Carriage Return + Line Feed is used to mark the end of each record.

```
"Part Number","Name","Description","Price"¶
"9087654","Radio","AM/FM","25.77"¶
"980343","Casette","dual-deck","199.00"¶
"89234","Television","widescreen","599.00"¶
"892342","Alarm clock","travel/radio","12.25"¶
```

You will notice that the first records of all three of these exported data files contain PART NO, NAME, DESCRIPTION and PRICE, which are the field names from the database table. This means that the first record doesn't contain data; instead it holds field names and so the field structure has been embedded in the ASCII text file. More often than not, you will find that the field structure is not embedded in an exported database file that you need to process with LinkDB.

These example ASCII text files have been loaded as part of the LinkDB installation, and can be used in conjunction with the [LinkDB Tutorial] section of this manual if you would like to practice configuring and using LinkDB.

NB: XPress displays the control sequence CR+LF (^M^J) and the control sequence CR (^M) as

¶

This is because XPress ignores the LF control sequence when it encounters it after the CR control sequence. If you open an ASCII text file that contains CR+LF control sequences in XPress, when you save it out of XPress the LF control sequences will be stripped out.

Rules for Valid Data

The control character or string of characters used for the Record Separator must be unique from the character or characters used for the Field Separator and for the Prefix and Suffix if Prefix and Suffix are present. If Prefix and Suffix are not used, then the control character or string of characters used for the Record Separator not only have to be unique, but must not appear as part of the data anywhere in the ASCII file.

The control character or string of characters used for the Field Separator must be unique from the character or characters used for the Record Separator and for the Prefix and Suffix if Prefix and Suffix are present. If Prefix and Suffix are not used, then the control character or string of characters used for the Field Separator not only have to be unique, but must not appear as part of the data anywhere in the ASCII file.

The control character or string of characters used for the Prefix can be the same as the character or string of characters used for the Suffix or it can be different.

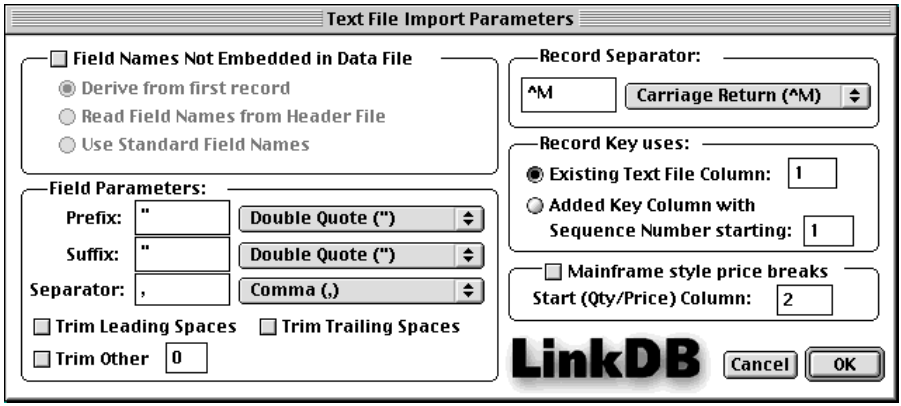
A control character or string of characters must be unique and mutually exclusive if it is to differentiate between Prefix/Suffix, Field Separator and Record Separator. So a control character or sequence of characters cannot be a contained in another sequence of characters that is being used as a control character.

A maximum of 255 fields is permissible.

There is no restriction on the length of fields.

Field names will be truncated to the first 20 characters

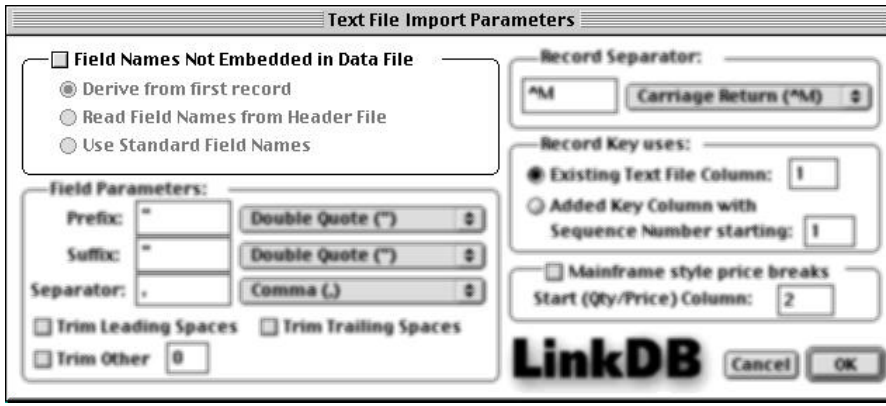
SETTING IMPORT PARAMETERS



The Text File Import Parameters dialog is accessed from the Configure Parameters menu option in LinkDB. It allows you to set up parameters within LinkDB that inform LinkDB of the structure of the data in the text file that you are about to process. From the Configure Parameters dialog you can also specify how Field Names will be derived for the file that LinkDB saves out from the data being processed, and perform some basic 'cleaning' of the data - removing leading or trailing spaces from the content of fields, for instance.

There are five main control areas within the Import Parameters dialog. We will now explore each one of these in turn.

FIELD STRUCTURE CONTROLS - NOT MANDATORY



The default setting for the field structure controls section of this dialog is to leave the box next to Field Names Not Embedded in Data File unchecked. If this box is not checked, then LinkDB will read the content of the fields in the first record held in the ASCII text file, use the content in its own internal structure as field names, and then discard the data it has just read from this first record in the ASCII text file. The second record in the ASCII text file then becomes the first real record in the database table file that LinkDB creates for you to use with LinkUP!.

Checking the Field Names Not Embedded in Data File box gives access to controls that affect how LinkDB looks for and assigns field names in the file that it generates from the ASCII text file being processed.

You only need to use these controls if the list of field names IS NOT held in the first record contained in the ASCII text file you are processing OR if the list of field names is held in the first record contained in the ASCII text file you are processing and you want that first record to be the first record in the database table that you are creating with LinkDB.

◉ Derive from first record

Selecting this option causes LinkDB to read the fields in the first record held in the ASCII text file, use them in its own internal structure as field names, and then write the record as the first real record in the database table file that it creates for you to use with LinkUP!.

◉ Read Field Names from Header File

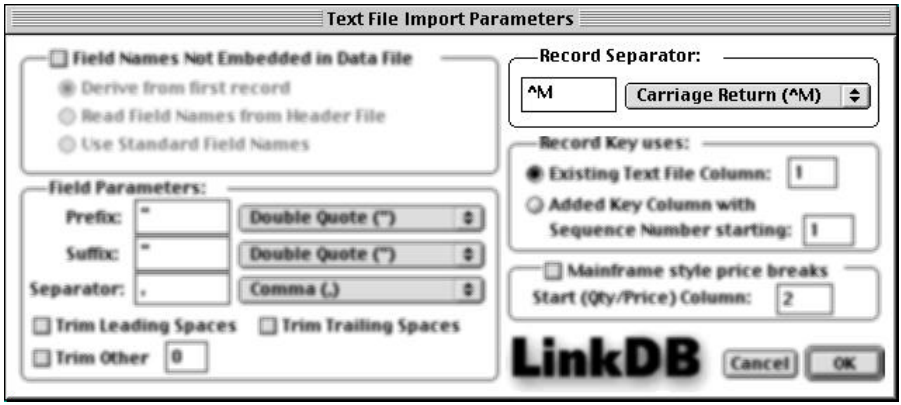
Selecting the Read Field Names from Header File option tells LinkDB that the field names are not embedded in the ASCII file as the first record. Checking the radio button next to this option means that during Import Database LinkDB will prompt you to select a second ASCII text file that contains the list of the field names that you wish it to use when it stores the data in its own internal structure. This file may have been supplied with the ASCII text file you are processing, or you may wish to create it yourself. If you need to create a Header File and don't know how to, jump to [Creating A Header File].

The structure of your Header File must mirror exactly the structure represented in the ASCII text file for which the Header File is to be used with.

◉ Use Standard Field Names

Selecting the Use Standard Field Names option tells LinkDB that the field names are not embedded in the ASCII file as the first record. Checking the radio button next to this option means that LinkDB will use default field names in its own internal structure, and you will be able to identify fields by the names FIELD1, FIELD2, FIELD3 and so on in sequence.

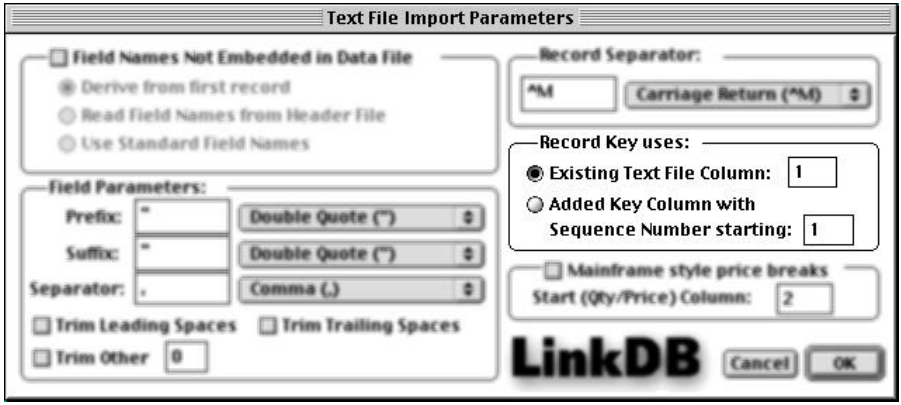
RECORD SEPARATOR CONTROL - CORRECT VALUE REQUIRED



You can either use the pop-up menu in this panel to select the character or character string that has been used in the ASCII text file to denote the end of each record in the data - or you can type the character or character string directly into the box.

For an explanation of the functions of the Record Separator jump to the [What you might find in ASCII report files] section of this manual.

KEY FIELD USES CONTROLS - ONE OPTION MUST BE SELECTED

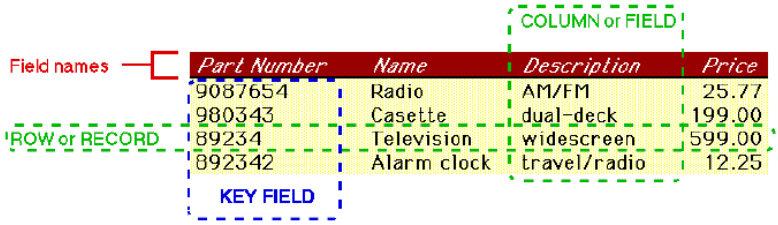


LinkUP! needs you to assign a field (or column) in every data source that you use as the field which acts as the Key field. This panel allows you to specify which of two possible ways the Key field will be assigned to the database table that LinkDB generates from the ASCII text file being processed. For an explanation of the functions of the Key field, jump to the [Key field] section of this manual.

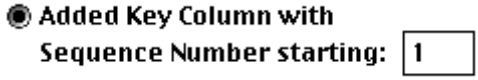
You can either select an existing field (which must contain unique data for each record in the file) as the Key field or you can force LinkDB to generate a new field in the database table file which contains sequential numbers which will be used as the Key field.

Selecting a field - or column - as the Key field

If you check the radio button next to Existing Text File Column, you need to key in the number of the field (counted from the left with the leftmost field being numbered '1') that you wish to use as the Key field. If we used our example database table and keyed in '1' next to the Field No option, we would configure LinkDB to generate a database table file where our Part Number field is used as the key field:



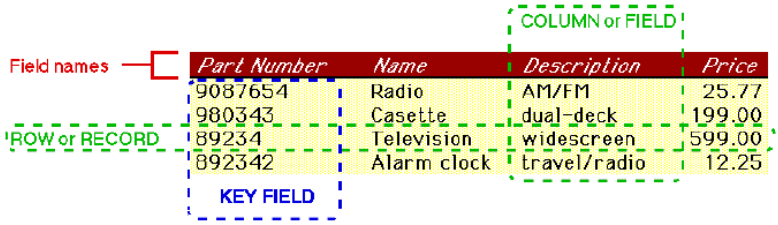
Generating a new Key Sequence as the Key field



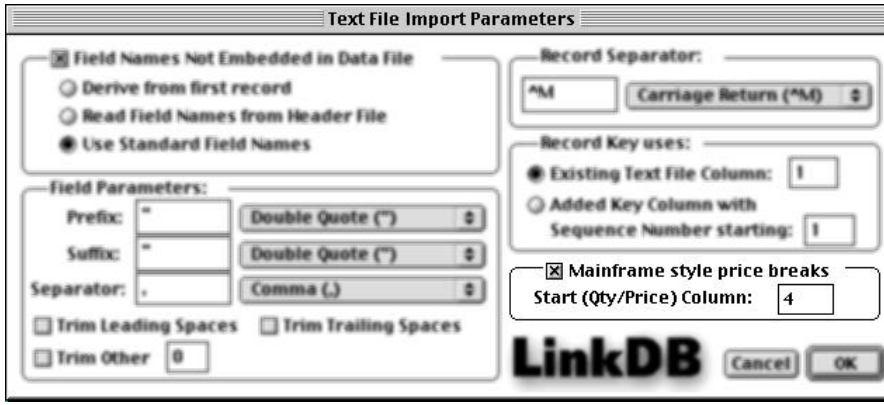
If you check the radio button next to the Added Key Column with Sequence Number starting option, you need to key in the number - an integer - that you wish to use to start the numeric sequence that will be written into a new field that LinkDB creates in the file that is writes from the content of the ASCII text file that you are processing. This new field will be used as the Key field.

The default is the integer '1'. You should use this default unless you are working with multiple files from a single data source and need to produce Key fields for several files and you want those Key fields to run in numeric sequence across those files. (The smarter move would probably be to amalgamate to data files into one file and then process that through LinkDB.)

If we used our example database table, selected Added Key Column with Sequence Number starting and left the default value '1' as it is, we would configure LinkDB to generate a database table file where a new field is located immediately to the left of the Part Number field. This new field would be assigned as the Key field:



MAINFRAME PRICE BREAKS CONTROL - OPTIONAL



Checking the Mainframe style price breaks box gives access to the controls that allow you to deal with a special instance where the ASCII text file being processed has been generated from a mainframe database and uses a special format for holding price breaks.

Some mainframe database systems use a data structure under which records holding stock items contain one or more quantity fields, each immediately followed by a field containing the unit price for that quantity. LinkDB needs to know the number of the column in which the first quantity/price pair is located so that it can unpack the data correctly when the ASCII text file is processed.

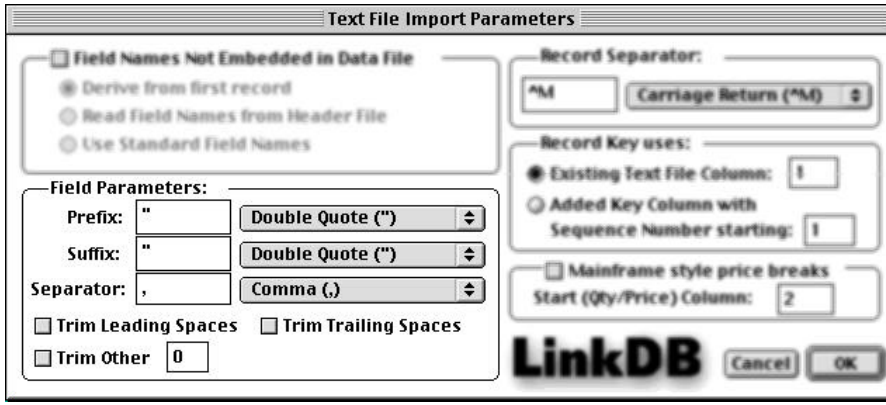
Part Number	Name	Description	1 to 5	Price	6 to 10	Price
9087654	Radio	AM/FM	25.77	25.77	25.77	20.99
980343	Casette	dual-deck	199.00	199.00	199.00	179.99
89234	Television	widescreen	599.00	599.00	599.00	589.99
892342	Alarm clock	travel/radio	12.25	12.25	12.25	11.99

This illustration represents a database table generated from data held in a mainframe data structure. In the data structure records that contain a stock item contain one or more quantity fields, each immediately followed by a field containing the price for that quantity.

Deriving the value for Start (Qty/Price) Column

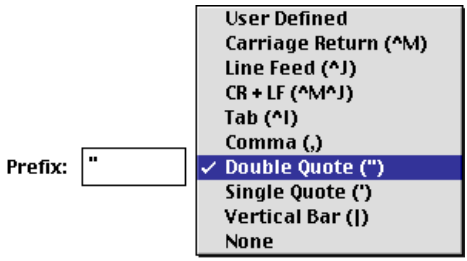
There are seven fields in the above example table and the leftmost field is always counted as '1'. Numbering from the left, the first price pair in our example starts in column 4, so '4' is the value that we would enter in the value box next to Start (Qty/Price) Column in the Mainframe Price Breaks section of the dialog.

FIELD PARAMETERS CONTROLS - SEPARATOR VALUE MANDATORY



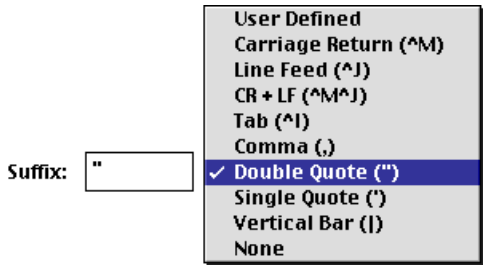
For an explanation of the functions of Field Prefix, Field Suffix and Field Separator jump to the [What you might find in ASCII report files] section of this manual.

Prefix - optional



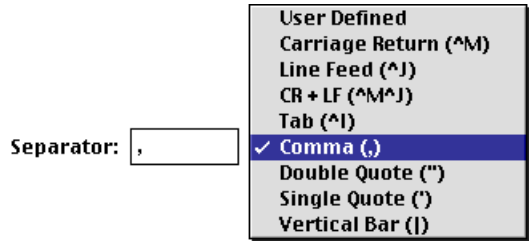
You can either use the pop-up menu in this panel to select the character or character string that has been used in the ASCII text file as a Record Prefix in the data - or you can type the character or character string directly into the box.

Suffix - optional



You can either use the pop-up menu in this panel to select the character or character string that has been used in the ASCII text file as a Record Suffix in the data - or you can type the character or character string directly into the box..

Separator - mandatory

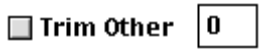


You can either use the pop-up menu in this panel to select the character or character string that has been used in the ASCII text file to denote the end of each field in the data - or you can type the character or character string directly into the box.

Trim Leading Spaces -optional / Trim Trailing Spaces -optional

Check these boxes if you would like LinkDB to strip out any spaces leading or trailing the data in the ASCII text file before it is written to the database table file that LinkDB generates for you.

Trim other - optional



Check this box if you would like LinkDB to strip out a single character or string of characters from the data in the ASCII text file as it is written to the database table file. (This could be used to strip out inappropriate currency symbols, for instance.)

NB; There is no restriction on the length of fields imported, but field names will be truncated to the first 20 characters. The maximum number of fields that can be imported is 255.

LOAD PREF/SAVE PREF

As was mentioned earlier, once you have configured LinkDB for your particular ASCII File, you can save this as a LinkDB Pref file. Next time you need to convert the same structure of ASCII file into a LinkDB file, simply load the correct Pref file and all the configuration is setup for you automatically.

To make things easier for you, with LinkDB come some Pref files for the most common database types. You will find Pref files already setup for Excel, 4-D, FileMaker Pro and more. So if you are working with any of these data sources, simply open the relevant Pref file and convert your data.

TROUBLESHOOTING LINKDB IMPORTS

In the Text File Import Parameters dialog take particular care when you specify the character or string of characters used as the Record Separator. If the ASCII text file you are processing uses a carriage return (CR) and a line feed (LF) to mark the end of a record, then be sure to enter both C and LF - the control string ^M^J in ASCII - in the Record Separator box.

If CR and LF is used in the ASCII text file as the Record Separator and you only enter CR - the control string ^M in ASCII - in the Record Separator box in LinkDB's Text File Import Parameters dialog, LinkDB will process the ASCII text file without reporting an error. When you come to use the file you generated with LinkDB problems will be caused in LinkUP! by the fact that the ASCII character ^J (LF) appear as the first character in the first field of every record in the database table you are using as a data source to update your XPress document.

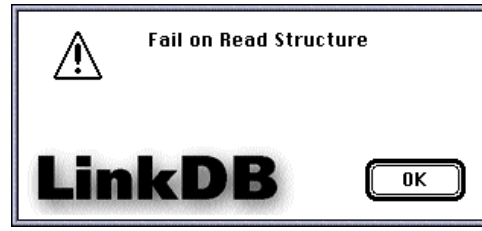
Nearly all of the problems you may encounter when importing an ASCII report file relate to the way the data is delimited in the file. The best thing to do is to take a look at the data and try to see what is going on. If at all possible, refer to the person who provided you with the ASCII report file and ask them to provide answers to the following questions about the file they supplied:

Is the field structure embedded in the file?
If not, what are the field names, and are they supplied in a separate 'header' file?
If so, how is it embedded - as the first record in the data, or in some other way?
What character or character string is used to separate fields?
What character or character string is used to mark the start of a field?
What character or character string is used to mark the end of a field?
What character or character string is used to mark the end of a record?

This should provide you with the information you need to set up the Import Parameters correctly and achieve a successful LinkDB data conversion.

LINKDB ERROR MESSAGES - WHAT THEY MEAN

Fail on Read Structure



LinkDB uses the information you supply in the Text File Import Parameters dialog in order to process the contents of your ASCII file.
This error message will appear if LinkDB is unable to use your parameters to derive a logical structure into which the data in your ASCII file can be imported.

The most likely cause of this error message is a problem with the Record Separator. It may be that your setting for the Record Separator in the Text File Import Parameters dialog does not match the character or character string actually used in the ASCII file to act as a Record Separator. ASCII text files generated on the Macintosh tend to use CR alone as a Record Separator; files generated on the PC usually use CR and LF. Double check your Record Separator setting and try running the import again...

If you are certain that there is not a problem with the Record Separator, and you receive this error message then it may be that the character or character string you specified in the Text File Import Parameters dialog as the Record Separator is the same as character or character string you specified in the Text File Import Parameters dialog as the Field Separator. Double check your Record Separator and Field separator settings and try running the import again...

If you are certain that neither the Record Separator nor the Field Separator settings have caused the problem, and you receive this error message then it may be that the character or character string you specified in the Text File Import Parameters dialog as the either the Field Prefix or as the Field Suffix is the same as character or character string you specified

for either the Record Separator or for the Field Separator. Double check your Record Separator and Field separator settings and try running the import again...

If there is still a problem, check that your settings for the Field Structure Controls are correct and that your Header File, if you are using one, exactly mirrors the structure of the data contained in the ASCII text file that you are seeking to process.

If the Fail on Read Structure error message still appears, then the content of your ASCII text file could well be the source of the problem. Refer to the [What you might find in ASCII Text files section of this manual, and in particular to the [Rules for Valid Data] section.

Error locating field suffix

Essentially this means that LinkDB has read in the character or string of characters that you assigned in the Text File Import Parameters dialog, so has identified a valid Field Prefix. Unfortunately, it has been unable to find a valid Field Suffix to match up with the Field Prefix.

When LinkDB encounters a Field Prefix, it treats any characters it encounters after that Field Prefix control character or string of control characters as valid data belonging to the data in the field currently being processed rather than as control codes or delimiters - until it encounters a Field Suffix character, which flags the fact that all the data characters in the current data field have been processed.

It may be that your setting for the Field Suffix in the Text File Import Parameters dialog does not match the character or character string actually used in the ASCII file to act as a Field Suffix. Double check your Field Suffix setting and try running the import again...

If you are certain that there is not a problem with the Prefix and Suffix settings and the Error Locating Field Suffix error message still appears, then the content of your ASCII text file

could well be the source of the problem. Refer to the [What you might find in ASCII Text files section of this manual, and in particular to the [Rules for Valid Data] section.

EXAMINING THE CONTENT OF THE ASCII TEXT FILE

If you still have problems, it is time to take a look at the content of the ASCII file you are trying to process.

Open a new XPress document and create a text box. Using CRTL-E, import the ASCII report file as text, or open the file in a word processor cut and paste the text into an empty XPress text box. Once you have the content of the ASCII text file in XPress, select View-Show Invisibles so that you can see any control characters that are embedded in the file.

By looking at this XPress document, you should be able to see whether the field structure is embedded as the first record in the file, and should be able to work out which ASCII character or combination of characters is used to mark the start and end of each field, which ASCII character or combination is used to indicate the end of each record, and whether or not an ASCII character or combination is used to separate records.

You will also see if there are spaces or other characters before or after the content in records.

NB: XPress displays the control sequence CR+LF (^M^J) and the control sequence CR (^M) as

This is because XPress ignores the LF control sequence when it encounters it after the CR control sequence. If you open an ASCII text file that contains CR+LF control sequences in XPress, when you save it out of XPress the LF control sequences will be stripped out.

Problems with the naming of Fields

If your examination of the content of the ASCII file in XPress reveals that field names are not present as the first record and you have been having difficulty processing the file with LinkDB, then you may need to supply your own header file containing field names, or request LinkDB to assign standard field names in the form $FIELD_1$, $FIELD_2$, $FIELD_3$ for you. For more details on these options jump to the section [Setting Import Parameters] and refer to the instructions on configuring the Field Structure Not Embedded in Data File controls in the Import Parameters dialog.

ADVANCED TROUBLESHOOTING: EDITING ASCII TEXT FILES...

If you are confronted with an ASCII text file that LinkDB cannot import and convert, then it may be possible to use the Edit-Find/Change facilities within XPress to amend the file so that LinkDB can use it.

This approach should only be taken if you really cannot source a new version of the ASCII text file that conforms to the needs of LinkDB.

Remember to save your edited file from XPress as ASCII text only rather than as an XPress document using File -Save as. Also bear in mind that XPress ignores the LF control sequence when it encounters it after the CR control sequence. If you open an ASCII text file that contains CR+LF control sequences in XPress, when you save it out of XPress the LF control sequences will be stripped out.

One problem that you may be able to fix by editing the ASCII text file is the problem caused if the record separator and field separator are the same character or sequence of characters - this means that LinkDB cannot differentiate between the end of a field or the end of a record. Searching for double occurrences of the character or sequence of characters and replacing them with the character plus another, unique, character string should create a file that LinkDB can work with.

For instance if the character string ^J (LF) has been used in the ASCII text file both as a record separator and as a field separator then LinkDB will not be able to import and process the file successfully. Providing the character string ^M (CR) is not used elsewhere in the file

as a field prefix, field suffix or in the content of any field, replacing all occurrences of ^J^J with ^J^M, saving the edited file as text-only should solve the problem.

The character sequence ^J^M will appear at the end of all records in the ASCII text file and ^J will now appear between records as the Record Separator. By setting up these parameters in the Text File Import Parameters dialog it should be possible to import and process the data with LinkDB.

CREATING A HEADER FILE

If your ASCII text file does not hold the field names as the first record and you know the names of the fields you can create a suitable text file that LinkDB can use during the import to assign field names to the file that it generates.

Open a new document in XPress. You need to know exactly how the fields and records are delimited in the ASCII text file you want to process with LinkDB before proceeding. Then it's a simple matter of opening a text box in XPress before typing in the names of the fields that you wish to use with the data. You need to type the field names in the correct order. If the data file uses a single character or a character string as a prefix or suffix to every field, then you must too. The character or string of characters used to separate records in the data must appear between the records you are creating here... and you must use the same character or string to mark the end of the single record that you are creating.

Bear in mind that LinkDB only uses the first 20 characters of data used as a Field Name.

QUICK TIP

To make sure that you replicate the structure of your data accurately, you can open the ASCII text file for which you are creating the Header File, copy and paste the first record into a new file and then edit the data in the fields, changing the data into meaningful field names. All the control characters will be left untouched and in the right place if you are careful...

THE LINKDB DATA COVERSHEET

This form is supplied as an XPress document on your LinkUP disks and also includes the 'Rules for Valid Data'.

If you are relying on someone else, perhaps someone in your company's IT department, to generate the ASCII file you will need to process with LinkDB, then you may find it helpful to send the Cover Sheet and Rules documents to that person, possibly along with the sample ASCII filed from the LinkDB tutorial so that they generate the ASCII file in the most appropriate format.

If you can encourage them to complete the Cover Sheet and return it with the ASCII data file, then you should have all the information you need in order to set up the parameters in LinkDB's Text File Import Parameters dialog.

We have included a sample Cover Sheet with the LinkDB program files on your installation disk.

The following page is an example of a possible Cover Sheet:

Question 1 - field names

Please state how many fields there are in the database file that has been exported: ....

Is the field structure embedded in the ASCII export file - that is, will the first record contain the field names?

If Yes, please tick here [ ] and go to Question 2

If No, Please list the field name on this sheet in the order in which the fields appear in the data. Ideally, we would like the field name information to appear in the ASCII file as the first record, or in a separate Header File. If you are supplying a Header File, please provide its name here:.....

FIELD NAMES.....

Question 2 - price break information

Does the database from which this ASCII file is generated use quantity/price pairs in order to store quantity and price break information? If it does and if this ASCII file contains records that contains mainframe-style price break pairs, please state the column number in which the first record of the first pair appears:.....

Question 3 - record separator

Please indicate the ASCII character or ASCII character string that is used in the ASCII export file as a record separator:....

Question 4 - field separator

Please indicate the ASCII character or ASCII character string that is used in the ASCII export file as a field separator:....

Question 5 - field prefix and suffix

Is an ASCII character or ASCII character string used in the ASCII export file as a field prefix?

If yes, please indicate that character or string:....

If no, please tick here [ ]

Is an ASCII character or ASCII character string used in the ASCII export file as a field suffix?

If yes, please indicate that character or string:....

If no, please tick here [ ]

PLEASE REFER TO THE RULES FOR VALID DATA sheet for guidance on the data structure and use of control code characters that will render the ASCII file useable.

Thank you.