
EccoExtDoc

Table Of Contents

&nbsp;imatch()	4
add_depend_item()	5
allc()	6
alld()	7
allp()	8
anyc()	9
anycf	10
anyd()	11
anyf()	12
anyp()	13
anypf()	14
Arithmetic Operators	15
Auto Assign Rules	16
avgc()	19
avgd()	20
Boolean Operators	21
cfn()	23
cft()	24
cfv()	25
Condition	26
countc()	27
countd()	28
countp()	29
create_item()	30
create_sub_item()	31
Dave's note on how hard this is..	32
Definitions	33
doubleplusgood	36
Ecco Links	38
eccoext4nerds	39
eccoext4poets	40
Expression	43
Features	44
Features Summary	46
Flags Values Conditions	48
free rar tools	52
fv	53
fvc()	54
fvd()	55
getc()	56
getdv()	57

getpv()	58
get_contain_date()	59
get_contain_date()lua	61
get_depend_items()	63
get_folder_items()	64
get_folder_value()	65
get_item_children()	66
get_item_text()	67
get_item_type()	68
get_select_items()	69
gft()	70
gfv()	71
gifv()	72
gpfv()	73
has_contain_date()	74
has_contain_date()lua	75
has_depend_item()	76
here	77
home	78
How_do_I_contribute	79
iff	80
il()	81
Index	82
Installation and Getting Started	83
Introduction	84
ireplace()	85
item_id	86
itl()	87
left	88
len()	89
List of Contributors	90
lower	91
Lua launch bug	92
match()	93
maxc()	94
maxd()	95
maxp()	96
minc()	97
mind()	98
minp()	99
msgbox()	100
now()	101
Quick Start	102
remove_depend_item()	106
replace()	107
right	108

Sample LUA Programs	109
Sample Program 1	111
Sample Program 2	113
Sample Program 3	115
Sample Program 4	117
Sample Program 5	118
Sample Rule 1	120
Sample Rule 2	123
Sample Rule 3	128
Sample Rule 4	130
Sample Rule 5	131
Sample Rule 6	132
Sample Rule 7	133
Sample Rules	134
security warning	135
set_folder_value()	136
set_item_text()	137
side issues and background	138
slangs_original_autoexec_doc	139
sub items not in folder	143
substr	144
sumc()	145
sumd()	146
Table of Contents	147
today()	148
trim()	149
Tutorials	150
TV	151
tvc	152
tvd	153
upper	154
USB Mode	155
Using Dependencies	159
Valuable Resources	165
What Is Ecco	166
Whats_a_wiki	167
Why use a Wiki instead of a forum?	168

imatch(string,pattern)

Description

This is the same as `match()` except that the case (upper or lower) is ignored
Returns that part of the string that matches the pattern
string can be any valid string or reference to a folder or item
the pattern can be any Regex pattern, see examples

Example

```
imatch(T 123456 234567,\s*t\s+(\d+)\s+\d+\s*)  
returns the string 123456
```

Related Rules

[match\(\)](#), [replace\(\)](#), [ireplace\(\)](#)

add_depend_item(item_id, ditem_id)

Description

This function allow the user to set dependencies through program control

Example

```
for inc1 = 2,MaxNumber do  
add_depend_item(NumberItems[inc1],NumberItems[1])  
end
```

this loop will cycle through a selected list of items and make the last item dependent on all the prior selected items

allc(condition)

Description

return true if all child items meet the condition.

Example

allc(substr(tv(),1,2)=="M ")

will return a true value only if the item text for all child items start with "M "

TV() returns the item text of the child item to the substr() function.

Related Rules

[alld\(\)](#), [allp\(\)](#), [anyc\(\)](#), [anycf\(\)](#), [anyd\(\)](#), [anyf\(\)](#), [anyp\(\)](#)

[Rules](#)

alld(condition)

Description

Just like the allc() function but it works on depends items.
Returns true if all depends items meet the condition.

Example

alld(has_contain_date())

checks the item text for each depends item and if they all contain a valid date returns true

alld(substr(tv(),1,2)="M ")

Will return a true value only if the item text for all depends items start with "M "

TV() returns the item text of the depends item to the substr() function

Related Rules

[allc\(\)](#), [allp\(\)](#), [anyc\(\)](#), [anycf\(\)](#), [anyd\(\)](#), [anyf\(\)](#), [anyp\(\)](#)

[Rules](#)

allp(condition)

Description

Just like the allc() function but it works on parent items.
Returns true if all parent items meet the condition.

Example

allp(has_contain_date())

checks the item text for each parent item and if they all contain a valid date returns true.

allp(substr(tv(),1,2)=="M ")

will return a true value only if the item text for all parent items start with "M "
TV() returns the item text of the parent item to the substr() function.

Related Rules

[allc\(\)](#), [alld\(\)](#), [anyc\(\)](#), [anycf\(\)](#), [anyd\(\)](#), [anyf\(\)](#), [anyp\(\)](#)

[Rules](#)

anyc(condition)

Description

similar to allc() except that it returns true if any of the child items meet the condition.

Example

anyc(substr(tv(),1,2)="M ")

will return a true value only if the item text for any child items start with "M "

TV() returns the item text of the child item to the substr() function

Related Rules

[allc\(\)](#), [alld\(\)](#), [allp\(\)](#), [anycf\(\)](#), [anyd\(\)](#), [anyf\(\)](#), [anyp\(\)](#)

[Rules](#)

anycf([folder_name], condition, [recursive])

Description

Check to see if any subfolder satisfies the condition.

"Recursive" means that all descendants are checked (not just the immediate children). The default is that "recursive" is true.

When only one argument is specified it is assumed to be the condition and the folder_name is assumed to be the folder where the rule is applied

When the first argument is a string it is treated as the name of the folder to be checked and the second argument is the condition

When the first argument is not a string it is treated as the condition and the second argument is recursive

Examples

anycf(cfv())

is true if any descendant of the folder has any value

cfv() returns true if the folder has value false if not

anycf("Notepad 1", cfv())

is true if there are subfolders of the folder "Notepad 1"

anycf(cfv(), false)

means any child of the self folder has any value

anycf("Notepad 1", cfv(), false)

means any child of the folder "Notepad 1" has any value

anycf(cft()==1)

Will return a true value only if there is a child folder with type 1 (Checkmark see [gft\(\)](#))

[cft\(\)](#) returns the folder type for any folder.

Related Rules

[anyf\(\)](#), [anypf\(\)](#)

anyd(condition)

Description

Similar to anyc() except that it returns true if any of the depends items meet the condition.

Example

```
anyd(substr(tv(),1,2)=='M ')
```

Will return a true value only if the item text for any depends items start with "M "

TV() returns the item text of the depends item to the substr() function.

Related Rules

[allc\(\)](#), [alld\(\)](#), [allp\(\)](#), [anycf\(\)](#), [anyc\(\)](#), [anyf\(\)](#), [anyp\(\)](#)

anyf(condition)

Description

Similar to anyc() except that it returns true if any of the folders meet the condition.

Example

anyf(cft()==1)

Will return a true value only if there is a folder with type 1 (Checkmark see [gft\(\)](#))
[cft\(\)](#) returns the folder type for any folder.

Related Rules

[anycf\(\)](#), [anypf\(\)](#)

anyp(condition)

Description

Similar to `anyc()` except that it returns true if any of the parent items meet the condition.

Example

`anyp(substr(tv(),1,2)=="M ")`

Will return a true value only if the item text for any parent items start with "M "

TV() returns the item text of the parent item to the `substr()` function

Related Rules

[allc\(\)](#), [alld\(\)](#), [allp\(\)](#), [anyd\(\)](#), [anycf\(\)](#), [anyd\(\)](#), [anyf\(\)](#)

anypf([foldername], condition)

Description

like anycf, but check the parent

Example

see [anycf\(\)](#) for examples

anypf(cft)==1)

Will return a true value only if there is a parent folder with type 1 (Checkmark see [gft\(\)](#))
[cft\(\)](#) returns the folder type for any folder.

Related Rules

[anycf\(\)](#), [anyf\(\)](#)

Arithmetic Operators

There are only four operators that are permitted in expressions and calculations as follows

+
add two numbers, strings, or an integer to a date

-
subtract two numbers, dates, or an integer from a date

multiplication of two numbers

/
division of one number by another

General

Auto Assign rules take on the general form:

- `++:/*flag*/:/*expression*/:/*condition*/`

Where the [Flag](#) determines when the rule will execute, [Expression](#) (or value) is a build up of functions or rules that are what you want to do and [Condition](#) is a logic expression that if true will cause the expression to execute if false will not. Following are the specific functions or rules that are built into eccoext. There are two distinct classes of functions, "Expression" and Lua scripts. Lua scripts will only work as a launch tool, (LUAscript: {function name of function in luacmd.lua file}) or in a rule when the trigger flag includes "L".

Expressions

[allc\(\)](#)
[alld\(\)](#)
[allp\(\)](#)
[anyc\(\)](#)
[anycf\(\)](#)
[anyd\(\)](#)
[anyf\(\)](#)
[anyp\(\)](#)
[anypf\(\)](#)
[avgc\(\)](#)
[avgd\(\)](#)
[cfn\(\)](#)
[cft\(\)](#)
[countc\(\)](#)
[countd\(\)](#)
[countp\(\)](#)
[cfv\(\)](#)
[eval\(\)](#)
[fv\(\)](#)
[fvc\(\)](#)
[fvd\(\)](#)
[gfv\(\)](#)

[get_contain_date\(\)](#)[getcfv\(\)](#)[getev\(\)](#)[getdv\(\)](#)[getmin\(\)](#)[getmax\(\)](#)[getpfv\(\)](#)[getpv\(\)](#)[gft\(\)](#)[gfv\(\)](#)[gifv\(\)](#)[gpfv\(\)](#)[has_contain_date\(\)](#)[iff\(\)](#)[il\(\)](#)[itl\(\)](#)[left\(\)](#)[len\(\)](#)[lower\(\)](#)[match\(\)](#)[imatch\(\)](#)[maxc\(\)](#)[maxd\(\)](#)[maxp\(\)](#)[minc\(\)](#)[mind\(\)](#)[minp\(\)](#)[now\(\)](#)[replace\(\)](#)[ireplace\(\)](#)[right\(\)](#)[substr\(\)](#)[sumc\(\)](#)[sumd\(\)](#)[today\(\)](#)[trim\(\)](#)[TV\(\)](#)[TVC\(\)](#)[TVD\(\)](#)[upper\(\)](#)

Lua Expressions

The following expressions are LUA functions that allow LUA to interact with ecco. In addition to these functions the entire LUA programming language is available for use. Many of the examples as part of these functions include some LUA commands. For a complete guide to the LUA programming language reference should be made of the online [LUA manual](#) is one that may be of use.

[add_depend_item\(\)](#)
[create_item\(\)](#)
[create_sub_item\(\)](#)
[debug\(\)](#)
[get_contain_date\(\)](#)
[get_depend_items\(\)](#)
[get_folder_items\(\)](#)
[get_folder_value\(\)](#)
[get_item_children\(\)](#)
[get_item_text\(\)](#)
[get_item_type\(\)](#)
[get_select_items\(\)](#)
[has_contain_date\(\)](#)
[has_depend_item\(\)](#)
[item_id](#)
[msgbox\(\)](#)
[remove_depend_item\(\)](#)
[set_folder_value\(\)](#)
[set_item_text\(\)](#)

avgc(value [, condition])

Description

Returns the average value of the child items when condition is true.
If the condition is not included, it is assumed to be true.

Example

```
avgc(fv("times") * fv("Price"), fv("Price") > 1)
```

Returns the average value of the value of the [times] folder times the value of the [Price] folder where the value of the [Price] folder is greater than 1avgc(value [, condition]).

Related Rules

[avgd\(\)](#)

avgd(value [, condition])

Description

Behaves similarly to [avgc\(\)](#) function, but uses depends items.

Returns the average value of the depends items when condition is true.

Example

```
avgd(fv("times") * fv("Price"))
```

Returns the average value of the value of the [times] folder times the value of the [Price] folder there is no condition therefore a true condition is assumed.

Related Rules

[avgc\(\)](#)

Boolean Logic Operators

=

- equals

==

- Equals

!

- not
- do not confuse this with the flag usage of the symbol. See [Rules](#) for its application in flags.

!=

- not equal

<

- Less than

>

- greater than

<=

- less than or equal

>=

- greater than or equal

<>

- not equal

~=

- not equal

?=

- equal ignore case

~=

- Regexp include string

~~=

- Regexp include string ignore case

cfn()

Description

Only used in [anycf\(\)](#), [anypf\(\)](#), and [anyf\(\)](#), to get the folder value being checked of the current item.
returns the name of the folder

Examples

see examples in [anycf\(\)](#)

anycf(cfn()=="Status"))

is true when there is a child folder with the name Status

anycf(cfn()=="Orphan"))

is false when there is no child folder with the name Orphan

Related Rules

[anycf\(\)](#), [anyf\(\)](#), [anypf\(\)](#), [cft\(\)](#), [cfv\(\)](#)

cft()

Description

Only used in [anycf\(\)](#), [anypf\(\)](#), and [anyf\(\)](#), to get the folder value being checked of the current item. Returns the type of folder see [gft\(\)](#)

Examples

see examples in [anycf\(\)](#)

Related Rules

[anycf\(\)](#), [anyf\(\)](#), [anypf\(\)](#), [cfv\(\)](#), [cfn\(\)](#)

cfv()

Description

Only used in [anycf\(\)](#), [anypf\(\)](#), and [anyf\(\)](#), to get the folder value being checked of the current item. SBF is the same as `anycf(cfv())`

Examples

see examples in [anycf\(\)](#)

Related Rules

[anycf\(\)](#), [anyf\(\)](#), [anypf\(\)](#), [cft\(\)](#), [cfn\(\)](#)

Condition

(or when when the Expression or value will be applied)

Description

The condition may also be an [Expression](#) however the result will be either true or false. If the Condition is true then the calculated value will be applied if false then it will be left blank or empty. There are some special rules that can be applied to the condition over and above the functions that are included in the expression.

Examples

++:++:SBF

Check the assigned folder to see if any of its subfolder(s) is(are) checked. The current folder item will be checked, set true, if the condition is true otherwise no change.

++:++:ITT~~='hello'

If the Item Text contains the string 'hello' the assigned folder will be checked, set true, and if not, the folder will be unchecked.

++:++:[Done]:[Done] && ITT ~~= 'hello'

Set folder value to <Done> value if the item text contains string 'hello' (It can be used as conditional Merge Column Value)

ITT ~= 'hello' && <Done>

Item text has the string 'hello' and the folder Done is set

ITP~=1 && <Done>'20070101'

Item type is text (1) and folder Done is '20070101'

countc([condition])

Description

Returns the number of child items under the item text.
If no condition then the condition implies all child items
The count does not include the child itmes of child items.

Examples

countc()

returns the number of child items for the current item

Related Items

[countd\(\)](#), [countp\(\)](#)

countd(condition)

Description

just like countc function, but it's use on depends items

Returns the number of dependant items under the item text.

If no condition then the condition implies all dependant items

Example

countd()

returns the number of dependant items for the current item

Related Items

[countc\(\)](#), [countp\(\)](#)

countp(condition)

Description

just like countc function, but it's use on depends items
Returns the number of parent items under the item text.
If no condition then the condition implies all parent items

Example

countp()

returns the number of parent items for the current item

countp(il(>0))

returns the number of parent items on item levels greater than 1 (excludes the TLI)

Related Items

[countc\(\)](#), [countd\(\)](#)

create_item(text)

Description

This function allow the user to enter a new item into ecco through program control
When created the item_id of the item is returned to the program.

Example

```
set_folder_value("Instructions",create_item("test item text"),1)
```

This line will create an item text "test item text" and set the Instructions folder to true.

The Instructions folder in this case is a checkmark folder and the 1 forces it to be checked

Related Rules

[create_sub_item\(\)](#)

create_sub_item(item_id, text)

Description

This function allow the user to enter a new child item to the current item_id into ecco through program control

When created the item_id of the child item is returned to the program.

Example

```
Pitemid=create_item("This is a test of adding a child item")
```

```
set_folder_value("Release",Pitemid,"3.x.x")
```

```
Citemid=create_sub_item(Pitemid,"This is a child item")
```

```
set_folder_value("Release",Citemid,"4.x.x")
```

These lines of code will create an item with text "**This is a test of adding a child item**" and set the Release folder value to 3.x.x.

Using the itemid of the parent a new child item is created "**This is a child item**" and the Release folder value is set to 4.x.x

Related Rules

[create_item\(\)](#)

It is sometimes really difficult to write a lay-person's explanation of a difficult concept. (I really don't like the term "dummie's guide" but the term has seemingly been adopted to mean: "a well written guide for an intelligent person to understand a subject which is not in their own particular area of expertise...")

I taught high-school math for a couple of years after college, and clearly recall the realization one day while teaching a Calculus class that when someone didn't understand what I said, that repeating the same words only louder did absolutely nothing to improve comprehension...

I'm sitting here looking at a book titled: "Relativity: The Special and the General Theory" written by a man named Albert Einstein in 1916. In the preface he says: "The present book is intended, as far as possible, to give an exact insight into the theory of Relativity to those readers who, from a general scientific and philosophical point of view, are interested in the theory, but who are not conversant with the mathematical apparatus of theoretical physics. The work presumes a standard of education corresponding to that of a university matriculation examination, and, despite the shortness of the book, a fair amount of patience and force of will on the the part of the reader."

With this as inspiration, I will attempt to at least pique the interest of some non-technical readers who love Ecco as much as I do, and who might benefit from embracing some of the new possibilities opened up by the eccoext code.

Definitions

Though the following information is available from the eccoext.eco and example.eco files, and other documents, It has been useful to accumulate the information and bring it together as some of the terms, though understood by Ecco users, are perhaps not as familiar to casual users of Ecco Pro. This list includes native Ecco terms as well as Eccoext, Regex and Lua terms all of which are used to one degree or another in the extension to Ecco Pro.

()

- use to group expression
- used to contain function parameters
- in patterns used to show terms to be matched

{1}

- means the first Regex subexp

{0}

- means use the whole match string

{n}

- means the nth subexp string
- there may be more than one subexpression extracted and each extraction is numbered accordingly
- i.e. `ireplace(ITT,"(w?)\d{2})\d{3}", "H:\20$2\3\")`

definitions above do not match example - see discussion on this page - [Alec Burgess](#) Oct 12, 2007 8:09 pm

[folder_name]

- means the value of the item in the current folder
- @ prefix references the dependent item folder value
- ^ prefix references the parent item folder value
- For Example:
 - [Done] uses the value of the Done folder;
 - [@Done] uses the value in the dependent item Done folder; and
 - [^Done] uses the value in the parent item Done folder.

Arithmetic Operators

- addition, subtraction, multiplication and division

Boolean Operators

- logic operators such as equals, greater than etc.

IFC

- Item folder count
- Returns the number of folders associated with the current item

ITT

- means Item Text
- Returns the contents of the item text for use in expressions

ITP

- means Item Type

PTT

- means Parent Item Text
- Returns the item text of the nearest parent to the currently selected item
- returns a null value for all TLI

SBF

- means if there is child or sub child folder set

SLI

- sub level item

TLI

- Top Level Item in any notepad

A guide to autoassignment

Autoassignment is a way to sort material automatically into ecco folders depending on its content. It's useful and powerful in all sorts of circumstances - for instance, we want Ecco to recognise that web addresses are special, and put them into the "net location" folder so that they can be double-clicked and then visited at once.

Autoassignment is where most of the hidden magic powers of Eccoext are concealed and it needs clear and careful documentation. There is [some in the Eccoext.eco file](#) that comes in eccoext.rar, but it is very well concealed: in version 3.6.2, for example, it is item 76 in the change log. It is, however, written for hard-core geeks. I have copied it [here](#) and if you are fluent in regular expressions or pattern matching you can go there straight away. The rest of this page is for people who are not.

The non-geek version

Like everything else that computers can do, autoassignment proceeds by rules. That's not the way we express ourselves in natural language. So to get from what we want ("do something clever and useful") to what the computer can understand is a two or three step process. First decide what you want; then turn it into a set of rules; then translate those rules into eccoext.

Eccoext understands only rules that say *If this do that*

More particularly, its rules all have the form: if **some thing** is true about **some item** in ecco, perform **some action**.

An example: in natural language you want to move all todo items more than three days overdue to a folder marked urgent

The wish can be rephrased as a rule:

if **any to-do item** is **more than three days old**, put it in **"Urgent"**.

Which is then turned into Eccoext

```
++::: [Urgent] : [To Do] > (TODAY-3)
```

That last step is the magic one, and needs unpacking ...

All eccoext rules have the [general form](#):

++:/*flag*/:/*expression*/:/*condition*/

The home of the eccoextension:
[the Yahoo! Groups NEW Ecco_Pro](#)

Official Bug report site for the ecco extension:
<http://www.EccoBug.com>

Official extension Help forum:
<http://www.SlangHelp.com>

Where to get other FREEWARE add-ons:
<http://www.EccoTools.com>

Homesite of:
YSWT
[eccoMagic](#)

David G.
[Cold-Mountain Software](#)
(where you can check out his very cool TranSender add-on for eccoPro)

David's helpful ecco freeware site:
[Sourceforge](#)

LUA Programming Tutorial manual
<http://lua-users.org/wiki/LuaTutorial>

sites YSWT gives cautions about:

Background: These sites are controlled by the same person, beware.

1. What used to be the home of ecco life, but which is filled with email harvesting bots --> use the hide email option!!!, and which has become filled with ridiculous mis-information about ecco. Intentional suppression of posts, etc.. etc.. so beware!!

the "[old](#)" [EccoPro](#) yahoo list (see [full text of caution](#) by YSWT)

2. What is somewhat of a trick to get \$10.00 from you,

[Compusol](#)

there is simply nothing there... it's all been outdated and superseded by the new extensions and add-ons and developments. Better 'zip' install files are available for FREE, the 'utilities' are all outdated. Others have used the word "scam", but that is subjective.

We are all waiting with great anticipation for slangmgh to enlighten us here...

OK, at great risk of over simplifying, this is an attempt to describe how eccoext works... ([Dave's note on how hard this is...](#))

Please please please, add to and clarify this!

Below is Dave's very lucid insight and explanation, to which I add:

1) Some of the incredible fixes are plain, old-fashioned, one byte changes to the ecco code, allowing flexible multi-folder pulldowns, proper OLE object copying, numbering past 999, etc. On one hand it's something so tiny, that produces huge results. On the other hand note it is pure Slang genius behind the code breakdown/analysis and discovery of 99.9% of these key fixes. Simply astounding to me personally how clearly and deeply Slang has been able to decypher the code and pinpoint the ultimate and key 'fix' points.

2) Some of the incredible power is just plain old great concept + brilliant design + solid coding!! The structure of key folder assign tool is the same add-on tools have always been, using Ecco's built in DDE interface to manipulate in a programmed way the underlying data. True, that we can directly call the internal DDE routines without needing the DDE messaging, bypassing the middle-man so to speak, but that is a recent Slang breakthru bonus... the tools worked really great even before...

3) One key is the 'sandwich' approach. Using monitoring of the 'front end' (user input/actions), with 'back end' DDE based data manipulations. With this, amazing innovation has been possible. Add to this Slang's brilliant eye for detecting and uncovering the structures and routines hidden in the Ecco assembly code, and neat, otherwise impossible tricks become possible-- locking folders, item focus, etc.. and there in a nutshell, you have it!

Note that this is my current understanding and I certainly hope that my knowledge will evolve over time. My view will expand by having others revise and expand this page with corrections and additional insights. I certainly hope that it can ultimately give some insight into eccoext that will enable non-technical users to appreciate the possibilities that are opened by eccoext.//

For the most part computers just do one thing at a time, tediously executing instructions in the order that a programmer provided them.

Computers are VERY predictable, and unless there is a hardware failure do things the same way every time. (Ecco has not changed in over a decade, so it is VERY predictable)

In the "good old days" computer programs were "patched" by choosing a place in a program where you wanted to have something behave differently, and inserting a different instruction. Typically if the program looked like this:

```
000022 Do thing A
000023 Do thing B
000024 Do thing C
000025 Do thing D
```


you could replace instruction #000023 with something different, something like Goto 000087 (one of the first instructions that computers understood is to go to a different place for then next instruction, rather than to the next sequential instruction, sort of like a treasure hunt...)

so now our "program is something like:

```
000022 Do thing A
000023 Goto 000087
000024 Do thing C
000025 Do thing D
...
000087 Do thing B
000088 Do thing X
000089 Do thing Y
000090 Goto 000024
```

The computer certainly doesn't know what's going on, it just goes where it is told to go, and does what it is told to do... [Note that "Do thing B" still is executed, just that the instruction to do it was moved to make room for the "goto" instruction...]

In a more modern and complex world, it is possible for one program to run or "load" another program. If done correctly, the "loader" has complete control of the program that is loaded. So, one program can load another, and once the loaded program is under the loader's control, the loader can "inject" new instructions into the program similar to the "patches" from the old days...

By carefully monitoring how a program runs, it is possible for a talented programmer to figure out precisely what instruction is called every time a program such as Ecco does a particular thing, for example, when an item is changed or added.

If a "loader" program that "wraps" Ecco first: starts Ecco, then inserts a "patch" in the right place, it is possible to have new added code executed.

Ecco provided an "application programming interface" or API so that external programs could interact with Ecco and Ecco's data. This interface used a mechanism called DDE to pass messages back and forth between an external program and Ecco. Though very useful, many of the API features seemed to be provided for synchronization programs. The ability to interact with Ecco's auto-assign rules, presentation (like colors), and notepad filters were not in the API. The API also did not allow real-time interaction where an external program would be notified if something changed in ecco. The external programs needed to periodically check for changes and performance wasn't very good.

eccoext appears to work by implementing a "wrapper" that "loads" Ecco, and then "injects" patches or "hooks" that allow the loader to take control whenever the auto-assign code would normally be executed. Then eccoext appears to use other "hooks" to call the same code that would be used by the API to manipulate the data such as folder assignments, etc. Because eccoext appears to use the same code that was used by the API for manipulating the Ecco data, it can be assumed that it is quite reliable.

One of the more advanced features of eccoext is the ability to execute embedded code or "macros" when an auto-assign rule would normally be applied. The macros use a language called "Lua" which is heavily used in the computer game industry. Lua is a "high level" "dynamically typed" language that is similar in many ways to Python and Ruby. It turns out that Lua is a fairly obvious fit (once we've been shown the proof) for creating an embedded macro language for Ecco. Lua is also targeted at embedded applications such as inclusion in handheld devices and cell phones. It is my understanding that the Lua libraries provide most of the regular expression handling that is exposed by eccoext. (actually, if not mistaken, that is the perl REGEX engine doing that magic). Again, we can feel assured that the artful matchup of Ecco and Lua relies on lots of code that has already seen many years of refinement.

One thing that many of you might be thinking about these techniques is: "isn't that the way malware and viruses are created?", and you are quite correct in making that connection. But these "black hat" techniques are also used extensively in advanced development tools that are created for debugging and profiling programs during development. Their application to Ecco gives an incredible "new life" to this code that has not changed in a decade...

Enough for tonight...

DaveG

Expression or value

(what is entered into folder when expression is true)

Description

This may be any valid rule or combination of rules that results in a value that will be entered into the folder. Some basic rules are:

- for checkmark type folder, value may be left empty
- for date type folder, empty value means Today
- for other type folder, empty value means delete the current folder value
- value can have the same syntax as [Condition](#), see follows

Examples

See also the [Rules](#) for examples with specific functions.

Eccoext improves Ecco in two main ways. It fixes old irritations, and adds wholly new features.

First, a table of bugs/ annoyances which have been fixed.

Ecco Limitation	Eccoext Improvement
Mouse scroll wheels didn't work	They do now
Recurring dates failed after the year 2000	They don't now
Sorting sub items in outlines was limited	All levels can now be easily sorted
Searching was a little clumsy	It is now quick and easy to jump to the next occurrence of selected text
Lists could only be numbered up to 999 items	You can have up to 64,000 now
Multi Folder columns had to be edited in a cramped and inconvenient window	You can now see as many as you want
Some Ecco entries might get lost ("become orphans") if their parent folders vanished	This no longer happens: all orphan items now have their own folder
Copying information in and out of Ecco was difficult	Eccoext handles more formats (html, rtf ...) for getting text out of Ecco and makes copying text into it slicker and easier

Then there are the Features that no one imagined Ecco might be missing.

[(Each of these could use a page of explanation. Please feel free to write it.)]

1. You can assign most any hotkey to most any ecco function. EASILY! (just like in Word... ;)
2. You can see how many items are in each and every folder...
3. SUPER POWER AUTO-ASSIGNMENT RULES... including LUA syntax and REGEX matching...
 1. AUTO ASSIGNMENT CAN BE BASED ON:
 1. Other folder values
 2. Item Text
 3. Parent's folder values or item text
 4. Children's folder values or item text
 5. Linked item's folder values or item text
 6. Dates and times in the text of an item

2. Assign calculated values.. (not just checkmarks)...
4. You can Control the display color of text items via the folder rules
5. You can copy from an Ecco Outline in HTML format and then paste into notepad or some other text editor to get *instant html!!!*
6. You can make items dependent upon other items. For instance, you can make an item change to a 'to-do' only when another item has been completed.
7. You can fill in value of Field based on lookup of value of item text of field of another linked item.
8. You can run EccoPro from a USB memory without Admin priv.
9. You can lock items so that you don't accidentally erase them.
10. You can start minimized to tray/Close to tray/Minimize to tray
11. Tooltips now show context/folder/depends information
12. Some other programs have gained a "Send selected text to Ecco" right click menu: very quick and simple. This doesn't work with MS Office or Firefox, though. You can use [a global hotkey](#) with them. [needs explanation]
13. You can now write programs in many languages (perl, Python, JScript, VBScript and LUA) they are activated upon item/folder changes and access built in Ecco commands (such as create item, collect values from folders, etc.) and Ecco variables (such as currently selected item) and run them directly from inside Ecco upon trigger of event, or using the Launch menu. To access Python or perl, need installed on your system the language's "active" installation. LUA is built into the extension, and JScript and VBScript are built into most modern Windows systems.
14. Copying and pasting have been improved
 1. You can copy without subitem/Paste Block(Reduce global pointer/items)
 2. You can copy with column/Paste with column(Exchange data with Excel freely)
 3. You can copy/Paste RTF

Features in Detail

This page is for an explanation of the features in detail. Some headings are provided but please add and expand as you see fit.

LUA

Besides being able to use LUA functions within auto assign rules they may also be created and run as stand alone scripts.

LUA Auto Assign Rules

These are similar to the regular eccoext rules but the flag is set to **L**. With this flag the expression that follows is assumed to be LUA commands and execute as such. All other flags are disregarded ie +, !, F I etc. The condition may still be set which will determine if the rule will be executed or not but the expression itself must be LUA commands. See the [Auto Assign Rules](#) page for details and descriptions of the eccoext lua commands. Reference may also be made to the [LUA manuals](#) on the web for more comprehensive description of the LUA language and its functions. See also the [examples](#) that are a part of this wiki,

LUA Scripting

With the power of the LUA language there is a quasi macro language for ecco, a marvelous feat for a program that has been dead since 1998, well abandoned but not forgotten. LUA scripting may be written as stand alone functions that can extract selected ecco data and then manipulate it or store it. The method is simple.

A text file labeled luacmd.lua must exist in the c:\program\ecco directory. This is the same directory that eccoext and ecco reside. It is important that this file luacma.lua be a text file as that is what the interpreter will expect. Within the file there may be several functions each taking the form:

```
function name()  
command block
```

end

So each piece of code that you wish to execute will be of the above form and one can follow the other. It is important to have the parenthesis at the end of the name() as without it the LUA interpreter will not recognize it as a function.

Here are [sample LUA scripts](#)

flag

(or when the rule is to work)

There are a series of characters that Trigger the method of auto assigning information to folders. Each has a purpose and depending on their individual or combined use they will affect the outcome differently.

!

- If the folder already contain value, then override the value, else set the value.

+

- Set the value into the folder when the ITT is created and the folder is empty.
- Caution: If the user were to enter data into the folder manually, prior to the condition becoming true, and subsequently the condition were to become true the folder will not be updated by the value of the expression. If the manually entered information were to be later deleted and the condition is true then the value of the expression will be entered into the folder.
- See [Sample Rule 5](#) for an example of this action.

-

- The folder value will be cleared if:
 - when '-' is not used with '!' and '+', and if the condition is true, then the folder value will be set,
 - when '-' is used with '!' and '+', and if the condition is false, then the folder value will be clear.

T

- Indicates that the rule only applies to TLI item.
- *when used no sub item or child will be acted upon so if the expression forces a value into a folder then it will only go into the folders associated with the TLI*

I

- Rule will be triggered only when item text or item level change or item created.
- Though the rule will be automatically generated based on the value it is possible to force its action with the use of M (see below).

F

- Rule will be triggered only when the item folder value changed.
- Though the rule will be automatically generated based on the value it is possible to force its action with the use of M (see below).

L

- Indicates the use of LUA script.
- When this flag is used then +-! will not do anything.
- And you should add appropriate 'CPDIFS' flag.

C

- Rule will be triggered when any child item is changed
- Though the rule will be automatically generated based on the value it is possible to force its action with the use of M (see below)

P

- The rule will be triggered when the parent changes.
- Though the rule will be automatically generated based on the value it is possible to force its action with the use of M (see above).

D

- Rule will be triggered when any depends item changes (see depends).
- though the rule will be automatically generated based on the value it is possible to force its action with the use of M (see below)

S

- The rule will execute whenever when the file loaded.
- Though the rule will be automatically generated based on the value it is possible to force its action with the use of M (see above).

M

- Rule flag is manually set, eccoext will not automatically generate the flag.
- *version 3.6.2.12 deleted this flag due to confusion in its use may be reinstated if there appears to be a need*

N

- Indicate the rule should be treated as normal rule (not 'C'/P/'D' rule) too.
- It can be 'CPD' rule at the same time.

Z

- Indicate the normal rule (non 'CPD' rule) should be execute after all other rules have been executed.

E

- This flag will force the rule to only be activated if the "Check folder rule" or the "Check all folder rules" is explicitly executed.

X

- This flag will prevent the rule from executing.

Comments

Following is the order of execution for the flags:

1. Execute the normal rules without 'Z' flag for current item

2. Execute the 'C' rule for parent item repeatedly
3. Execute the 'P' rule for all sub-item repeatedly
4. Execute the 'D' rule for all depends item repeatedly
5. Execute the normal rule with 'Z' flag for current item
6. Check the Orphan

When execute 'CPD' rule for parent item/sub-item/depends item, if there is folder change, eccoext will execute the rule nestly for this item.

The 'C'/P'/D' rule can be used together, ie. ++:DC!-:"High":allc(fv("Done")) && alld(fv("Done"))

If the +-! flag not set, default is +

if the flag 'F' and flag 'T' is not set, default is 'FI'

Examples

T+

Execute the rule when the item is first *created* (+),

Only enters a value in the TLI folder, will not enter values into subitems or child items (T).

C+

The same as T+ but executed only when the child item has been changed (C).

T!

Adds or changes the folder value when the item text has been modified (!),

Only enters a value in the TLI folder, will not enter values into subitems or child items (T)

C!

The same as T! but will add or change the folder value when the child item has been changed (C).

T!-

Adds or changes the folder value when the item text has been modified (!),

Only enters a value in the TLI folder, will not enter values into subitems or child items (T),

If the condition is false then the folder will be cleared (-).

[ExtractNow](#) is a simple utility that allows you to extract multiple archives quickly and easily. ExtractNow is not a complete archival solution. It's main purpose it to allow the user to *extract multiple archives easily*. Free / Donation-ware.

[7-zip](#) is another great free utility that handles .rar files. Cannot create .rar however.

There is a list of other tools at [freedownloads center](#), or you can always use Google...

FV(folder_name)

Description

returns the content of the "folder_name" of the item being processed
used within other rules

Example

getpv(fv(Release), il()== 0)

returns the content in the "Release" folder of the top level item (il()==0)

sumc(fv("price"),gfv("sample")==fv("sample"))

will sum the values in the folder "price" for all child items as long as the value in the child item folder "sample" equals the value in the parent item folder "sample".

Related Rules

[FVC\(\)](#), [FVD\(\)](#), [TV\(\)](#), [TVC\(\)](#), [TVD\(\)](#), [gfv\(\)](#)

Other References

http://tech.groups.yahoo.com/group/ecco_pro/message/1785

FVC(folder_name[, condition])

Description

returns the contents of the folder associated with first child item if the condition is true
Otherwise the result is left empty

Examples

FVC(Release)

returns the contents of the folder Release of the first Child item
the lack of a condition implies always true

FVC(Release,left(getpv(tv(), il) = 0),1)~')

will only return the contents of the Release folder of the first child item if the first character on top level item TLI text is ~

Related Rules

[FV\(\)](#), [fvd\(\)](#), [tv\(\)](#), [tvc\(\)](#), [TVD\(\)](#)

FVD(folder_name[, condition])

Description

returns the contents of the folder associated with first dependant item if the condition is true
Otherwise the result is left empty

Examples

FVD(Release)

returns the contents of the folder Release of the first Child item
the lack of a condition implies always true

FVD(Release,left(getpv(tv(), il())== 0),1)~')

will only return the contents of the Release folder of the first child item if the frst character on top level
item TLI text is ~

Related Rules

[FV\(\)](#), [FVC\(\)](#), [TV\(\)](#), [TVC\(\)](#), [TVD\(\)](#)

getcv(expression, condition)

Description

Returns the child item value in the expression if the condition is satisfied.

This function is used to find the appropriate child item and return the expression, you should specify the second arg as condition, the first arg is expression, eccoext will enumerate for all child items, and if the condition is meet eccoext will evaluate the expression (first arg), and return the result.

Example

```
getcv(fv("count") * fv("price"), fv("price") > 100)
```

Related Rules

getdv(expression,condition)

Description

This function is used to get depends item information. It returns the value of the expression, if the valuation of the condition is true. When a folder value or the item text is referenced in the expression it references the depends item. If folder values are referenced in the condition they are referenced to the current item folder values.

If there are more than one depends items then the results that are returned are for the first one in the list or the lowest numbered depends item. For instance if there are two depends having ItemID number 11 and 15 then the results that are returned are for the Item with Item ID 11 only.

Example

getdv(tv(),fv(folder_name)!=''')

returns the item text of the depends item if the value of the folder_name for the current item has some value.

getdv(fv(depends_folder_name), fv(folder_name)=='')

returns the value of the depends item depends_folder_name if it the value of the current item folder_name has no value.

Related Items

[getc\(\)](#), [getpv\(\)](#), [itl\(\)](#), [il\(\)](#)

[Rules](#)

getpv(expression, condition)

Descripton

get parent value based on the expression that satisfies the condition

Example

getpv(fv(folder_name), il()== 0)

retuns the top level item (il()==0) text

getpv(tv(), itl() - il() == 2)

returns the item text of the current item's grandfather

Related Rules

[getc\(\)](#), [getdv\(\)](#), [getpfv\(\)](#), [itl\(\)](#), [il\(\)](#)

[Rules](#)

get_contain_date()

Description

get the date expression contained in item text
Works with the following date formats

- today, tomorrow, yesterday, the day before yesterday, the day after tomorrow
- [next|last] Mon. Monday, Tues. Tuesday, etc.
- Jan. 10, February 12, etc
- 12/31, 11-25, etc
- 2008-8-1, 2009/9/1
- mm-dd-yyyy, mm/dd/yyyy

where the year is not specified the current year is assumed

Relative date expression may be used to calculate a date in the future

- +(-)xd plus or minus "x" days (eg. +5d, -27d)
- +(-)xw plus or minus "x" weeks
- +(-)xm plus or minus "x" months
- +(-)xy plus or minus "x" years
- -2y9m 6w2d (calculates from "today")

Examples

get_contain_date() for each of the following will return the indicated child item

2007/5/5

20070505

+5d

returns the date 5 days from the current date

2007/5/5+5d

20070510

Other examples of valid date expression

- 9/17 + 60d (September 17 plus 60 days)
- tomorrow + 4m
- last Friday - 7w
- next Tues + 1y4m3d (next Tuesday plus 1 year, 4 months and 3 days)
- 2009/5/22 + 2y 7m 19d

Related Rules

[has_contain_date\(\)](#)

get_contain_date(item_id)

Description

Get the date expression contained in item text.

Works with the following date formats

- today, tomorrow, yesterday, the day before yesterday, the day after tomorrow
- [next|last] Mon. Monday, Tues. Tuesday, etc.
- Jan. 10, February 12, etc
- 12/31, 11-25, etc
- 2008-8-1, 2009/9/1
- mm-dd-yyyy, mm/dd/yyyy

where the year is not specified the current year is assumed

Relative date expression may be used to calculate a date in the future

- +(-)xd plus or minus "x" days (eg. +5d, -27d)
- +(-)xw plus or minus "x" weeks
- +(-)xm plus or minus "x" months
- +(-)xy plus or minus "x" years
- -2y9m 6w2d (calculates from "today")

Examples

x = item_id;

y = get_contain_date(x)

x is assigned the item id which changes as the data file changes and hence must be checked everytime;

y is assigned the date in the item text of the item with item id x

See eccoext expression [get_contain_date\(\)](#) for other examples of valid date expression

Related Rules

[has_contain_date\(\)](#)

[Rules](#)

get_depend_items(item_id)

Description

Returns a a table of all depends items.

Example

```
x=get_depend_items(item_id)
max=table.maxn(x)
if max ==0
then
m="no"
else
m=string.format("%.0f",max)
end
msg="has ".. m .." depends items"
msgbox(msg,"")
```

This bit of code will create a table of depends item ids (x) of the current selected item.

A message is printed that says either that there are no depends or gives the number of depends items.

Related Rules

none

[Rules](#)

get_folder_items(folder_name)

Description

Return a table of all item ids for items that are in a folder.

Example

```
x = get_folder_items("Date Expressions");  
max=table.maxn(x);  
msgbox(max,"# items");  
for i=1,max do;  
msgbox(get_item_text(x[i]),"Value");  
end
```

- This block of code will show the item text for each of the items that are in the [Date Expression] folder.
- First the item ids are assigned to a table x, the maximum index of the table is assigned to max and displayed in a message box, and then the item text is displayed in a message box.
- The bold items are eccoext LUA expression the rest of the code are LUA coding.

>

Related Rules

none

[Rules](#)

get_folder_value(folder_name, item_id)

Description

returns the value of the named folder for the specific item id.

Examples

get_folder_value("Release",item_id)

This will return the value of the contents in the release folder for the current item.

The item_id will first get the current item item id then use that to determine the value of the folder.

```
x = get_selected_items()
```

```
max = table.maxn(x);
```

```
y = {}
```

```
for i=1,max do y[i]=get_folder_value("Date Stamp",x[i]) end
```

To get more than one item at time, for instance to get the values of all selected items in a notepad, first a table of ids needs to be created (x);

The array length needs to be found (max); then

A new array (y) is created with the values of the folder.

Related Rules

[set_folder_value\(\)](#)

[Rules](#)

get_item_children(item_id)

Description

Returns a table of children item ids.

Example

```
x=get_item_children(item_id)
max=table.maxn(x)
if max ==0
then
m="no"
else
m=string.format("%.0f", max)
end
msg="has ".. m .." Children"
msgbox(msg,"")
```

This bit of code will create a table of children item ids (x) for the children of the selected item.

A message (msg) is printed that says either that there are no children or gives the number of children items.

Related Rules

none

[Rules](#)

get_item_text(item_id)

Description

Returns the item text of the identified item.

Examples

get_item_text(item_id)

Gets the item text of the current item.

x=item_id

get_item_text(x)

The variable x is set to the current item id and then used in the expression to return the current item text.

x=get_select_items()

max=table.maxn(x)

y={}

for i=1,max do y[i]=get_item_text(x[i]) end

This bit of code will set up an array of item texts for each item that was selected.

- x is an array of all selected item item ids
- max is the total number of items in the array
- the for loop creates the output vector y of each item text.

Related Rules

none

[Rules](#)

get_item_type(item_id)

Description

This is a function that returns the type of the item as follows:

- 1: text
- 2:OLE
- 0: non-exist

get_select_items()

Description

Returns a table of item id values for all selected items.
May be used in further programming to modify blocks of data.

Examples

see programming examples for use.

Related Rules

none

[Rules](#)

gft(folder_name)

Description

Returns the folder type as follows:

- 1 = CheckMark
- 2 = Date
- 3 = Number
- 4 = Text
- 5 = Pop-Up List
- 0 = folder doesn't exist

Examples

gft("Rule Expressions")

will return a 1 as this is a checkmark folder

gft("Last Update")

Returns a 2 as this folder is a date folder.

iff(gft("Price")==3 AND gft("Quantity")==3,gvf("Price")*gvf("Quantity"),0)

Returns the product of the folders "Price" and "Quantity" only if both of these folders have the type Number (3).

Related Rules

[gft\(\)](#), [gvf\(\)](#)

gfv(folder_name)

Description

Returns the contents of the "folder_name" of the item or line in which the rule exists.

Example

sumd(fv("price"),gfv("sample")==fv("sample"))

will sum the values in the folder "price" for all depends items as long as the value in the depends item folder "sample" equals the value in the parent item folder "sample".

gfv("price")*gfv("quantity")

Returns the product of the values in the folders "price" and "quantity".

gfv("Depends")

Returns a null if there are no depends otherwise it will return the number of the depends items.
If more than one depends all values are returned seperated with commas

iff(gfv("Project")<>"" AND left(itt,1)<>"#",'"'+gfv("Project")+ "'"+ITT,itt)

In the conditional the contents of "Project" are tested for content and then are added to the ITT only if "Projects" has value and the ITT does not start with a #.

Related Rules

[fvc\(\)](#), [FVD\(\)](#), [TV\(\)](#), [TVC\(\)](#), [TVD\(\)](#), [gft\(\)](#), [gifv\(\)](#)

Other References

http://tech.groups.yahoo.com/group/ecco_pro/message/1785
[Rules](#)

gifv(folder_name)

Description

Returns the value in the named folder of the nearest parent item that has a value.
Equivalent to `getpv(fv(foldername), fv(foldername))`.

Examples

gifv("Release")

Returns a null if there is no value otherwise it will return the value 3.6.2.3 (the number in the Release folder for this rule, see the ecco rules.eco file).

Had the parent item Examples had a value then, it being the nearest parent item, the value of the "Release" folder associated with the Example parent item would have been returned.

Related Rules

[gfv\(\)](#), [gpfv\(\)](#), [gft\(\)](#)

gpfv(folder_name)

Description

Returns the value in the named folder of the parent item.

Examples

gpfv("Depends")

Will return a null if there are no depends otherwise it will return the number of the depends item. If there are more than one depends then all values are returned separated with commas.

Related Rules

[gfv\(\)](#), [gfv\(\)](#), [gft\(\)](#)

has_contain_date()

Description

will return true if the item text has a date value somewhere in the string false otherwise

Works with the following date formats

- today, tomorrow, yesterday, the day before yesterday, the day after tomorrow
- [next|last] Mon. Monday, Tues. Tuesday, etc
- Jan. 10, February 12, etc
- 12/31, 11-25, etc
- 2008-8-1, 2009/9/1
- mm-dd-yyyy, mm/dd/yyyy

Related Rules

[get_contain_date\(\)](#)

has_contain_date(item_id)

Description

will return true if the item text has a date value somewhere in the string false otherwise
Works with the following date formats

- today, tomorrow, yesterday, the day before yesterday, the day after tomorrow
- [next|last] Mon. Monday, Tues. Tuesday, etc
- Jan. 10, February 12, etc
- 12/31, 11-25, etc
- 2008-8-1, 2009/9/1
- mm-dd-yyyy, mm/dd/yyyy

Example

```
x=item_id;
```

```
y=has_contain_date(x)
```

x is assigned the item id of the item and

y returns true or false depending on whether there is a date expression in the item.

Related Rules

[get_contain_date\(\)](#)

has_depend_item(item_id, ditem_id)

Description

this function returns a 0 or 1 if the first item (item_id) is dependent on the second item (ditem_id)

Example

```
x=get\_select\_items\(\)
maxx=table.maxn(x)
for inc1=1,maxx do
for inc2=1,maxx do
msgbox
if has_depend_item(x[inc1], x[inc3])==1 then
msgbox("Item "..get_folder_value("ItemID",x[inc1]).." Depends
"..get_folder_value("ItemID",x[inc3]),"Confirm depends reference")
end
end
end
```

This bit of code will check all selected items and determine if they have any dependencies. There is no check in the script for self reference

Welcome to the eccoextdoc space

Getting Started

To get started, click on the 'edit' link above to add content to this page. You and other contributors can also comment on pages using the 'discussion' link at the top of every page.

About This Space

Your space is currently **protected**. It can be seen by everyone, but only space members can edit pages. As the space organizer, you may change your space's [settings](#), [look and feel](#) or [permissions and members](#). You can also change your [subscription options](#).

Need Help?

For more information on how to use Wikispaces, see our [help section](#).

Contact Us

Please contact help@wikispaces.com for assistance.

[Note: this is an example of what a missing page will look like, it isn't really missing...]

Welcome to the [eccoext](#) documentation Wiki

This is a place where we can work together to create a wiki based user guide for the exciting eccoext application.

Please feel free to join the project and make any changes to this wiki that you think are appropriate.

The eccoext application is an extremely important step forward for the Ecco community since it brings new functionality to Ecco. Note: the eccoext requires windows OS newer than Win'98.

Hopefully this user guide can evolve into a resource that will allow non-technical users to understand and use the power of this exciting development.

A discussion forum for users of eccoext can be found at [EccoMagic Forums](#).

Discussion groups for Ecco can be found at the Yahoo! "New" eccoPro list / forum [Ecco_Pro](#)

OK, so: [What's a wiki?](#) and [How do I contribute?](#) (just click on a link to see more info on that subject.) [Note: if you click on a link, and you find instructions on how to use the wiki like [here](#), then you can assume someone thinks a page needs to exist with that info, but hasn't created it yet. Feel free to edit the page and create the missing info!] Be sure to take credit for your contribution!! Sign up on the [Official Contributor's List](#) ! And if you copy information from elsewhere, i.e. a news group posting or discussion list, then it would be appropriate to make a polite attribution.



Just join the group (so that we know who you are) and get to work...

There is a button on every page that says "Edit This Page"... go for it!

Use the little "link" icon to create new pages...

First select the text you want to make into a link then click the icon...

You are also **VERY** much invited to contribute Documentation, suggestions, sample rules, etc., etc. at the extension help forum,

<http://SlangHelp.com>

feel FREE TO DUPLICATE posts there and edits here!

in fact, feel free to copy (with attribution please) SlangDoc from **anywhere** and at it in context here!!

iff(condition,true,false)

Description

executes the true statement if the condition is true otherwise the false statement is executed.

Examples

iff([checkbox],"T","F")

this expression will enter a T into the folder if the [check box] folder has a value and F otherwise.

**iff([Engineering Projects],""H:\20"+match(ITT,'\b(\d\d)\d\d\d\b')+"" +
match(ITT,'\b\d\d(\d\d\d)\b')+""""", "")**

If the folder [Engineering Projects], a checkbox folder, has value then a string based on the item text is created of the form "h:\20dd\ddd" where the 'd' represents extractions from the ITT.

If the folder [Engineering Projects] has no value then "" , null, is used.

References

il()

Description

Used within other rules to set the item level from which information is to be extracted.

Example

getpv(tv(), il() == 0)

returns the item text tv() of the current item's TLI parent

getpv(tv(), itl() - il() == 2)

Returns the item text of the current item's grandfather

Related Rules

[getc\(\)](#), [getdv\(\)](#), [getpv\(\)](#), [itl\(\)](#)

[please feel free to annotate these links...][Blank page example](#)

[A guide to autoassignment](#)

[Contents](#)

[Dave's note on "dummies" guides](#)

[Definitions](#)

[Ecco Links](#)

[EccoExt for "Geeks"](#)

[EccoExt for "Normal" people](#)

[Features](#)

[Free .rar tools](#)

[Home](#)

[How do I contribute?](#)

[Installation and Getting Started](#)

[Introduction](#)

[List of contributors](#)

[Quick Start](#)

[Rules](#)

[Slang's original eccoext doc](#)

[Tutorials](#)

[USB Mode](#)

[Using Dependencies](#)

[What is a Wiki?](#)

[What is Ecco?](#)

[Why use a Wiki instead of a forum?](#)

This page will eventually be the "long-form" version of installation and getting started. For now, you will need to rely on the "[Quick Start](#)" and on the eccoext.eco file that is included in the .rar distribution.

Some people have had problems getting the eccoext.ext file to load into ecco: they double-click on it, and Ecco starts, but without the systray icon, and without any of the extra functionality, such as brackets after folder names in the folder view window within Ecco.

If this happens, a workaround is to start ecco (ecco32.exe) first, in the normal way, and then run the command

eccoext.ext -l

from a command window.

That is to say, eccoext with the command line switch -l ("l" for "load", not digit one)

Eccoext is a piece of software that [extends the capability](#) of [Ecco Pro](#).

This is a very exciting development since eccoext includes some enhancements that were previously not possible using the original techniques for creating add-ons to [Ecco](#).

Eccoext currently provides [enhancements](#) at several different levels. At the most basic level there are some very nice ease of use enhancements like: seamless support for the scroll wheel of a mouse, correction of a few Y2K related bugs, a number of additional menu items, and visible counts of the number of assignments to folders visible in the "folder view". These additions take no effort on the part of the user to realize benefits.

At a more "intermediate" or "power user" level, there are some very exciting enhancements to Ecco's "auto assignment" feature. Rather than being limited to simple string matches (with a few "wild card" characters), eccoext enables use of "Regular Expressions" and formulas similar to what one would find as a "power user" in an application like Excel.

At an advanced level, of interest to programmers is a whole new set of possibilities for creating embedded programs or macros, and the possibility of creating extensions that rely on real-time events in Ecco.

The creator of eccoext has done a great job of applying some very advanced programming and debugging techniques to enable some exciting progress for a program that has been un-maintained for a decade.

Some links to topics of interest might be appropriate here: (please add more even if you are just creating a wish list...)

[How does eccoext work \(the "Scientific American" lay person's version\)](#)

[How does eccoext work \(the "Deep Geek" nerd's version\)](#)

[What is Ecco?](#)

ireplace(string,pattern,replace_string[,original_string])

Description

Similar to [replace\(\)](#) but ignores the case of the letters.

Replaces the matched part of the string with the `replace_string`.

Last argument is logical value, if true the original string is returned if no match found and, if false, an empty string is returned if no match is found

Example

ireplace(itt,"rules","Expressions",true)

returns the ITT except for the Related Rules where "Related Expressions" is returned

ireplace(itt,"rules","Expressions",false)

returns a blank except for Related Rules where "Related Expressions" is returned.

Related Rules

[replace\(\)](#)

item_id

Description

Returns the value of the item id for use in programming.

This value needs to be checked programatically as it changes as the data changes.

Example

x = item_id

x is assigned the value of the item id which may then be used in further programming

itl()

Description

returns the value for item level (see also [il\(\)](#))
generally used within other rules

Example

getpv(tv(), itl() - il() == 2)

returns the item text of the current item's grandfather

getpv(fv("folder_name"), itl()-il()== 1)

returns the value in the folder for the parent item one level up

Related Rules

[getc\(\)](#), [getdv\(\)](#), [getpv\(\)](#), [il\(\)](#)

Left(field,number)

Description

Returns the left most characters from the string where <number> represents the number of characters to extract

Example

left("Hello World", 5)

returns Hello, the five left characters in the string

left(ITT,10)

returns the ten right most characters of the item text

Related Rules

[right\(\)](#), [substr\(\)](#)

len(string)

Description

Returns the length of the string.

Examples

len('eccoext')

returns 7

len([ITT])

returns the length of the item text assuming it is a string

len([folder_name])

returns the length of the string or number in the folder_name

Related Rules

OFFICIAL LIST OF EccoExtDoc CONTRIBUTORS:

[Andrew Brown](#)

Albert Schepers

DaveG (who founded this Wiki!)

and special thanks to

Albert Schepers

lower(string)

Changes the string characters all to lower case

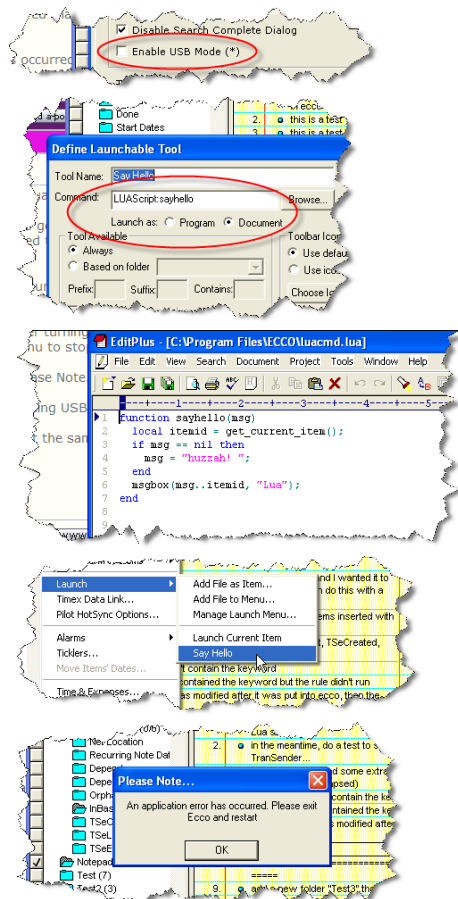
Examples

lower ('Hello World')

returns hello world

Related Rules

[upper\(\)](#)



match(string,pattern)

Description

returns that part of the string that matches the pattern
string can be any valid string or reference to a folder or item
the pattern can be any Regex pattern, see examples

Example

```
match("T 123456 234567","\s*\w\s+(\d+)\s+\d+\s*")
```

returns the string 123456

```
match(123.456,"(\d*)(?=\.)")
```

returns the integer part of the number 123

Related Rules

[imatch\(\)](#), [replace\(\)](#), [ireplace\(\)](#)

maxc(expression[, condition])

Description

returns the maximum for all children values if the condition is true
If no condition given then the condition is assumed to be true.

Example

maxc(fv("Price"))

returns the maximum value in the price folder for all child items

Related Rules

[minc\(\)](#), [maxd\(\)](#), [mind\(\)](#), [maxp\(\)](#), [minp\(\)](#)

maxd(expression[, condition])

Description

similar to maxc() except it works on all depends items
returns the maximum for all depends values if condition is true
If no condition given then the condition is assumed to be true.

Example

maxd(fv("Price"))

returns the maximum value in the price folder for all depends items

Related Rules

[minc\(\)](#), [maxc\(\)](#), [mind\(\)](#), [maxp\(\)](#), [minp\(\)](#)

maxp(expression[, condition])

Description

similar to maxc() except it works on all depends items
returns the maximum for all parent values if condition is true
If no condition given then the condition is assumed to be true.

Example

maxp(fv("Price"))

returns the maximum value in the price folder for all parent items

Related Rules

[minc\(\)](#), [maxc\(\)](#), [mind\(\)](#), [maxd\(\)](#), [minp\(\)](#)

minc(expression[, condition])

Description

returns the minimum for all children values if condition is true
If no condition given then the condition is assumed to be true.

Example

minc(fv("Price"))

returns the minimum value in the price folder for all child items

Related Rules

[maxc\(\)](#), [maxd\(\)](#), [mind\(\)](#), [maxp\(\)](#), [minp\(\)](#)

mind(expression[, condition])

Description

just like minc() function, but it's use on depends items
returns the minimum for all depends values if condition is true
If no condition given then the condition is assumed to be true.

Example

mind(fv("Price"))

returns the minimum value in the price folder for all depends items

Related Rules

[minc\(\)](#), [maxc\(\)](#), [maxd\(\)](#), [maxp\(\)](#), [minp\(\)](#)

minp(expression[, condition])

Description

just like minc() function, but it's use on depends items
returns the minimum for all parent values if condition is true
If no condition given then the condition is assumed to be true.

Example

minp(fv("Price"))

returns the minimum value in the price folder for all parent items

Related Rules

[minc\(\)](#), [maxc\(\)](#), [mind\(\)](#), [maxd\(\)](#), [maxp\(\)](#)

msgbox(message, box_title)

Description

Displays a message box with the message being the text to be displayed and box_title the string used as the message box title. This is useful for debugging LUA code to be used in eccoext when using the expression evaluation tool.

Only a single line of text can be displayed at a time.

The message box is not sizeable however the box size will be adjusted by the length of the message. It is unaffected by the length of the title.

Examples

msgbox("one","Box Title")

will display the word one in the message box having a title Box Title

for i=1,max msgbox(x[i],"Item Text")

this will display values of x from 1 to max in a box titled Item Text

now()

Description

returns the current date and time
can be used with other date functions

Example

now()

returns yyymmddhhmm

now()+5

returns the date and time five days from the current date

right(now,4)

returns the time only hhmm extracting the right most 4 characters

Related Rules

[today\(\)](#)

OK, let's get going, its' really easy!

A) If you have Ecco already installed, go to B), otherwise start here.

BTW: This (= section A) is my very first contribution ever to a wiki ever on this planet. Exciting. :-) ws

Section A Installing ECCO

(tested under Windows XP Home edition, service pack 2)

- First you need to install the original ECCO installation package. It is available in the file section of the [Ecco Pro Forum on Yahoo!](#).
 - *You need an Yahoo Groups account for that and also register for the Ecco group (may take it's time).*
 - *In the file section of the Ecco Pro Forum look for this: [**Ecco Pro v. 4.01**](#) (dated March 23, 2007), click on it.*
 - *You will see two links:*
 - [Ecco Pro from Netmanage site - Setup32.sfx.part1.exe](#)
 - [Ecco Pro from Netmanage site - Setup32.sfx.part2.rar](#)
 - *Download both files to your computer*
 - *Start the ...part1.exe. This will create a new file: "Ecco Pro from Netmanage site - Setup32.exe", which seems to be 100% identical to the Setup32.exe mentioned above.*

- Now you are ready for installation. Except if you intend to sync your ecco files with palm device:
 - Then you will need to run the palm software before you install ecco.
 - To install just follow the instructions for your palm device but note that there are some changes you will need to make to ensure that the sync process will work properly after ecco is installed.
 - It is recommended that you follow some of the links and discussions in the discussion groups for more detail on installing palm software.

- Install Ecco: Start the setup file (setup32.exe).
 - If you get an error "Corrupt installation detected" make sure your windows account has admin rights, If you want to use ECCO from a Windows account with only guest rights, I think, it should be ok, if you just change this account temporarily to admin rights, and after installation, you change this back to a guest account.
 - If you're running x64 OS, you'll need the 'pre-install' version. You'll find that at [EccoTools.com](#) . Find the 'Single directory' install, and copy it to program directory in the 32 bit program special directory in your x64 system. Take a look around the EccoTools board, and you'll find all sorts of helpful tools and add-ons.

- First it tells you "searching for installed components", this may take some minutes. After that at my (wsws) system the process froze. This solution (which I found via google) worked for me, but please: take care! No guarantee!: closing as much small applications in the tray, and in the Taskmanager looking for "ntvdm.exe" and close down this process (but without guarantee, and I not really know what this means for the rest of your system. do it on your own risk!).
- Next problem might be an error message "Mfc42.dll is in use...." You might just ignore this, or you may try what worked at my system: I stopped the installation process and rebooted windows in safe modus (including network things), and within this safe modus again started the installation. This worked fine.
- At the end there was one more message: ""Installation has been unable to detect a valid installation of the U.S. Robotics Pilot Desktop software. Please install the U.S. Robotics Pilot Software prior to installing the ECCO conduits." This may be safely ignored unless you are intending to sync with a Palm device and if you are and have already installed the US robotics software this message will not appear.

Now your naked ECCO should be working and you should be able to go on with the next steps explained below in section B)

Section B Using eccoext

First, download the latest version of eccoext from the Yahoo! Groups [Ecco_Pro forum](#). (While you are there, also download the replacement default.ect file which should replace your existing default.ect file in the ecco32.exe program directory. The new file fixes a latent corruption in the structure of the original template, and fixing it up-front will keep you smiling as you use EccoPro and can save many a-tear-drop down the road.)

Unpack the .rar file using your favorite compression tool. The latest version of WinZip works well, but there are also [free utilities out there](#), including a nice (and Free) one in the [Ecco_Pro forum](#) Files section.

Inside the .rar file is an Ecco file called eccoext.eco It contains release information and everything you *really* need to get started. For what it's worth, I'm expecting that you are already an Ecco "power user", if not, you should go spend some time getting up-close-and-personal with Ecco. Then opening and reading the provided file will make a lot more sense. **(A more 'user friendly' file with the rules and explanations is available in the Ecco_Pro file section).**

OK, since you are probably in too much of a hurry to wade through the .eco files, here's a beta for you...

FIRST!!! back up your data! This is "bleeding edge stuff!" Work with a test file until you feel comfortable that everything is safe. It is **TOTALLY YOUR** responsibility to care for your own data! If there is a bug in the code that translates all of your data into Swahili, it's **your** problem, not any-one else's!!! Back up now!!! (FWIW, I ran it in a Virtual PC until I was comfortable that it was predictable and that I wouldn't @#\$% things up...)

- Make sure you are using Windows-2000, Windows-XP, or Windows Server 2003. Windows 95, 98, ME, etc. are not supported by the extension and Vista might work but isn't tested...

- Extract all of the files in the .rar to your Ecco install folder.
 - For me that was C:\Program Files\ECCO
 - (I just moved the eccoext.exe and eccoext.ini there... if you are reinstalling eccoExt.exe, you might want to keep your old eccoext.ini if you have made any customizations...)

- Close Ecco. (since you are reading this in a browser that should be OK... if you were reading the instructions in Ecco things would now become a little dicy...) You should even check that there isn't an "ecco32.exe" process in task manager... (or reboot if that doesn't make sense...)

- Double click on eccoext.exe, or drag and drop a shortcut to your desktop (Ctrl-Shift drag drop) and double click the shortcut.

- An Ecco icon will appear down in your "system tray" (that's where the clock is in the lower right corner of most Windoze screens...)

- Click the icon and use the menu to open Ecco.

- Enjoy!

PLEASE PLEASE PLEASE update these instructions if you think you can clarify them. Please don't flame me (I'll cry, but I'll get over it...) just make the fixes you think are appropriate...

Make as many bug reports as you can!! (there is a prize for the most bugs found).. Report Extension bug reports at:

<http://EccoBug.com>

And if you have any burning questions, post them on the [ecco_pro forum at Yahoo!](#) or if you have more "deep-geek" technical questions, join the conversation at [EccoMagic...](#) also..

You can get & GIVE help in the Extension's help forum at <http://SlangHelp.com>

remove_depend_item(item_id, ditem_id)

Description

this function removes the dependents, the second item (ditem_id), from the first item (item_id).

Example

```
x=get\_select\_items\(\)
maxx=table.maxn(x)
for inc1=1,maxx do
for inc2=1,maxx do
if has_depend_item(x[inc1], x[inc3])==1 then
remove_depend_item(x[inc1], x[inc3])
msgbox("Item "..get_folder_value("ItemID",x[inc1]).." Depends
"..get_folder_value("ItemID",x[inc3]),"Removed depends reference")
end
end
end
```

This bit of code will check all selected items and determine if they have any dependencies and if there is it removes the references.

replace(string,pattern,replace_string[,original_string])

Description

Replaces the matched part of the string with the replace_string.

Last argument is logical, true or false, and if true returns the original string, if false returns an empty string.

Examples

replace(itt,"Rules","Expressions",true)

Returns the ITT except for the pattern matched "Rules" where the replacement string "Expressions" is returned.

replace(itt,"Rules","Expressions",false)

Returns a blank except for the pattern matched "Rules" where the replacement string "Expressions" is returned.

replace(ITT,"(w?)(\d{2})(\d{3})", "H:\20\$2\\$3",false)

Assuming the ITT contains "07328" the returned string is "h:\2007\328"

If the ITT contains any other value then nothing is returned, the last parameter is 'false', the default condition if left out.

If the last parameter is true the value of the ITT is returned.

replace([folder],"(?<=\d)\.d+", "",1)

The interger value of contents of folder "folder". (Actually, removes the period and ALL trailing numbers which follow another number.)

Related Rules

[ireplace\(\)](#)

Right(string,number)

Description

Returns the right most characters from the string where <number> represents the number of characters to extract

Example

Right("Hello World", 5)

returns World, the five right characters in teh string

Right(ITT,10)

returns the ten right most characters of the item text

Related Rules

[left\(\)](#), [substr\(\)](#)

Sample Lua Programs

Primer

It is highly recommended that the lua programming manuals be studied to understand the syntax and structure of lua programs. To assist, the following are some pointers based on a beginner's experience in programming in lua for eccoext:

- there is no error trapping or reporting, if the code fails generally nothing happens and there is no feedback given as to the reason for the failure;
- a typical error can be the omission of the "then" as part of the "if" statement or similar typographical errors;
- take care to distinguish between the lua code contained in rules and programs. The program uses the lua 5.1 code whereas the rules use the slang interpretation of lua;
- some of these samples are written as eccoext rules and others as functions. The only difference is that the functions start with a name and end in a corresponding 'end'. To change the scripts simply add the function name and the closing end statement: it is a common mistake to leave off the closing end statement.

Most programs may be tested using the eccoext evaluator prior to implementing it into a function. The evaluator is very useful for debugging of sections of code to activate in eccoext use CTRL ALT K.

Page Explanation

This page is dedicated to eccoext Lua programming examples developed by users. Each sample program links to a new page where the details of the program are presented. The last sample is empty for the benefit of anyone who wishes to add to the samples, simply click on it to take you to a fresh page and then edit the new page. To edit a page you will need to be registered with this site and will have to be logged on. And please, if you do add to the samples put a link to the next sample program empty page for the next person.

[Sample Program 1](#)

A program to extract the item text from selected records in the ecco file and save to a CSV file.

[Sample Program 2](#)

A program to count the words and characters of selected items

[Sample Program 3](#)

A sample program to extract data from folders and concatenate the information to the item text and then

write the result back to the item text.

[Sample Program 4](#)

A script that will automatically assign multiple dependancies when more than one item is selected.

[Sample Program 5](#)

Extraction of depends items and displaying of the depends references.

[Sample Program 6](#)

Awaiting Creation.

Sample Program 1

Description

This bit of code was developed for the purpose of testing the input/output capabilities of eccoext LUA programming. Once as a CSV file the collected data can be imported to other files for other purposes. The intent is to extract job costing information which could then be used for creating invoices and time records for payroll.

Program Code

```
1. function DataExtract()
2.   NumberItems=get_select_items()
3.   MaxItems=table.maxn(NumberItems)
4.   msgbox(MaxItems .. " items selected", "")
5.   if MaxItems >0 then
6.     filename="c:\\test.txt"
7.     io.output(filename, "w+")
8.     for i=1,MaxItems do
9.       y=tostring(i)..", "..get_item_text(NumberItems[i]).."\n"
10.      msgbox(y, "")
11.      io.write(y)
12.    end
13.    io.close()
14.  else
15.    msgbox("no selected items", "")
16.  end
17. end
```

Code Lines Description

1. Name of the function mandatory for all functions
2. Sets a variable **NumberItems** to the total number of selected items. Please note that the intent is to select all items in a notepad after they have been filtered and sorted in this way there is no need to do the filtering after the fact. It is possible though to import the data into a data base and then do further manipulation to suit specific needs.
3. This is a LUA programming function that sets **MaxItems** to the size of the vector created in line 1.

4. This is a redundant piece of code but useful in testing as it displays **MaxItems**.
5. This conditional statement tests to see if the **MaxItems** is greater than zero. If it is then data is extracted, if not then a warning message is displayed and the program does nothing.
6. A filename is set, in this case it is set to test.txt in the root of drive c. Any name or location will work. In this case, for testing it was a simple name and convenient location.
7. Another LUA programming function that opens a file for output using the filename in line 5 and the "w+" indicating to clear the file before writing to it.
8. A for loop from one to **MaxItems**
9. Set a string value based on the data contained in the selected items and store as variable **y**.
10. Another redundant message box but again useful in testing and seeing that something is happening.
11. The LUA programming function that writes the data to the file opened for output.
12. The end of the for loop after all of the selected items have been worked upon with the appropriate data extracted and stored in the CSV file.
13. The LUA programming function that closes the file.
14. This is the else statement from the if **MsgBox** > 0.
15. The message that there were items selected.
16. The end of the iff statement
17. The end of the function

References

See the [LUA programming Manual](#) and [ecomagic](#) forum for more details.

Sample Program 2

Description

This program uses two techniques to count the number of words in the selected itme text. The first uses an algrithm based on the number of characters and then dividing by 6 the second uses regular expressions to count the number of words by ... counting the number of words (excluding numbers). This later is a unique way of solving the problem in a single line of code. The first variation is a detail of counting the words in one item of text and the variation on the theme will count the text for any selection of items returning the individual itme counts and the total count.

First Variation

```
--  
-- Ecco LUA extension  
-- copy into luacmd.lua in the ecco directory  
--  
-- assign to the tool menu as a document  
--  
-- by Larry Yudelson, ads@yudel.com November 2007  
--  
--  
--  
function wordCount()  
fullText=" "  
NumberItems=get_select_items()  
MaxItems=table.maxn(NumberItems)  
  
msgbox(MaxItems .. " items selected", "")  
for i=1,MaxItems do  
  
-- y=tostring(i)..", "..get_item_text(NumberItems[i]).. "\n"  
  
thisText = get_item_text(NumberItems[i])  
fullText= thisText.." " ..fullText  
end -- next i  
  
len=string.len(fullText)  
  
-- count it with string.len(variable)  
  
wc=len/6  
msgbox("item has "..len.." chars and "..wc.." words!", "")
```

-- let's initialize the variables for this string

wc=0 -- start with no words

-- looks like we could do it with a REGEX

```
for w in string.gmatch(fullText, "%a+") do
```

```
wc=wc+1
```

```
end -- next w
```

```
msgbox(wc.." words found","ECCO WC")
```

```
end
```

--- end wordCount

Variation on a theme

This variation derives from [Sample Rule 6](#) but extend it to sum the word count over a series of items including subitems. This code only displays the values but it is possible to either store the value in a folder or in an external file. To store the value in a folder use the `set_folder_value()` function.

Alternatively it is possible to create a new line item or subitem in which case the `set_item_text()` and `create_sub_item()` would be useful.

```
function WordCountExt()
```

```
NumberItems=get_select_items()
```

```
MaxItems=table.maxn(NumberItems)
```

```
msgbox(MaxItems .. " items selected","")
```

```
tc=0.
```

```
for i=1,MaxItems do
```

```
wc=0.
```

```
for w in string.gmatch(get_item_text(NumberItems[i]),"%a+") do
```

```
wc=wc+1
```

```
end
```

```
msgbox(wc.." item "..tostring(i).. " word count","")
```

```
tc=tc+wc
```

```
end
```

```
msgbox(tc.." total Word Count","")
```

```
end
```

[\[Back to sample LUA programs\]](#)

Sample Program 3

Description

This piece of code extracts the folder values for each selected item and then concatenates the folder value to the item text based on certain rules. It is highly recommended that this code be tested on an unimportant data file as the information will be written to the data file once you execute it cannot be undone.

Code

```
1. id=get_select_items()
2. max=table.maxn(id)
3. for i=1,max do
4. project=get_folder_value("Project",id[i])
5. text=get_item_text(id[i])
6. date=get_folder_value("Appointments",id[i])
7. msgbox("Project "..project.."\\n".. "Item text "..text.."\\n".. "Date = "..date,"")
8. if project~="" and string.sub(text,1,1)~="#" and date~="" then
9. set_item_text(id[i],"#" ..project.." "..text)
10. end
11. if project == "" and date~="" then
12. msgbox("Item without project numbers", "")
13. end
14. end
```

Code Explanation

1. the item ids for all the selected items are collected in a table "id"
2. the length of the table "id" is determined and saved as "max"
3. for loop to loop through each of the selected items
4. store the [Project] folder value in "project"
5. store the ITT value in "text"
6. store the [Appointments] folder value in date
7. a message box that displays the stored values
8. the first conditional that checks to see if project and date have value as well whether the initial character of the text equals "#"
9. set the ITT with a new value based on the string "#"+project+text
10. end of the first conditional
11. This second conditional is a check on whether there the Project folder has no value and the

appointment has value

12. a message box displays the message that an item has no project number

13. end of second conditional

14. end of for loop

[Edit This Page](#)

Sample Program 4

Description

This piece of code extracts the item numbers from selected text then assigns dependencies. The last item selected will depend on all of the prior items selected.

Code

```
1. function dependsMany()  
2. NumberItems=get_select_items()  
3. MaxNumber=table.maxn(NumberItems)  
4. if MaxNumber <= 1 then  
5. msgbox("Need more than one item selected","Error 001")  
6. else  
7. for inc1 = 1,MaxNumber-1 do  
8. add_depend_item(NumberItems[MaxNumber],NumberItems[inc1])  
9. end  
10. end  
11. end
```

Code Explanation

1. Function name
2. the item ids for all the selected items are collected in a table "NumberItems"
3. the length of the table "id" is determined and saved as "MaxNumber"
4. If statement to verify that more than one item is selected
5. error message if improper selection
6. else statement for if
7. loop from 1 through one less than the maximum number of selected items
8. add the dependencies to the last selected item
9. end of for loop
10. end of if statement
11. end of function

Sample Program 5

Description

This piece of code extracts the item dependents and displays same.

Code

```
1. x=get_select_items()
2. maxx=table.maxn(x)
3. msgbox("Items Selected"..maxx,"X")
4. for inc1=1,maxx do
5. y=get_depend_items(x[inc1])
6. maxy=table.maxn(y)
7. msgbox(maxy,"Y")
8. if maxy>0 then
9. message="Item"..get_folder_value("ItemID",x[inc1]).." Depends "
10. for inc2=1,maxy do
11. message=message.. get_folder_value("Depends",x[inc1])
12. if inc2<maxy then
13. message=message.." "
14. end
15. end
16. msgbox(message,"Depends Item")
17. for inc3=1,maxx do
18. if has_depend_item(x[inc1], x[inc3])==1 then
19. msgbox("Item"..get_folder_value("ItemID",x[inc1]).." Depends
   "..get_folder_value("ItemID",x[inc3]),"Confirm depends reference")
20. end
21. end
22. end
23. end
```

Code Explanation

1. get the id numbers for the selected items
2. determine the maximum number of items selected
3. message box to display the number of items selected
4. do loop to cycle through each item selected
5. get the id for all dependent items for the selected items

6. determine the maximum number of dependent items
7. message box to display the number of dependent items for each of the selected items
8. if there are more than one dependents then create a message
9. set up the start of the message getting the ItemID folder value
10. For loop to cycle through each of the dependent items
11. concatenate the ItemID folder value to the start of the message
12. if there are more depends then
13. concatenate a comma to the end of the message prior to looping back
14. end of if statement
15. end of for loop
16. message box to display the built up message
17. for loop to increment through the selected items to determine if the selected items are dependant on any of the other selected items
18. if conditional to verify that the specific item has depends items
19. message box to display the item dependencies based on the ItemID folder
20. end of if statement
21. end of for loop
22. end of if statement
23. end of for loop

Notes

This bit of code displays the ItemID values that are created when the dependencies are set up (get_folder_value()). Item_id and ItemID are different and should not be confused when working with LUA scripts and eccoext dependents.

Sample Rule 1

This page explores a simple rule to concatenate information taken from the item text to be placed in the [Net Location] folder. The value in the Net Location folder is then used to automatically open a folder on a server. There are two variations of the rule to explore the subtle differences and possibilities. First is a rule that makes use of conditionals within the value and the second places the conditionals at the end of the statement.

The item text (ITT) in this case is very specific consisting of five numbers and a possible leading 'W' taking the form, for example, 07289 or W07301 where the first two characters represent the year (07) and the remaining three a project number, the leading W indicates that the project is an inspection project whereas without the W it is an engineering project.

The first rule developed took this form:

```
++:I!+-:iff([Engineering Projects],""H:\20"+match(ITT,'b(d\d)d\d\d\b')+"" +  
match(ITT,'b\d\d(\d\d\d)\b')+""",iff([Inspection Projects],""I:\20" +  
match(ITT,'b\w(d\d)d\d\d\b')+"" + match(ITT,'b\w\d\d(\d\d\d)\b')+""","")):len(ITT)=5 OR  
len(ITT)=6
```

where;

++: required at the start of all rules

I!+-:

the flags that tell the rule when it is to work or not work (see Flags Values Conditions for details) the flag basically triggers when the text is initial created and anytime that it changes.

iff(the start of the iff() statement (see rules for details) which is in the form iffcondition,true,false)

[Engineering Projects], The condition statement which in this case is true if the ITT is in the [Engineering Projects] folder where [Engineering Projects] is a check mark folder therefore if checked it is true.

""H:\20"+ this is a fixed string which sets up the initial value to drive H and the start of the year 20xx

"match the match function which extracts information from

	a string match(string,pattern)
(ITT,'\\b(\\d\\d)\\d\\d\\d\\b')	the item text first two numbers are extracted from a total of five (see patterns for more information) in this case the first two numbers represent the year
 \"+\"\\\"+\"	this sets up the slash required to indicate a subdirectory the balance of the rule is just extraction of different parts of the ITT and concatenating it to the start
 len(ITT)=5 OR len(ITT)=6	two conditionals to force the value to be calculated and inserted only if the length (len(string)) of the ITT is either 5 or 6.

The final result is "H:\2007\289" or "I:\2007\301", based on the examples above. It should be noted that the double quote marks are not necessary but are inserted as if the [Net Location] value has spaces in it, unless it is captured between double quotes, it will not work.

The rule was changed after examination and study to be simpler and more readable. The change is based on the fact that more than one rule may be applied at any time. In this case the rule was split into two rules as follows:

```
++:MTI!+:\"H:\20\"+match(ITT,'\\b(\\d\\d)\\d\\d\\d\\b')+\"\\\" +  
match(ITT,'\\b\\d\\d(\\d\\d\\d)\\b')+\"\":[Engineering Projects]
```

```
++:MTI!+:\"I:\20\"+match(ITT,'\\b\\w(\\d\\d)\\d\\d\\d\\b')+\"\\\" +  
match(ITT,'\\b\\w\\d\\d(\\d\\d\\d)\\b')+\"\":[Inspection Projects]
```

Note the difference in the flags MTI!+- to manually set the flags (M) and then to only work on the top level item (T) and then on first creation and when ever the ITT is changed. The concatenation remains the same as above but there are no imbedded conditionals, the conditional is simply the [Folder Name] at the end, after the last colon(:). This simplifies the rules making them work faster and are easier to read.

*[the expression can simple replaced by "iff([Engineering Projects],
replace("ITT","(W?)(\\d{2})(\\d{3})", "H:\20\$2\\\$3", "")"]
[by slangmgh]*

In keeping with this change the rules can read:

```
++:MT!+:ireplace(ITT,"(w?)(\\d{2})(\\d{3})", "H:\20$2\\$3"): [Engineering Projects]
```

++:MT!+:ireplace(ITT,"(w?)\d{2})\d{3}", "I:\20\$2\3"): [Inspection Projects]

Sample Rule 2

The Set Up

The Top Level Item (TLI) is:

- "Appointment Text"

and the first child item or sub item (SLI) is:

- "T 232411 232423"

or;

- "V 123 456"

where the "T" stands for truck; "V" stands for van; and the numbers represent odometer readings. There is nothing else in the line only the "T", or "V", and the two numbers.

Required

To extract the numbers and place them in a separate folder so that the mileage can be determined and other calculations performed for tax purposes.

Discussion

AT this point there are several possible solutions from extraction of the numbers and putting them into appropriate folders to doing calculations on them before putting placing a value in a folder. For this exercise all solutions that were presented in the Ecco_pro Yahoo group will be presented along with an explanation. As not all the information, or at least not all of the requirements were carefully read, was presented in the set up the final solution differs from the first solution. Following the flow of thought should give some insight into rule creation and possibilities.

It should be noted that for anyone who followed the discussion that slang hinted at the solution in the very first response, just not the final answer.

Solutions

For the purposes of this sample it will focus on the truck ("T") only, to make the final solution work for the van ("V") it will be simply a matter of substitution.

First Suggestion

This first suggestion is based on gathering the information from the item text using the string functions and extraction as follows:

- ++: !+: iff(Left(ITT,2)="T ",0.+substr(ITT,10,6)-substr(ITT,3,6),0):[Appointments]

where:

- ++: *required*
- !+ trigger you may alter depending on what you need
- iff() *conditional to test and evaluate based on condition*
- Left(ITT,2)="T " checks the left most character for T (Truck. Note the space after the T as any word that begins with T would trigger the rule but "T " is never used, unless we type poorly.
- 0.+substr(ITT,10,6)-substr(ITT,3,6) *true condition extracts substrings and subtracts. Note the 0. to force the results to be numeric*
- 0 the false condition in this case 0 though you could use "" to force a null value
- [Appointments] *conditional that forces the calculation only if the value is in the calendar. If you enter the value in another folder it will have no effect.*

The only change you need to make for the Van folder is to change the "T " to V ". One assumption is that the value is always input the same, that is, a Character a blank 6 numbers a blank and then 6 numbers. There may be other characters after the initial string for instance:

- T 123456 234567 travel to Texas and back

The "travel to Texas and back" would be ignored as the extraction is for specific sections of the text starting at 1 and ending at 15.

Second Suggestion

The first suggestion was based on the using string extraction, this second is based on pattern matching as follows:

- ++: !+: 0.+match(ITT, "^\\s*T\\s+\\d+\\s+(\\d+)\\s*\$")-match(ITT, "^\\s*T\\s+(\\d+)\\s+\\d+\\s*\$")

where:

- 0.+ forces the expression to be a number
- match() *will extract from a string a specific value*
- ITT the item text being matched

- `"^` *the start of the line*
- `\s*` any number of spaces from 0 to ...
- `T` *the letter T must be the first Character in the line*
- `\s+` any number of spaces from 1 to ...
- `\d+` *any number of digits from 1 to ...*
- `\s+` any number of spaces from 1 to ...
- `(\d+)` *the digits to be extracted therefore surrounded by ()*
- `\s*` any number of spaces from 0 to ...
- `$` *end of the line*

The one draw back to this method, the pattern must be precisely known and invariant. That is if the input were:

- T 123456 234567 test

It would fail as there is no provision in the pattern to accommodate the last word. If the pattern were to accommodate the word then when the input does have the ending text it it fail as well. It should be noted though that the number of spaces or the number of digits does not matter in this case. So:

- T 234 345

works just as well as;

- T 1234 123

As a variation the following contribution was made but could require a change in the data entry

Second Suggestion Variation

There are two regex pattern expressions in the above example as follows:

- `"^\s*T\s+\d+\s+(\d+)\s*$"`
- `"^\s*T\s+(\d+)\s+\d+\s*$"`

I think (not tested) the following should allow throwing away anything past the two sets of digits and (optional spaces)

- `"^\s*T\s+\d+\s+(\d+)\s*.*$"`
- `"^\s*T\s+(\d+)\s+\d+\s*.*$"`

where `.*` matches anything (possibly nothing) to end-of-line: `$`

PCRE regex allows the following construct (using named sub-patterns):

- `^\s*(?<VehicleType>T)\s+(?<StartMiles>\d+)\s+(?<EndMiles>\d+)\s*.*$`

and that this replacement instruction:

- Vehicle \$1 Start \$2 End \$3

changing "T 123456 234567 test" to "Vehicle T Start 123456 End 234567"

With this replaced text it would then be possible to parse the item text as follows:

- `0.+parse("^s*(?<VehicleType>T)\s+(?<StartMiles>\d+)\s+(?<EndMiles>\d+)\s*.*$")match(ITT,$3-$2)`

or (better):

- `0.+parse("^s*(?<VehicleType>T)\s+(?<StartMiles>\d+)\s+(?<EndMiles>\d+)\s*.*$")match(ITT,${EndMiles}-${StartMiles})`

For reference some of the above information was cribbed a page from RegexBuddy Help from:

<http://buralex.googlepages.com/ReplacementTextReferenceJGSoftRegexB.htm> which discusses how various implementations support replacement text references.

Other references:

<http://www.pcre.org/pcre.txt>

<http://lrexlib.luaforge.net/>

<http://wiki.tcl.tk/2255>

Third Suggestion

This suggestion builds are the information from the first two suggestions and provides a final solution that matches the requirements. This solution was as follows:

- `++:P!+-:match(TVC(),"^s*\T\s+(\d+)\s+\d+\s*$"):`

in the truck start folder; and

- `++:P!+-:match(TVC(),"^s*\T\s+\d+\s+(\d+)\s*$"):`

in the Truck end folder, where:

- `:P!+-:` this flag will only work on the parent item inserting text the first time, any time there is a change and deleting the value if the condition is not met
- `TVC()` //this forces the parent item to look at the first child item for the string

Final Thoughts


This has been an interesting exercise and the contributions from the group were invaluable. Now if we look at the first response by yoursowelcomethanks:

- "key rules is TVC([cond]) Get item text of the child with condition"

we see that the solution was right before us to begin with. In going through the exercise, the contributions were a great learning tool. Any one reading this is encouraged to follow the entire thread http://tech.groups.yahoo.com/group/ecco_pro/message/1817 for subtleties as this Sample can not begin to capture all of the development.

Contributors

In order of first contribution to the thread:

- Don Johnson
- yoursowelcomethanks
- Albert Schepers
- Dave Gustafson
- r634718
- -  [Alec Burgess](#) Sep 1, 2007 7:34 pm
- Slang

A useful Regular Expression rule

Make a folder called "Domain" and give it this rule. Then, for every item that has a URL associated with it in the "Net Location" folder, the "Domain" value will be the root of that URL, so that, for example, all stories clipped from a particular newspaper will have the domain "www.guardian.co.uk". The underlying URL will still be in the "Net Location" folder, so that double-clicking on the text will still take you there. But the Domain is often much more informative to human eyes.

The auto-assign rule for the "Domain" folder is this:

```
++: !+-: {1}: [Net Location]~~='http://([^\s/]+).*'
```

Reading left to right:

++: says "This is a slang rule"

!+- says "Run it after every item change"

{1} says "Set the value of this folder to whatever ends up between the first set of brackets in the regular expression that follows"

The Regular Expression itself is the last bit of the rule:

which extracts from an entry in the "Net Location" folder everything after the initial

`http://`

up to the next forward slash.

This is because `[^/]` means "Any character but a forward slash" and following it with a plus sign means "Any consecutive run of characters that are not forward slashes". Wrapping the whole lot in brackets means "Remember this bit for later use" - so that it can be copied into the domain name folder. Finally, the `.*` just means "anything at all up to the end of the line". In this instance it is thrown away; in other rules, it might be used.

Further, rather sillier example

We could change this rule so that it captured both the domain and the filename from the original URL.

We already know that the domain is everything between

`http://`

and the next forward slash.

Let's assume, then, that the filename is everything after the *last* forward slash in the URL.

So we modify the capturing regex

[Net Location]~~='http://([^/]+).*'

so that the end reads ([^/]+).+/(.+)

The new, bold part of the expression is everything between the first and the last forward slashes. It is written differently from the first part because we wanted the first regex to stop matching at the *first* slash it came to; we want this one to stop matching at the *last* slash. I hope the difference is clear. In this imaginary url:

http://thefirstbit/could/be/followed/by/many/slashes/before/theurl
([^/]+) matches **thefirstbit** and the following .+ matches **/could/be/followed/by/many/slashes/before/**
Finally, wrapping the end of the Regular expression in brackets makes it available just like the first part, the domain. Since the Domain was {1}, the filename will be {2}

So, we can rewrite the whole rule as ...

```
++:!--:'From a file called {2} found in {1}':[Net  
Location]~~='http://([^/]+).+/(.+)'
```

which would give a column which showed, rather verbosely, more detail about where a text snippet had come from.

Contributor

Taken from the Yahoo Groups Post "Very Simple Rule Example" by
Andrew Brown

A Concatenation Rule

Decription

This rule is used to take a value from two folders and combine them. The result is placed in the **[Net Location]** folder so that ECCO can directly access the file. The two folders are **[Folder]** and **[File Name]** where **[Folder]** is intended to contain the full directory location of the file and **[File Name]** is the name of the file in the directory. For instance *N1 ABE-SDC Meeting 1 draft agenda R3.doc* is the file name in the folder *"J:\regs_standards\Accessible Built Environment\General"* the quotes are required as there may be spaces in the string and ecco will not recognize the complete string without the quotes.

Rule Description

```
iff([File Name]== "", "", ""+substr([Folder],2,len([Folder])-2)+[File Name]+ ""):[Location]="Server"
```

This rule creates a string that starts and ends with the addition of a single quote so that the ecco will recognize the entire string when it launches the item. This is done by the ' ' ' to force the quote into the string.

```
iff([File Name]== "",  
/*an if statement that determines if [File Name] has value  
  
"" ,  
/*null is returned if true otherwise  
  
""+substr([Folder],  
/*the substring of the [Folder] is extracted, the quotes are stripped off  
  
2,len([Folder])-2)+  
/*the quotes are stripped off  
  
[File Name]+ "" )  
/*the [File Name] is added to the string  
  
:[Location]="Server"  
/*this rule only is applied if the [Location] folder value equals "Server"
```

Sample Rule 5

Description

A rule to automatically create an entry into a folder except if it has already been manually entered. This is actually two rules which take advantage of the + FLAG function.

Set Up

There are three folders [A], [B], and [C]. [A] is the folder that we want to enter data either automatically when the ITT is first created and the condition containing folder [B] is true. If the condition is false and the value of [A] is entered manually then the value of [A] should not change and hence the need for a third folder [C]. [A] is a date folder, [B] may be any type of folder and [C] is a check mark.

The Rules

In [C];

++:++:[A]

This rule forces a checkmark into [C] if [A] has value. If [A] is entered manually then [C] has value.

In [A];

++:today+2:[B] and ![C]

The value in [A] will be set to today + 2 if both [B] has value and [C] has no value. if [A] had been entered manually then [C] would have value and hence the condition would be false whether [B] takes on value at some later time or not. It should be noted that if the condition is true and the value of [A] is set then [C] will also be set.

The Contributors

Taken from the ecco_pro yahoo forum thread with the following contributors in order of contributions:

rbrandes1
yoursowelcomethanks
Albert Schepers

Sample Rule 6

Description

This rule will count the number of words entered in the item text updating it when you have completed the typing.

Rule

```
++:L: wc=0.  
for w in string.gmatch(get_item_text(item_id),"%a+") do  
wc=wc+1  
end  
set_folder_value("comments",item_id,wc)
```

Credit

based on the work of Larry Yudelson in [Sample Program 2](#)

Sample Rule 7

Description

This rule will extract the last three characters of a string and then use this information to insert a long name into a folder.

Rule

```
++:FI!+-:imatch("
pdf=Adobe,xls=Spread
Sheet,doc=Word,htm=HTML,wpd=WordPerfect,php=Program,com=HTML,ppt=Presentation,"
imatch([File Name], ".{3}\s*$")+ "(.+?)"):
[Location]="Web" or [Location]="Server"
```

Explanation

This rule does a match on a string Line 3 and replaces it with the long names contained in line 2.

Line 3 takes the contents of the folder [File Name] and extracts the last three characters ".{3}\s*\$", allowing for spaces after the characters, and then compares that to the three character strings on the left side of the equal sign in line 2, and then replaces it with long file name on the right side of the equal sign.

Example

[File Name] = "<http://tech.groups.yahoo.com>"
the rule would return "HTML"

Sample Rules

Page Explanation

This page is dedicated to eccoext rule examples developed by users. Each sample rule links to a new page where the details of the rule are presented. The last rule is empty for the benefit of anyone who wishes to add to the samples, simply click on it to take you to a fresh page and then edit the new page. To edit a page you will need to be registered with this site and will have to be logged on. And please, if you do add to the samples put a link to the next sample rule empty page for the next person.

[Sample Rule 1](#)

A rule to extract project information from the item text to create an entry in the [Net Location] folder. The entry is used to open a directory on a server where project information is stored.

[Sample Rule 2](#)

A review of the news group ecco_pro thread that extracts and separates information from a child item text and places the contents into new folders.

[Sample Rule 3](#)

A rule that extracts the domain name part of the web page or address.

[Sample Rule 4](#)

A simple concatenation Rule to join the values of two folders, with some string manipulation, to create a document location that can be launched when placed in the [Net Location] folder.

[Sample Rule 5](#)

A double rule taking advantage of the + FLAG to create a value automatically or, if it has been entered manually, then to leave it unchanged.

[Sample Rule 6](#)

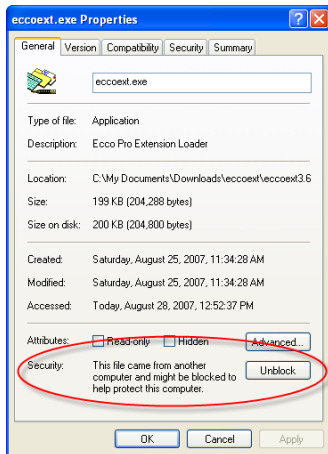
A simple Lua script rule to count the number of words in the ITT see also the [Sample Program 2](#) for the LUA program function.

[Sample Rule 7](#)

A rule to compare the file type, the last three characters in the file name, and to insert a long name for the file type into a folder: a dictionary type rule.

[Sample Rule 8](#)

Awaiting creating



set_folder_value(folder_name,item_id,value)

Description

Sets the value into the named folder (folder_name) associated with the specified item.

Example

```
set_folder_value("Release",item_id,"3.6.5.0.0")
```

Sets the value of the Release folder for the current item to 3.6.5.0.0.

```
wc=0.  
for word in string.gmatch(get_item_text(item_id),"%a+") do  
wc=wc+1  
-- msgbox(word,"")  
end  
count=tostring(wc)  
set_folder_value("C",item_id,count)
```

This routine counts the number of words in the item text of the current item.

wc is the word count and is initially set to zero and later incremented by 1 as the for loop steps through each word of the item text.

The for loop uses the string.gmatch() function to match each entire word in the string, in this case the [get_item_text\(\)](#) ecco lua command.

Related Rules

[get_folder_value\(\)](#)

[Rules](#)

set_item_text(item_id, text)

Description

this function will replace the item text with "text"

if you test this function using the evaluator do take care as you can not undo the replacement once it has been done.

Example

++:L:

A=get_item_text(item_id)

B=get_folder_value("Release",item_id)

text=A .. " "..B

set_item_text(item_id,text)

this rule will take the current item (item_id) text, store it in A then get the Release folder value and store it in B, then concatenate the values into text and finally change the item text with the new string consisting of the item text plus the value of the release folder.

Related Rules

[Rules](#)

'The Old List'

Words of caution from YSWT, on the "old" EccoPro yahoo list, which was taken over by the owner of the 'compusol' site:

1. The site is infected by email harvesting bots. This is a serious, and known issue. It can be easily fixed, but those who were able to gain control of the old list have reasons for keeping the situation as it is. Most likely because they want to represent the list as having thousands of members. It does not, but that number shows up as the membership number and the new owners don't want to have the true number show up. In any case, don't venture there without the hide email option!!
2. For whatever dynamics of the individuals involved the old forum is now filled with misinformation about ecco, and silently suppresses various topics and posts including about the eccoext and other modern developments. Notably, the new ownership of the 'old forum' deleted and banned the links to the "new" EccoPro list [Ecco_Pro](#) (the home of the eccoPro revitalization) [which may or may not have to do with the compusol site offer to 'sell' for \$10 access to long ago outdated and superseded now FREE software, or 'packages' available freely from the 'New" Ecco_Pro list.].
3. Without mentioning any names, note also that among the moderators at the 'old' list, is a plagiarist who tried to rip off some of my own work, going so far as to edit out the instructions, copyright notice, and attribution, and post the work with a false explanation of where it came from. Not sure if it's funny or sad, but the compusol site predominantly displays a 'quote' from this same person-- a quote plagiarized from it's original author, who, although fully known to the compusol owner (the quote was taken from the author's forum post), is not credited nor mentioned in any way.
4. This is not to say that the 'old' list is without *any* value. Just as one example: Pastor Don (in addition to being a founding moderator of the "new" list, volunteered also to help moderate also at the "old" list, where he can also be found.
5. Final note, many of the members at the 'old' list have been discovered to be imposters, and even the ID of some of the moderators is in doubt-- a supposed 'famous' member having bogus email, and not responding to simple direct information query to confirm id. Anyhow, beware!!

Just an observation that there is good information on the old list especially on how to get ECCO PRO to work and sync with palm devices. I have been active and inactive in a limited way on the old list for many years and have only recently become interested again (as a result of eccoext) but was discouraged when a recent posting was criticized and told not to refer persons to other sites (read the new list). There are many good people involved and I sincerely appreciate all those who provide their knowledge and expertise.

Albert's comment is well taken, and note: on the new forum File Section you can find the ['Complete' Archives of the 'old' forum](#), containing some 20,000+ posts of amazing stuff.

Original documentation, lifted from Eccoext.eco

(Todo: return cleaned version to eccoext distribution)

Enhanced auto assign rules

- The new auto assign rules support column values and simple Boolean expressions
- The new auto assign rules have the following format
 - ++:*flag:value:expr*
 - The first '++:' is the invariable prefix that introduces one of the new rules
 - The **flag** determines how and when the rule will be applied. It may be one or more of the following characters.
 - T the rule applies to TLI items
 - D the rule applies for dependency checks
 - P the rule applies for parent changes
 - C the rule applies for sub item changes
 - I the rule applies when an item's text or an item's level changes
 - F the rule applies when an item's folder changes
 - S the rule applies when file first Load for Date change
 - L the rule is a LUA script
 - when this flag is set, the '+-!' flag will not be checked
 - and the script follows right after the ':'
 - + the folder will be set if the expr is true and the folder value has not set
 - - the folder will be cleared if the expr is false
 - ! the folder will be set if the expr is true; if the folder already has a value, the new value will overwrite the old one
 - If the folder type is date, the folder will be set to the present time if the value is empty and expr is true, otherwise use the value
 - If the type of folder is not date, it will use value as set
 - If the +-! flag is not set, the default is + (the folder will be set if the expr is true and the folder has no existing value)
 - if the flag 'F' and flag 'T' are not set, the default is 'FI'
 - '**value**' is the folder value which will be set if the expr is true
 - for checkmark type folders, the value can be empty
 - for date type folders, an empty value means Today
 - for other types of folder, an empty value means remove folder
 - value have the same syntax as expr: see below.
 - '**expr**' uses the following syntax
 - expression tokens
 - () are used to group an expression
 - ITT means Item Text
 - PTT means Parent Item Text

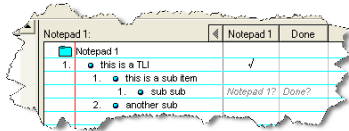
- ITP means Item Type
- SBF means if there is child or sub child folder set
- [folder name] means the folder value of the item
 - if the folder name has prefix '@', that means the dependent items' folder values.
If the folder name has the prefix '^', that means the parent item's folder value.
Otherwise, it refers to the current item folder value
 - for example value [Done] means use Done folder value as folder value
 - and value [@Done] means the Done folder values of the dependent items
 - and value [^Done] means the Done folder value of the parent item
- { } means the first Regex subexp
 - value {0} means use the whole match string
 - value {1} means the first subexp string
- = equals
- == equals
- != does not equal
- <> does not equal
- ! not
- > greater than
- >= greater or equal to
- < less than
- <= less than or equal to
- ?= equal ignore case
- ~= Matches a string against a regular expression
- ~~= Matches a string against a regular expression, ignoring case
- + add two numbers or strings, or add integer to date
- - subtract two numbers or dates, or sub interger from date
 - multiply two numbers
- / divide two numbers
- NOW The current date and time
- TODAY The current date
- && and
- || or
- AND and
- OR or
- NOT not
- TRUE true
- FALSE false
- number any number
- 'string' string
- "string" string
- expression functions
 - len(str) length of string
 - match(str, pat) return the matched part of a string: 'pat' is a regular expression
 - imatch(str, pat) like match, but ignore case
 - replace(str, pat, rep) replace all the parts of 'str' that match the regex 'pat' with 'rep'
 - ireplace(str, pat, rep) like replace, but ignoring case

- iff(cond, true_value, false_value) if cond is true, return true_value else return false_value
- expression examples
 - Test whether an item's text includes the string 'hello'
 - ITT ~= 'hello'
 - Test whether an item's text includes 'hello' and that it has a value in the folder Done.
 - ITT ~= 'hello && <Done>
 - Test whether an item's type is text and it's value in the Done folder is '20070101'
 - ITP==1 && <Done>=='20070101'
- expr supports regex expressions for the operators '~=' and '~~='. Regexes are also supported by match, imatch, replace, ireplace function
 - The regex language supports the following special characters:
 - \ Quote the next metacharacter
 - ^ Match the beginning of the string
 - . Match any character
 - \$ Match the end of the string
 - | Alternation
 - () Grouping (creates a capture)
 - (?:) Non capture group
 - [] Character class
 - Greedy closure (match the longest possible string)

** Match 0 or more times

- + Match 1 or more times
- ? Match 1 or 0 times
- {n} Match exactly n times
- {n,} Match at least n times
- {n,m} Match at least n but not more than m times
- Non-Greedy closure (match the shortest possible string)
 - Add "?" after the greedy closure
- Escape character
 - \t tab (HT, TAB)
 - \n newline (LF, NL)
 - \r return (CR)
 - \f form feed (FF)
- Predefined classes
 - \w alphanumeric [0-9a-zA-Z]
 - \W non alphanumeric
 - \s space
 - \S non space
 - \d digits
 - \D non digits
 - \b word boundary
 - \B non word boundary
- New auto assign rule example

- Timestamp the TLI item with its creation time (Date folder)
 - ++:T+::
- Timestamp the TLI item modification time (Only on Date folder)
 - ++:T!::
- Check a checkmark folder if any of its subfolders contain any item
 - ++:+::SBF
- Check a checkmark folder if any of its subfolders contain any item; if none do, uncheck it.
 - ++:+-::SBF
- If an item contains the text 'hello' check the folder; if not, uncheck the folder
 - ++:+-::ITT~~='hello'
- Set folder value to <Done> value if the item text contains the string 'hello' (This can be used as conditional Merge Column Value)
 - ++:+:[Done]:[Done] && ITT ~~= 'hello'
- Examples of assignment rules using Regular Expressions:
 - Find a web address to place in [Net Location]
 - ++:!:-'http://'+{0}:ITT~~='www\.* \.(comlnetledu)'
 - Find a mail address to place in <E-Mail>
 - ++:!::{0}:ITT~='\w+@\w+\.\S+'
 - Find the post message number of the form 'Message #1111'
 - ++:!::{1}:ITT~='Message #(\d+)'
- When the item text changes or folder value changes or is set or cleared, the rule will be checked automatically, but when you change the rule, you need to reapply the rule. There are two ways to do this.
 - To check the new auto assign rule for one Folder over all item
 - Open the folder window
 - Select the folder you want to operate on
 - Click the right mouse button
 - Select the menu item <Check Folder Rules>
 - To check the auto assign rule for all Folders over all items
 - Open the folder window
 - Click the folder button to see the folder menu or
 - Click the right mouse button on any folder
 - Select the menu item <Check All Folder rules>



Even though sub items are not in the folder, they will show in the notepad unless they are hidden with a filter.

substr(string, start [, length])

Description

Extracts a portion of the string. The string may reference text values in any folder or the string may be a fixed value.

Start sets the starting position for the extraction with 1 being the first position.

Length is optional. If left off then the extraction is from the starting position to the end and if the length exceeds the end of the string then extraction is from the start to the end of the string.

Examples

substr("Hello World",1)

returns Hello World

substr("Hello World",1,5)

returns Hello

substr("Hello World",7,5)

returns World

substr(ITT,1,2)

returns the first two characters of the item text

Related Rules

[right\(\)](#), [left\(\)](#)

sumc(value[, condition])

Description

for sum all children values if condition is true

If no condition given then condition is assumed to be true.

Example

++:C!-:sumc(fv("times") * fv("Price"), fv("Price") > 1):countc(fv("Price")) > 0

the flag for this example indicates that it will only act when a child level item changes (C!-)

the value is the sum of the product of the "times" folder and the "Price" folder (sumc(fv("times") * fv("Price"), fv("Price"))).

The contained condition is that the value of the "Price" folder must be greater than 1 (fv("Price") > 1)

the ending condition is that the count of the child folders is greater than 0 (countc(fv("Price")) > 0)

Related Rules

[sumd\(\)](#)

sumd(value [, condition])

Description

just like sumc() function, but used on the depends items
for sum all depends values if condition is true
If no condition given then the condition is assumed to be true.

Example

++:D!-:sumd(fv("times") * fv("Price"), fv("Price") > 1):countc(fv("Price")) > 0

the flag for this example indicates that it will only act when a dependslevel item changes (D!-)
the value is the sum of the product of the "times" folder and the "Price" folder (sumd(fv("times") *
fv("Price"), fv("Price")>2).

The contained condition is that the value of the "Price" folder must be greater than 2 (fv("Price") > 2)
the ending condition is that the count of the depends "Price" folders with a value greater than 2, is greater
than 0 (countd(fv("Price")>2) > 0).

Related Rules

[sumc\(\)](#)

Some of these links are also in the "Navigation Bar" on the left side of the screen.

This is a more detailed list, the index should have almost all of the pages.

[Introduction](#)

[What is Ecco ?](#)

[Quick Start](#)

[Features](#) (short course)

[Features](#) (in more detail)

[Rules](#) (Details and examples of many possibilities.) Don't Miss This!!

[Regular Expressions](#) (short course)

[Tutorials](#)

[Valuable Resources](#)

[Definitions](#)

[Index](#)

today()

Description

returns the current date
can be used with other date functions

Example

today()
returns yyymmdd

today()+5
returns the date five days from the current date

Related Rules

[now\(\)](#)

trim(string)

Description

Function removes the trailing and leading blanks from a string.

Example

Trim(" hello ")

Returns "hello"

Related Rules

[left\(\)](#), [lower\(\)](#), [right\(\)](#), [substr\(\)](#), [upper\(\)](#)

Tutorials

Here are some tutorials to get you started with using some of the eccoext features. Hopefully someone will soon contribute an "eccoext for dummies".

- [Using Dependencies](#)
- [USB Mode](#)
- [A guide to autoassignment](#)
- [Sample Rules](#)
- [Sample LUA Programs](#)

TV()

Description

gets the item text

there are no conditions for this rule

generaly used only within other rules

Example

`getpv(tv(), il() == 0)`

retuns the top level item (IL()==0) text

Related Rules

[FV\(\)](#), [FVC\(\)](#), [FVD\(\)](#), [TVC\(\)](#), [TVD\(\)](#)

TVC([condition])

Description

Get item text of the first child item with condition

Examples

TVC()

Always returns the item text of the first child item.

The lack of a condition implies always true.

TVC(left(getpv(tv()), il)= 0,1)~')

will only return the item text of the first child item if the first character of top level item TLI text is ~

Related Rules

[FV\(\)](#), [FVD\(\)](#), [FVC\(\)](#), [TV\(\)](#), [TVD\(\)](#)

TVD([condition])

Description

Get item text of the first depend item if the condition is true.

Please see [using dependencies](#) on setting and use of depends folder.

Examples

TVD()

always returns the item text of the depends item

the lack of a condition implies always true

TVD(left(ITT,1)='~')

will only return the item text of the depends item if the first character of the item text is ~

Related Rules

[FV\(\)](#), [FVD\(\)](#), [FVC\(\)](#), [TV\(\)](#), [TVC\(\)](#)

upper(string)

Description

Changes the string characters all to upper case. String may be any alpha characters or an expression that extracts text from a folder.

Examples

upper('hello')
returns the string HELLO.

upper(ITT)
returns the contents of the item text in upper case.

Related Rules

[lower\(\)](#)

The contents of this page are accurate as of 2007-Aug-15, eccoext version 3.6.1.7

The main focus of this page is in outlining how USB mode works from a power user perspective, and what are the gotchas and bugs. Since eccoext is actively being developed, the gotchas and bugs may be resolved soon, in which case an attempt to record the solution on this page will be made.

USB ECCO INSTALLATION STEPS

Follow the set of instructions at <http://www.eccomagic.com/forum/YaBB.pl?num=1186357673>

[see also [CLEANUP HOST SYSTEM](#) below]

Enabling USB mode

USB mode is enabled when USBmode=1 is included in file eccoext.ini and eccoext was *not* started with command-line option -l.

Usb mode is disabled when USBmode=0 is included in file eccoext.ini or eccoext was started with command-line option -l

USB ECCO - HOW IT WORKS

[main source:

http://tech.groups.yahoo.com/group/ecco_pro/message/1814%5D/\http://tech.groups.yahoo.com/group/ecco_pro/message/1814//

To clarify what USBmode means for eccoext, it's the ability to transfer your Ecco program folder and settings from your desktop PC - the source - to a USB key to run ecco on another computer -the target or host - where ecco isn't installed.

USB mode is NOT a 2-way synchronization between your USB key and your desktop ecco folders; once you run ecco in USB mode via eccoext, your desktop ecco settings will diverge from the settings on the USB key - if you keep using both your desktop ecco folder and your USB key ecco folder. In practice, such divergence, may not matter much to you, if you're not in the habit of customizing ecco. As for ecco data files, you can synchronize them with your desktop PC using your favorite synchronization

tool.

USB mode is NOT a method to run stealth ecco on a host computer without leaving any traces; USB mode does leave traces in the file system and registry of the host computer. Hence USB mode can be said to "not play nice" to a host computer where ecco is already installed.

[Note: the eccoext author is actively engaged in eliminating this problem from future versions.]

USB mode works by storing in the ecco program folder the registry and configuration settings that ecco would normally place in the system registry and in c:\windows.

Let's take a look at how USB mode interacts with the host file system and registry.

HOST FILE SYSTEM

When eccoext is first run in USB mode on the source PC, it copies the configuration files in c:\windows to the ecco program folder. From then on, eccoext ignores the files in c:\windows completely, whether they still exist or not. It relies entirely on its own copies of those files. [see also [FILES CREATED BY ECCOEXT](#)]

One exception is c:\windows\win.ini which ecco modifies (once). Eccoext ignores this point. So, because of this, USB mode can be said to leave a "trace" of ecco on the host computer. Other than that, I didn't find any other traces left on the host files system.

HOST REGISTRY

The host registry is entirely another matter. When eccoext is first run in USB mode on the source PC, it copies the system registry ecco keys into a set of .ini files in the ecco program folder. Then when eccoext is started on a host computer, it creates two files, ecco_save.ini (temporary) and ecco.ini. I would think that ecco_save.ini holds the ecco registry keys of the host computer. Then eccoext overwrites the host registry with its own keys from ecco.ini. When eccoext exits, it currently seems to leave those keys in the host registry. It doesn't restore the host registry ecco keys from ecco_save.ini.

[see bug report thread

http://tech.groups.yahoo.com/group/ecco_pro/message/1827%5D//http://tech.groups.yahoo.com/group/ecco_pro/message/1827//

Until this problem is corrected, be wary that you may end up overwriting ecco settings in the host computer - if ecco is already installed on it. And be wary that you're leaving your ecco registry keys stored on the host computer.

FILES CREATED BY ECCOEXT

information current as of version 3.6.1.7

[main source:

http://tech.groups.yahoo.com/group/ecco_pro/message/1818%5D//http://tech.groups.yahoo.com/group/ecco_pro/message/1818//

When USB mode is enabled, ecco creates the following files in the ecco directory: ecco.fdb, ecco.cfx, ecco.alm, ecco_save.ini, ecco.ini, ecco_cfx.bak

ecco_save.ini is deleted when eccoext terminates.

When USB mode is disabled, ecco creates the following files in the windows directory %WINDIR%: ecco_cfx.bak

This means ecco will run with the ecco.cfx file in the windows directory, and eccoext will run with the ecco.cfx file in the ecco program directory when USB mode is enabled.

This happens because ecco looks for the configurations in the default location (c:\windows), and eccoext looks for the configurations in the location specified by USB mode, to determine what shows in the Toolbar (for example).

[source: http://tech.groups.yahoo.com/group/ecco_pro/message/1815

I didn't personally verify the points made in the above paragraph.]

CLEANUP HOST SYSTEM

The following batch file can be used to clean up the host system from traces in the registry and in c:\windows. It can also be used to clean up

after running the ecco installer for the purpose of creating a basic USB installation.

Please adapt the commands below to your system. You need to understand Windows batch programming. Tested on Windows XP Pro SP2. No warranty is given of any kind that this batch file will not destroy your Windows installation. You are completely on your own.//

```
REM clean file system
del c:\windows\ecco.*
del c:\windows\ecco_cfx.bak
del c:\windows\system32\DATZAP32.dll
del c:\windows\system32\prot1132.dll
del c:\windows\system32\TDLIR32.dll
REM below, if you don't have vim you can adapt using gnuwin32 sed
REM all it does is it removes the lines that ecco adds to
c:\windows\win.ini
vim +"/^\.WfxPbLinks.$/,/,;FName, LName, Company,/d" +wq
c:\windows\win.ini
REM clean registry entries
reg DELETE HKEY_CURRENT_USER\Software\NetManage\Ecco /F
reg DELETE HKEY_LOCAL_MACHINE\Software\NetManage\Ecco /F
reg DELETE HKEY_CURRENT_USER\Software\NetManage /F
reg DELETE HKEY_LOCAL_MACHINE\Software\NetManage /F
reg DELETE HKEY_CLASSES_ROOT\.eco /F
reg DELETE HKEY_CLASSES_ROOT\.ect /F
reg DELETE HKEY_CLASSES_ROOT\.ecc /F
reg DELETE HKEY_CLASSES_ROOT\.mtg /F
reg DELETE HKEY_CLASSES_ROOT\Applications\ECCO32.EXE /F
reg DELETE "HKEY_CLASSES_ROOT\NetManage EccoPro" /F
reg DELETE "HKEY_CLASSES_ROOT\NetManage EccoPro Mtg Mail" /F
reg DELETE HKEY_LOCAL_MACHINE\SOFTWARE\Classes\.ecc /F
reg DELETE HKEY_LOCAL_MACHINE\SOFTWARE\Classes\.eco /F
reg DELETE HKEY_LOCAL_MACHINE\SOFTWARE\Classes\.ect /F
reg DELETE HKEY_LOCAL_MACHINE\SOFTWARE\Classes\.mtg /F
pause
```

Several years ago I wrote a collection of tools for Ecco called EccoHelpers. eccoext has included much of that same functionality, and so I started my own exploration of eccoext with one of the extensions that I use most frequently.

As an FYI, I do a lot of project management. More than I would like... I prefer software architecture and hacking code, but frequently what I end up doing is project management. I'm actually a PMI certified "Project Management Professional" (PMP) which means that I end up managing lots of multi million dollar projects... Lots of critical path analysis, earned value calculations, etc...

Dependencies are a vital part of project management. There are several types of dependencies typically used in project plans, but the most common is a finish-to-start relationship. In essence this means that one or more tasks must be completed before another can start, or before a milestone is achieved. So if tasks A, B, and C must complete before task D can start, there is a dependency. In a simple project, there is often cascading dependencies: B depends on A, C depends on B, D depends on C, etc.

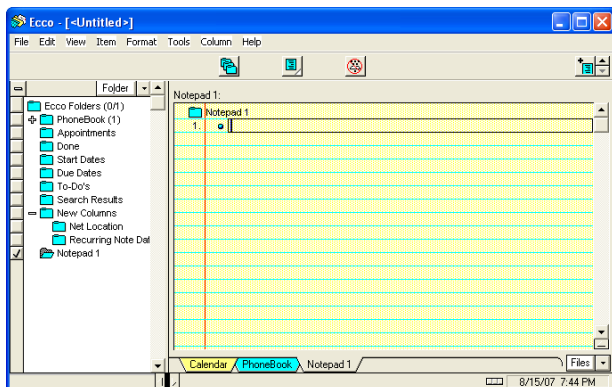
Task Name	Duration	Start	Finish	Predecessors	Aug 12, '07	Aug 19, '07
My project	4 days?	Wed 8/15/07	Mon 8/20/07			
Task A	1 day?	Wed 8/15/07	Wed 8/15/07			
Task B	1 day?	Thu 8/16/07	Thu 8/16/07	2		
Task C	1 day?	Fri 8/17/07	Fri 8/17/07	3		
Task D	1 day?	Mon 8/20/07	Mon 8/20/07	4		
Party Time	0 days	Mon 8/20/07	Mon 8/20/07	5		

This clip from MS Project shows cascading dependencies. They are indicated by the "Predecessors" column as well as in the Gantt chart.

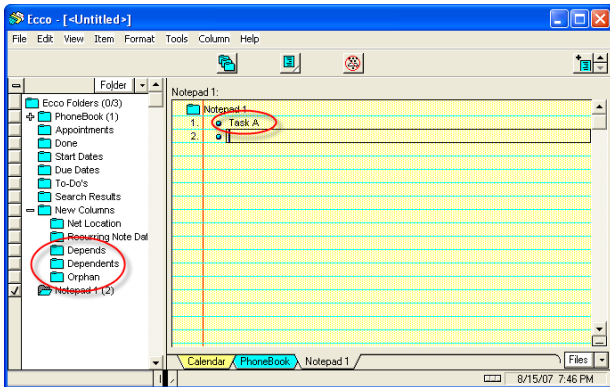
If you are doing personal task management, you might have lots of "multi step tasks" that have a single goal and a handful or more of "next actionable steps". To help me stay focussed I often like to see the ultimate multi-task goal as a TLI, and I have SLI's for the sub-steps. I use David Allen's GTD methods, and have a lot of this built into a GTD template in Ecco. I used to use my own Ecco Helpers to help manage this, but the "Depends" function in eccoext is superior since it is "real-time", i.e. as soon as a task is completed the "next actionable task" pops up! I don't have to wait for the next timed run of EH or use a workaround by keeping the queue filled with the next 3 actionable tasks...

So, here's how "depends" works...

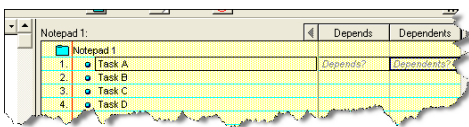
First I'll open Ecco using eccoext and open a new blank Ecco file: (see the [Quick Start](#) page)



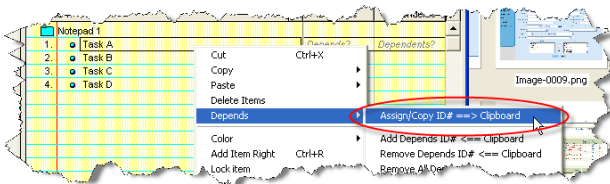
As soon as I enter an item, several new folders appear... The important ones for this discussion are "Depends" and "Dependents".



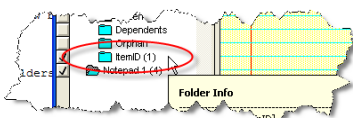
I can add those new folders as columns so that I know what is happening...



Now I right click on one of the items to see the "context" menu, and choose: Depends > Assign/Copy ID# ==> Clipboard

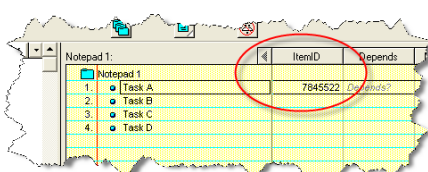


You'll notice that there is now yet another new folder called "ItemID".



Behind the scenes, the item has been assigned a unique ItemID, and that ID has been stored in the clipboard.

I've added the ItemID folder as a column so that I can see what is going on... You can see the unique ID that has been created for the item.



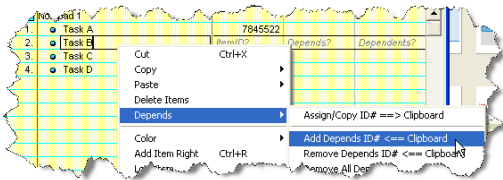
[Note! if you are following along exactly, you might have lost your clipboard. You might need to go and grab the ID again if you did any other cutting and pasting in Windows...]

Now I put my cursor in the an item that depends on Task A and right click.

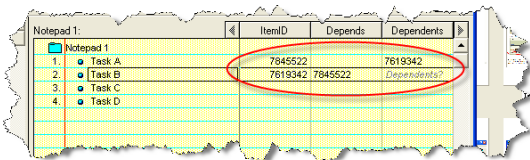
I then choose:

Add Depends ID# <== Clipboard

This copies the ItemID from the clipboard into the Depends folder/column for Task B.



What this does is make Item B dependent on Item A. You can also see that Task A now has Task B's ItemID in its "Dependents" folder/column.



By looking carefully at the ItemID numbers in the three columns you can see the "depends" relationships.

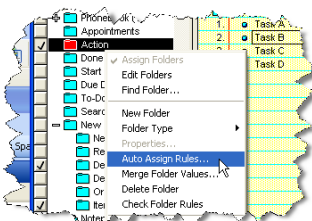
OK, so what!? What good it this?

In order to benefit, you need to create a "rule"...

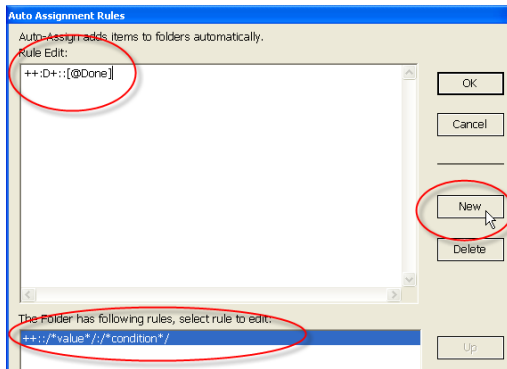
What we will do is create an "Action" folder that will hold our "next actionable tasks", and when any item is marked "Done" we will assign any dependent tasks to the "Action" folder...

I've created an "Action" folder, and what we need to do is create a rule for the folder. Any time an item in Ecco changes, all of the rules will be checked to see if the rule should be applied as a result of the changed item.

Right click the "Action" folder and click "Auto Assign Rules..."



The new eccoext auto assign rules dialog will open. Click "New" and enter the rule...



When I clicked "New" a blank template rule appeared in the list of existing rules for this folder (in the bottom half of the dialog) and also in the rule edit window (the top half). The blank template rule was:

```
++:/*value*/:/*condition*/
```

Let's look at this carefully. The first "++:" is how every rule begins. Anything between "/*" and "*/" is a comment and does nothing. So our blank rule really is just:

```
++:::
```

To be [more complete and perhaps more helpful for total beginners](#), it could say:

```
++:/*flags*/:/*value*/:/*condition*/
```

In fact the value and condition can both be fairly complex expressions. (If the "L" flag is present, they could even be replaced by programming code in the Lua language!)

Now let's look at the rule we entered:

```
++:D+::[@Done]
```

The "++:" at the beginning signals the beginning of a rule.

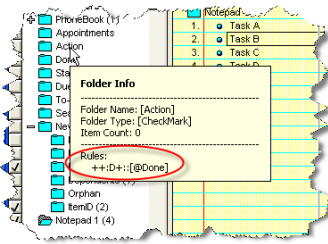
The "D+" is two flags: the "D" means that this rule only applies for a depends check, the "+" indicates that the folder that this rule is attached to will be set with the value expression if the condition expression is true and the folder currently doesn't have a value.

You will notice that the "/*value*/" comment has been removed, so the value expression is blank, we just see two colons "::". Since "Action" is a CheckMark folder, it will just be checked in this case, i.e. the item that changed will be "put into" or "assigned a value" in the "Action" folder.

The last part of our rule is the "condition expression" which is "[@Done]". The square brackets indicate that we will check a folder or column value on the item that changed. In this case the folder we will check is the "Done" folder. The "@" means that rule will not act on the item that changed, but instead will act on items that depend on the item that changed.

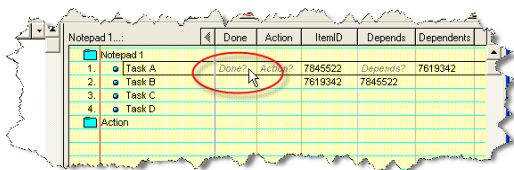
This might be a little confusing, so here it is again from a slightly different angle. When the "Done" folder of Task-A changes, the items that depend on Task-A will be modified by the value expression. Since this rule is attached to the "Action" folder, all of the dependent items will become associated with the "Action" folder...

When we close the auto-assign dialog and hover over the folder, we see that the "Action" folder now has this new rule... (the hover help is another really cool feature of the eccoext add-on...)



So now going back to our data, we see that "Task A" is item#7845522 and it has a dependent item#7619342. If we look at "Task B" we see that it is item#7619342 and that it depends on item#7845522.

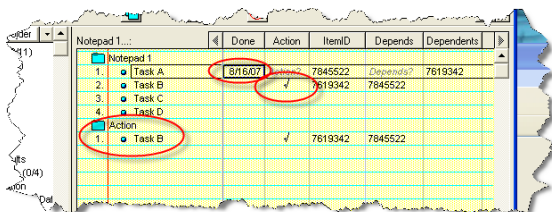
What do you think will happen when you click the "Done" column? Think about it!



OK, time's up, pencils down, pass your blue books to the center aisle...

When Task A is given an assignment in the Done folder, the Task A item is changed. This causes all of the auto-assign rules to run for the item that changed.

Our rule on the "Action" folder will fire and will see that it is watching for changes in the "Done" folder value for an item. This is indeed true, so our condition expression evaluates to true. We should apply the value part of our rule to the appropriate item. In this case, it isn't the item that changed, but we apply it to the item that "depends" on the item that changed! Item B depends on Item A so the value part of our expression is applied to the "Action" folder for Item B. Since Item A is now done Item B has become actionable.



This might SEEM like a lot of steps, but in reality it's very easy to use. You don't really need to look at the ItemID, Depends, and Dependents columns. Once the rule is set up, you can easily grab an item's ID with two clicks, select another item and paste the ID as a dependency with another two clicks.

This is *HUGE* if you have a team of programmers working on tasks and there are dependencies between the tasks. As a PM, I can assign the tasks and set the dependencies. When one programmer finishes something that someone else is waiting for, it automatically shows up in their task list. When implementing GTD, I can set up all of the actionable steps in a multi-step task, and my "Action" list stays uncluttered. I have the multi-step task or goal as a TLI and the detailed steps as SLI's. As each task is completed the next becomes visible, until the last is done and then the goal disappears as well. I will soon create a rule to mark the goal done as soon as all the sub-tasks are done... Since I organize my actions by

the context in which I can perform the action, the dependencies will remove the goal from view if the next actionable task associated with that goal needs to be performed in another context. Pretty slick!

As always, edit this page to your heart's content. My goal has been to make it as clear as possible for the persistent and highly motivated non-technical reader.

=DaveG

[Lua Tutorial](#) - If you are getting into scripting Ecco using Lua, this tutorial is a reasonable place to get going.

Ecco Pro is a program for collecting and organising information which was available commercially from 1991-1997, and since then has been free to use and download. Although it is old, it was always in many ways at least ten years ahead of its time and many of its users have stayed loyal because they have found nothing to match its combination of intuitive power and flexibility.

Ecco combines two ways of examining and arranging information: outlines and a spreadsheet-like grid. It also had a clipboard utility, known as the "shooter", which led you send it text or other information quickly and easily from other programs.

The outliner makes it very easy to arrange information in order of importance, or so that it tells a story, and then to shift it all around as priorities change, or a different story works better.

The spreadsheet grid makes it possible to tag any item in an outline with different qualities. These can be ordinary text tags (known in Ecco as 'Checkmark folders'); but they can also be dates, numbers, telephone numbers: they can even be text notes themselves.

As a very simple example of the power of this approach, imagine a journalist collecting notes for a story. Starting in the phone book, where all her contacts are kept, she can dial them from Ecco, and type the notes for each conversation in outline form under the phone book entry. Each conversation is also added to a folder for that particular story, along with any material from the web or anywhere else that seems relevant. By opening a fresh outline containing all the quotes for the story, it is easy to shuffle them around into their most effective order. By this time, the story has practically written itself. Once it is done, all of the material has been automatically tagged, dated and sorted, so that it is possible to come back years later and see exactly who talked about what.

Lawyers and (Real) Estate Agents have found the program just as useful.

However, since it has not been developed since 1997, Ecco Pro is showing its age in some respects, and the Eccoext program is an attempt to fix some of the things that have long irritated power users.

A "wiki" is a shared collection of hyperlinked pages.

When someone is looking for information, they can browse the pages to find it.

The magic of a wiki is that anyone can edit the pages and add new pages.

This wiki has one restriction which limits edits to "members" of the space.

Anyone can join, but if you put offensive content or deface the info, you will be banned from making further edits.

A wiki is incredibly free-form and the organization of the info will evolve over time.

If you see something that is wrong, don't complain, fix it.

If you see that something is missing, add it.

[Why use a Wiki instead of a forum?](#)

The person who created this wiki feels much more at home creating software than writing elegant prose.

Feel free to make any corrections that you would like!

If you don't understand something here, feel free to insert a question, and hopefully someone who understands the topic can illuminate the page.

Well, a forum is a great place for a conversation or discussion.

Post a question, get an answer.

But it's sometimes frustrating to need to read everything to find the tidbit you're looking for.

It's also frustrating trying to organize and format a forum to provide a more focussed presentation.

Personally I find it frustrating that things sort of get lost in the forums.

So... Once the questions have been asked and the solutions discussed on the forum, the resulting understanding can be cut and paste into a wiki which might make it easier to navigate. Again IMHO a hyperlinked wiki is a lot easier to use than a linear word document or PDF (but you can export the contents of the wiki and put it into a PDF if you like...)

The wiki sort of grows "organically", and it is easier to go back and tweak the content and the structure than can be done on a forum.

Of course "your mileage may vary" so choose what works for you.

A good example of a very successful wiki is [Wikipedia](#)

FWIW, Note to self: I'll need to go and make some updates to the Wikipedia entry for Ecco-Pro and add a link here...