



## 2L FIELD DEVELOPER v 11.X

### List of available commands in 2L

The commands are ordered by type. Each command is listed with the required command-parameters and a short description of the command is given.

An asterisk (\*) in front of the description means that the command is not implemented in the integrated test environment. You can not test this command during design.

(F) specified at the end of the description indicates, the command can only be used in form-mode.

(S) at the end of the description indicates, the command can only be used in spreadsheet-mode.

(F+S) at the end of the description indicates, the command can be used both in form- and spreadsheet-mode.

<b>Command type</b>	<b>Page</b>
File commands	2
Field commands	3
Search commands	5
Test-condition commands	6
Screenpage commands	7
Form and Sheet commands	8
(Error)Message commands	9
Information commands	10
Logfile commands	11
Button commands	12
Record commands	13
List commands	15
Commandfile commands	17
Filterfile commands	18
Timer commands	19
Other commands	20

## ***File commands***

If a filename specified in one of these file commands does not include a path, the actual project folder will be assumed.

If a filename includes a space, specify the filename between text quotes

APPEND <filename> <value>

    Add <value> to <filename> (F+S)

    If <filename> does not exist, it will be created

COPYFILE <oldfilename> <newfilename>

    Copy file from <oldfilename> to <newfilename> (F+S)

DELETEFILE <filename>

    Delete <filename> (without confirmation!!) (F+S)

DELETEFILE 0 <filename>

    Delete <filename> if it exists.

    Don't fail if <filename> does not exist (F+S)

IFFILE <filename>

    Test if <filename> exists (F+S)

SAVE <filename> <value>

    Save <value> to <filename> (F+S)

    If <filename> exists it will be overwritten

## **Field commands**

### **EDIT ALL**

Make all display data fields editable (F+S)

### **EDIT <fieldname>**

Make <fieldname> editable (F+S)

### **SETFOCUS <fieldname>**

Set the input focus (active input field) to <fieldname> (F+S)

### **NEXTFIELD**

Set the input focus to the next input field (F+S)

Input fields are N, A, M and S fields

### **SET <value> <fieldname>**

Fill <fieldname> with <value> (F+S)

Note, do NOT use any spaces in the value (or use SETVAL command !!!)

### **SETVAL <fieldname> <valuestring>**

Fill <fieldname> with <valuestring> (F+S)

<valuestring> can also be specified as a formula, e.g., %N2%+5 (only valid for numeric input fields)

<valuestring> may contain spaces.

### **SETFIELD <fieldname>/<fieldindex> <valuestring>**

Fill <fieldname> (or indexed field, 0=first input-field etc.) with <valuestring> (F+S)

### **SETACTFIELD <valuestring>**

Fill the active input field with <valuestr> (F+S)

### **DRIVER <drivename> <filename> <fieldname>**

\* Execute driver, get driver value from <filename> into <fieldname> (F+S)

### **RUNDRIVER <drivename> <filename> <fieldname>**

\* Load/Execute resident driver, get driver value from <filename> into <fieldname> (F+S)

### **LOAD <filename> <fieldname>**

Load contents (first line) of <filename> into <fieldname> (F+S)

### **LOADFROMREG <variable-name> <fieldname>**

Fill <fieldname> with the value from the registry-variable <variable-name> (F+S)

### **SAVETOREG <Variable-name> <value>**

Save <value> in the registry under <variable-name> (F+S)

If <Variable-name> does not exist in the registry, it will be created

### **DISPWIDTH <fieldname> <width>**

Change the displaywidth of <fieldname> to the <width> you specified (in pixels) (F+S)

### **AUTOCALC <mode>**

Sets AUTOCALC to <mode> (mode= ON or OFF).

If autocalc is set to OFF the fields will just before saving the actual record, or after a RECALC command, this saves processor time, for time-critical processes. (F+S)

Default AUTOCALC setting is ON

### **RECALC**



Recalculate immediately (F+S)

RECORDCOUNT <dataset> (<fldname>)

Return the number of records for the specified <dataset> in the variable \_COUNT (and the field <fieldname>) (F+S)

SUM <fieldname> <externalfieldname> <extdataset>

Sums the values of the <externalfieldname> column from <extdataset> and fills <fieldname> with this sum (F+S)

SETMIDSTR <fieldname> <startposition> <length> <valstr>

Fill <fieldname> with substring from <valstr> starting at <startposition> with <length> characters (F+S)

SETSUBSTR <fieldname> <index> <valstr>

Fill <fieldname> with the <index> substring from <valstr>, using space as substring separator (F+S)

SETSUBVAL <fieldname> <index> <sep> <valstr>

Fill <fieldname> with the <index> substring from <valstr>, with <sep> as separator <sep> can be a single character or a substring like \_X\_ (F+S)

SETRANDOMVAL <fieldname> (<minval>) <maxval>

Fill <fieldname> with a random value between 0 (or <minval>) and <maxval> (F+S)

INCVAL <fieldname>

Increment the numeric part of the actual value of <Fldname> (F+S)

DECVAL <fieldname>

Decrement the numeric part of the actual value of <Fldname> (F+S)

INPUT <Fldname> <Prompt>

Ask the user to enter text. Fill <fldname> with the user input (F+S)

SETINPUTTYPE <Fieldname> <Inputtype>

Change inputtype of <Fieldname> at runtime (F+S)

SETINPUTLENGTH <Fieldname> <Inputlength(.decimals)>

Change inputlength of <Fieldname> at runtime (F+S)

SETMINVAL <Fieldname> <Value>

Set minimum value check for <Fieldname> to <Value> (F+S)

SETMAXVAL <Fieldname> <value>

Set maximum value check for <Fieldname> to <Value> (F+S)

CLEARMINMAX <Fieldname>

Remove min/max values for <Fieldname> (F+S)

SETSPECIALS <Fieldname> <Specials>

Add <Specials> checks for <Fieldname> (F+S)

CLEAR SPECIALS <Fieldname>

Remove all special checks for <Fieldname> (F+S)

## ***Search commands***

**GETINDEX** <searchstring>

Search record in present dataset starting with <searchstring> (F+S)  
If found make this record the active record.

**GETNEXTINDEX** <searchstring>

Find next record in present dataset starting with <searchstring> (F+S)  
If found make this record the active record.

**FINDVALUE** <fieldname> <valuestring>

Search record in present dataset with value <valuestring> in field <fieldname>.  
If found make this record the active record (F+S)

**FINDVALUE -1** <valuestring>

Search record in present dataset with value <valuestring> in any field.  
If found make this record the active record (F+S)

**FINDNEXTVALUE** <fldname> <valstr>

Find next record in present dataset with value <valstr> in field <fldname>.  
If found make this record the active record. (F+S)

**SEARCH**

Start the search-procedure (as if the user selected the Search option from the Record-menu)  
(F+S)

## **Test-condition commands**

IF <condition>

Test if <condition> is correct.

<Condition> is an equation like %N1%>12 or %N3%>=3.5 or %RECID%=%NEWID% (F+S)

IF <condition> THEN <thencommand>

If <condition> is correct, execute/fail <thencommand>.

If <condition> not correct, continue with next command (ELSE REM OK)

(F+S)

IF <condition> ELSE <elsecommand>

If <condition> is correct, continue with next command (THEN REM OK)

If <condition> not correct, execute/fail <elsecommand> (F+S)

IF <condition> THEN <thencommand> ELSE <elsecommand>

If <condition> is correct, execute/fail <thencommand>

If <condition> not correct, execute/fail <elsecommand> (F+S).

IFVAL<value>

Test if value of active field is equal to <value> (F+S)

EMPTY <value>

Test if <value> (or %<fieldname>%) is empty (F+S)

EMPTYVAL <value>

Test if <value> (or %<fieldname>%) is empty (F+S)

EMPTYFIELD <fieldname>

Test if contents of <fieldname> is empty (F+S)

EMPTYLIST <fieldname>

Test if the list attached to <fieldname> is empty (F+S)

NOTEMPTY <value>

Test if the <value> is NOT empty (F+S)

NOTEMPTYVAL <value>

Test if the <value> is NOT empty (F+S)

NOTEMPTYFIELD <fieldname>

Test if contents of <fieldname> is NOT empty (F+S)

NOTEMPTYLIST <fieldname>

Test if the list attached to <fieldname> is NOT empty (F+S)

INLIST <listfield> <valstr>

Check if <valstr> is in list of <listfield> (F+S)

IFFILE <filename>

Test if file <filename> exists (F+S)

IFFIELD <fieldname>

ISFIELD <fieldname>

Test if <fieldname> exists in the present dataset(F+S)

ZERO <fieldname>

Test if actual value of <fieldname> is 0 (F+S)



NOTZERO <fieldname>  
Test if actual value of <fieldname> not 0 (F+S)

## **Screenpage commands**

- P  
Go to previous page (F)
- +P  
Go to next page (F)
- P <pagenumber>  
Go to page <pagenumber> (F)
- SHOW  
Redraw actual page immediately (F+S)
- SHOW <element>  
Show the (previously) hidden <element> (F)  
<element> can refer to a field or button.
- SHOWPAGE  
Show/refresh actual page (F)
- SHOWPAGE <pagenumber>  
Show/refresh <pagenumber> (F)
- HIDE <element>  
Hide the (previously) visible <element> (F)  
<element> can refer to a field or button.
- FULLSCREEN  
Expand form/sheet window to fullscreen (XP only, F+S)

## **Form and Sheet commands**

EDITMODE <mode>

Set edit-mode for present dataset to TRUE (to enable editing) or FALSE (to disable editing for all fields) (F+S)

FORM <formname> (<dataset>) (<searchstring>)

\*Swap to other form/data set, start with record found via <searchstring> (F)

-FORM <formname> (<dataset>) (<searchstring>)

\*Swap to other form/data set, without saving data (start with record found via <searchstring>) (F)

+FORM <formname> (<dataset>) (<searchstring>)

\*Swap to other form/data set with saving data (start with record found via <searchstring>) (F)

RUNFORM <formname> (<dataset>) (<searchstring>)

\*Run formmode as new task on top of the present form (swapping is faster, but requires additional memory to run) (F)

SHEET <formname> (<dataset>) (<searchstring>)

\*Swap to other form/data set in spreadsheet mode (start with record found via <searchstring>) (F)

-SHEET <formname> (<dataset>) (<searchstring>)

\*Swap to other form/data set in spreadsheet mode, without saving data (start with record found via <searchstring>) (F)

+SHEET <formname> (<dataset>) (<searchstr>)

\*Swap to other form/data set in spreadsheet mode, with saving data (start with record found via <searchstring>) (F)

RUNSHEET <formname> (<dataset>) (<searchstring>)

\*Run spreadsheet as new task on top of the present form (swapping is faster, but requires additional memory to run) (F)

END

Stop data-entry and go back to project-menu or start-form (F+S)  
Your last changes will be saved automatically.

CANCEL

Stop data-entry without saving data, and go back to project-menu or start-form (F+S)  
When you've made any unsaved changes, the program will ask you for confirmation to stop.

RETCOL

\*Specify return column in spreadsheet mode (S)

ADDCOL <Colnr>

\*Add column <Colnr> to spreadsheet and show it as D-field, the column does not have to be specified in the form (S)

ADDCOLS

\*Add all non-specified columns as visible D-fields to the spreadsheet (S)

## ***(Error)Message commands***

**ERROR <errortext>**

Show this error message (e.g. as cancel function for a button) (F+S)

**ERRORFILE <filename>**

Show content of <filename> as error message (F+S)

**MESSAGE <messagetext>**

Display <messagetext> in the message-line (at the top of the screen) (F+S)

**ALERT <alerttext>**

Pop-up an ALERT-box displaying the <alerttext> (F+S)

**CONFIRM <confirmtext>**

Ask the user to confirm <confirmtext>, continue if the user pressed Yes, fail if the user pressed No (F+S)

**BEEP <n>**

Alerts the user. Is often used with the MESSAGE command. (F+S)

<n> = 1 : Alert-beep; <n>=2 : Critical beep; <n>= 3 : Exclamation beep; other values : Asterisk beep

## ***Information commands***

INFO

Show standard info about actual form and record (like show info from the menu) (F+S)

INFO NEW

Clear the old info list (F+S)

INFO ADD <text>

Add <text> to the info list (F+S)

INFO ADDFILE <filename>

Add the contents of file <filename> to the info list (F+S)

INFO ADDFILE <filename> <starttext> (<endtext>)

Add parts of the contents of file <filename> to the info list. Start adding when <starttext> is encountered, and stop adding when <endtext> is found (F+S)

INFO SHOW

Show the info list (F+S)

INFO HIDE

Hide the info list (F+S)



## ***Logfile commands***

LOG <logfile> <value>

Add <value> (as new line) to <logfile> (F+S)

LOGFILE <textfile> <logfile>

Add the contents of <textfile> to <logfile> (F+S)

## ***Button commands***

### **KBCHAR**

This button acts as a keyboard key, using the first character of the description or name of the button (F+S)

### **KBSTR**

The full description (or name) of the button will be used as value in the active field (F+S)

## **Record commands**

### **FIRST**

Go to first record in data set (F+S)

### **PREV**

Go to previous record in data set (F+S)

**NEXT** Go to next record in data set (F+S)

### **LAST**

Go to last record in data set (F+S)

### **GOTOREC (+/-)<recno>**

Go to the specified record in dataset (if <recno> is specified with + or - , the jump to the new record is relative to the present record) (F+S)

### **SAVERECNO (<filename>)**

Save actual record number to RECNO.INI or <filename> (if specified) (F+S)

### **NEWREC**

Start data entry on a new record (F+S)

### **SAVEREC**

Save present record to memory (F+S)

### **ADDREC**

Copy present record as new record to data set (F+S)

### **DELREC**

Delete present record (after confirmation) (F+S)

### **DELLASTREC <Dataset>**

Delete last record form <Dataset> (without confirmation) (F+S)

### **DELONEREC <Dataset> <StartStr>**

Delete record(s) from <dataset> starting with <StartStr>

### **GETREC <dataset> <searchcondition>**

Get data from external <dataset>, where record matches <searchcondition>. A search condition is specified as <fieldname><condition><value> (F+S)

### **GETREC 0 <dataset> <searchcondition>**

Test external <dataset> if a record matches <searchcondition> (but do NOT fill fields from external dataset). (F+S)

### **GETSEARCHREC <fnr> <datafile> <searchstring>**

Search <datafile> for first record that matches <searchstring> in fieldnr <fnr> (fnr starts from 0) if found return this record in the SEARCHREC variable (F+S)  
<searchstring> can be a literal string, a variable (between%%)  
or a search condition on <colidx> (f.e. >12, >=88) or a "reverse condition" (12<=)

### **GETNEXTSEARCHREC <colidx> <dataset> <searchstring>**

Search external <dataset> for next record that matches <searchstring> in column <colidx> (F+S)

### **GETSREC <fnr> <fieldname>**

Fill field <fieldname> with substring <fnr> (fnr starts from 0) from SEARCHREC (F+S)



GETSVAL <fnr> <fieldname>

Fill field <fieldname> with substring <fnr> (fnr starts from 0) from SEARCHREC (F+S)

READSEARCHREC

Reread the actual searchrecord and fill the SEARCHREC variable (F+S)

DELSEARCHREC <n>

Remove the first <n> substrings from the SEARCHREC variable (F+S)

SAVEDATASET

\*Save data set to file (F+S)

CHANGED <mode>

Set changed mode of present record to True: (<mode>: YES or TRUE) or False (<mode>: 0 or FALSE) (F+S)

The command CHANGED FALSE followed by CANCEL, will not ask to confirm stop data-collection, even when you've made some changes in the present record.

GETINDEX <searchstring>

Search record in present data set starting with <searchstring> (F+S)

GETNEXTINDEX <searchstring>

Find next record in present dataset starting with <searchstring> (F+S)

GETINDEXREC <searchstring>

Search record in present data set starting with <searchstring> and fill SEARCHREC variable with the record found (F+S)

SORT <sortcolumns> (<dataset>)

Sort the <dataset> on the <sortcolumns> (comma-separated list). If dataset is not specified, the active dataset will be sorted (and saved to disk) (F+S)

ORDER <level> <dataset>

Reorder <dataset> on <level>, level 0 = first column only, 1 = first 2 columns etc.. (F+S)

JOIN <Dataset1> <Dataset2>

Update <Dataset1> with records from <Dataset2>. "Empty" records (index column is empty) from dataset1 will be removed. Matching records (same ID) will be overwritten with values from dataset2. New records from dataset2 will be added to Dataset1. (F+S)

REPLACE <Recid> <Fldname> <Value>

Replace content of <Fldname> with <Value> for those records starting with <Recid> (F+S)

REPLACE ALL <Fldname> <Value>

Replace content of <Fldname> with <Value> for all (filtered) records (F+S)

RECORDCOUNT <dataset> (<fldname>)

Return the number of records for the specified <dataset> in the variable \_COUNT (and the field <fieldname>) (F+S)

## List commands

- LIST <textfile>  
Display <textfile> as on-screen list (F+S)
- LIST HIDE  
Hide the on-screen list (F+S)
- LOADLIST <filename> <fieldname>  
Load the contents <filename> into <fieldnamelist> (F+S)
- LOADCODEDLIST <Filename> <Fieldname>  
Load the contents of <Filename> as coded list (linenr#value) into <Fieldname> list (F+S)
- ADDTOLIST <fieldname> <value>  
Add <value> to the list attached to <fieldname> (F+S)  
<fieldname> must be an A-field with list, N-field with list, Selection list or stack field (S-fields).
- ADDTOLISTML <Fieldname> <Value>  
Add <Value> to the Selection list of <Fieldname>, use multiple lines if necessary (F)
- CLEARLIST <fieldname>  
Clears the list attached to <fieldname> (F+S)
- DELFROMLIST <fieldname> <mode>  
Delete the selected line (or last line) from the <fieldname> list/stack, if <mode>= 0 : do not ask for confirmation, if <mode>=1 : ask for confirmation, if <mode>= 99: delete all items from the list (with confirmation) (F+S)
- SAVELIST <fieldname> <filename>  
Save the items in the <fieldname> list/stack as <filename> (F+S)
- SAVEDATLIST <Fieldname> <Filename> <Headerstr>  
Save the items in the <Fieldname>list to <Filename>, use <Headerstr> as list-header (F+S)
- NOTEEMPTYLIST <fieldname>  
Test if the list attached to <fieldname> is NOT empty (F+S)
- EMPTYLIST <fieldname>  
Test if the list attached to <fieldname> is empty (F+S)
- GETLISTSTRING <index> <listfield>  
Get <index> line from list attached to <listfield> into SEARCHREC. Fail if index is not appropriate. (F+S)
- GETSELSTRING <listfield>  
Get the selected item from the list attached to <listfield> and store in SEARCHREC-record. Fail if no line was selected (F+S)
- INLIST <listfield> <valstr>  
Check if <valstr> is in list of <listfield> (F+S)
- FINDLISTSTRING <fieldname> <searchstring>  
Test if the list attached to <fieldname> contains a line starting with <searchstring> (F+S)
- LISTINDEX <fieldname> <resultfield> <value>



Test if the list attached to <fieldname> contains the line <value>. If so, fill <resultfield> with the line number, starting from 0 (F+S)

FIELDLIST <Dataset> <Listname>

Create list file <Listname> containing all the field names from the header of <Dataset> (F+S)

VALUELIST <Colnr> <Dataset> <Listname>

Create list file <Listname> containing unique values from column <Colnr> of <Dataset> (F+S)

MAKEFILELIST <Listname> <Filespecs>

Create <Listname> containing the file names matching <Filespecs> (F+S)

## **Commandfile commands**

**CMD <filename>**

Execute multiple commands from command-file (F+S)

Note: the next command-file commands are only valid in command files !!

**DEBUG ON**

Set debug-mode ON or OFF (CMD) (F+S)

**DEBUG OFF**

Set debug-mode OFF (CMD) (F+S)

**DEBUG FILE**

Set debug-mode to file (CMD) (F+S)

All debugging messages from subsequent commands will be logged to 2ldebug.txt log file.

No user action is needed to continue.

Note for 2lfieldxp this file is called 2ldebugxp.txt, for the integrated test-environment :

2lfielddev.txt

**INCLUDE <filename>**

Include commands from another <commandfile> into the active command file (F+S)

**REM <remark>**

Put a remark line in your command-file (CMD) (F+S)

**ONFAIL :Label**

If one of the next commands in the command-file fails, the command-file will be resumed from :Label (CMD) (F+S)

**GOTO :Label**

Go to the next :Label in command file (CMD) (F+S)

**GOBACKTO :Label**

Continue executing command file from a previous :Label (CMD) (F+S)

**RESTART**

Continue executing command file from the beginning (CMD) (F+S)

**:Label**

Label-line to be used with ONFAIL, GOTO and GOBACKTO commands (CMD) (F+S)

**FAIL**

Stop executing command immediately with a fail (CMD) (F+S)

## **Filterfile commands**

**FILTER** <mode>

Set the filter-mode ON or OFF (F+S)

**SETFILTER** <Fieldname> <Condition> <Value>

Filter records on the condition <Fieldname> <Condition> <Value>.

<Condition> can be =, <=, => or <>

Multiple conditions have to be separated as condition1 | condition2 |...| conditionx (F+S)

**FILTERDAT** <fnr> <dataset> <filterdata> <condition>

Filter <dataset> using <condition> on column <fnr> into <filterdata>

Note:the header of <dataset> will be copied to <filterdata> (F+S)

**FILTERTXT** <fnr> <textfile> <filtertext> <condition>

Filter <textfile> using <condition> on column <fnr> into <filtertext> (F+S)

**FILTERLST** <fnr> <dataset> <filtertext> <exportcolumns> <condition>

Filter <dataset> using <condition> on column <fnr> into <filtertext>. Only those columns as specified in <exportcolumns> (comma separated list) will be output, the header record will not be included. (F+S)

Note: if condition is empty, all records will be used in the output.

**FILTERCOLS** <Colidx> <Dataset> <Filterreddataset> <Exportcolumns> <Condition>

Filter <Dataset> in column <Colidx> to <Filterreddataset> (output only Specified <Exportcolumns>) (F)

## **Timer commands**

Each form can have up to 10 user timers to perform periodic tasks when the form is active. Timeouts can be specified in seconds (default) minutes (e.g. 10m) hours (e.g. 1h). Any command or command-file can be used as <command>. Timers will automatically be stopped and deleted when the form or sheet is closed. (F+S)

**SETTIMER <timer> <timeout> <command>**

Create a user timer (from 1 to 10), to run the <command> each <timeout>.

Note, SETTIMER does NOT start the periodical execution of <command>. For this you need to call STARTTIMER <timer> or STARTTIMERS.

**STARTTIMER <timer>**

(Re)start timer <timer> now

**STARTTIMERS**

(Re)start all user timer now

**RESETTIMER <timer> (<newtimeout>)**

Reset timer <timer> to its original timeout (or to <newtimeout> if specified)

**STOPTIMER <timer>**

Stop timer <timer> now, until STARTTIMER <timer> is called

**STOPTIMERS**

Stop all timers now

**KILLTIMER <timer>**

Stop and delete timer <timer> now. The timer cannot be started again, unless it has been recreated with SETTIMER.

**KILLTIMERS**

Delete all user timers

## **Other commands**

**RUN** <filename>

Execute <filename> (F+S)

**RUNW** <filename>

Execute <filename> and wait for the execution to finish (F+S)

**PW** <password> (<inputtext>)

Let the handheld user enter a password in a pop-up input screen (using <inputtext> as prompt). The user input must be equal to <password> to continue. (F+S)

**RANDOM** (<minvalue>) <maxvalue>

Generate a random value between 0 (or <minvalue>) and <maxvalue>. Store this value in variable `_RANDOM` (F+S)

**SETFIELDPOS** <fldname> <newrow> <newcol>

Move <field> (any data-field or button) on the fly to <newrow> <newcol> position. Note this new position is lost when the form is re-opened. (F)

**SETROWPOS** <fldname> <newrow>

Move <field> (any data-field or button) on the fly to <newrow> position. Note this new position is lost when the form is re-opened. (F)

**SETCOLPOS** <fldname> <newcol>

Move <field> (any data-field or button) on the fly to <newcol> position. Note this new position is lost when the form is re-opened. (F)

**WAIT** <waittime>

Stop execution for <waittime> milliseconds (F+S)

**NEWDATE** <Fldname> <Datestring> <Count>

Calculate a new date from <Datestring> (in DATEFORMAT) + <Count> nr of days (Count can be negative), display the new date in <Fldname> (F+S)