

# Re: New "base document" available

---

*Source:* <http://coding.derkeiler.com/Archive/Cobol/comp.lang.cobol/2007-03/msg00525.html>

---

- *From:* "Pete Dashwood" <[dashwood@xxxxxxxxxxxxxxxxxxxxxxxxxxxxx](mailto:dashwood@xxxxxxxxxxxxxxxxxxxxxxxxxxxxx)>
  - *Date:* Sun, 1 Apr 2007 04:53:50 +1200
- 

This s a very good response, Rick. Thank you for taking my comments seriously and responding in a thoughtful and reasoned way.

I have added some comments below...

"Rick Smith" <[ricksmith@xxxxxxx](mailto:ricksmith@xxxxxxx)> wrote in message  
[news:130s9oip482tf38@xxxxxxxxxxxxxxxxxxxxxxxxxxxxx](mailto:news:130s9oip482tf38@xxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

"Pete Dashwood" <[dashwood@xxxxxxxxxxxxxxxxxxxxxxxxxxxxx](mailto:dashwood@xxxxxxxxxxxxxxxxxxxxxxxxxxxxx)> wrote in message  
[news:575qohF2b8qf8U1@xxxxxxxxxxxxxxxxxxxxxxxxxxxxx](mailto:news:575qohF2b8qf8U1@xxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

"Rick Smith" <[ricksmith@xxxxxxx](mailto:ricksmith@xxxxxxx)> wrote in message  
[news:130qevi9f1ebsd8@xxxxxxxxxxxxxxxxxxxxxxxxxxxxx](mailto:news:130qevi9f1ebsd8@xxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

"Roger While" <[simrw@xxxxxxxxxxxxx](mailto:simrw@xxxxxxxxxxxxx)> wrote in message  
[news:ejj78t\\$5g8\\$03\\$1@xxxxxxxxxxxxxxxxxxxxxxxxxxxxx](mailto:news:ejj78t$5g8$03$1@xxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Maybe I formulated this incorrectly.  
How to go about retrieving command line  
parameters?

Format 1 ACCEPT and DISPLAY statements using  
mneumonic-name. The implementor defines the  
mnuemonic-names and their behavior with the ACCEPT  
and DISPLAY statements. For X/Open, the names are:

ARGUMENT-NUMBER  
ARGUMENT-VALUE

The following works for me  
(using Micro Focus COBOL 3.2.24).

-----  
identification division.  
program-id. cmd-line.

Re: New "base document" available

```
data division.  
working-storage section.  
01 argc pic 99 value 0.  
01 this-program-name pic x(80).  
01 filler.  
02 argv occurs 0 to 99 times  
depending on argc pic x(80).  
01 x binary pic 9(4) value 0.  
procedure division.  
begin.  
accept argc from argument-number  
display 0 upon argument-number  
accept this-program-name from argument-value  
display this-program-name  
perform varying x from 1 by 1  
until x > argc  
display x upon argument-number  
accept argv (x) from argument-value  
display argv (x)  
end-perform  
stop run.  
-----
```

This is something that has been around for a long time.  
No one from the standards committee has seen fit to address this (Yes, I know from previous posts what was said)

Not true! The "Candidate features for a future revision" list has the following entry:

"22. 02-0160 – Command line & environment variables (scan of 93-1045) (Schepman/Woerner): J4 – 43, WG4 – investigate; see also 05-0029 – Environment variables (Klink)"

The problem may be that there are not enough people on J4 to investigate the changes required and to write those changes for the standard.

I wonder why people aren't flocking to be on J4... and how many people does

it take to document a simple function that returns a collection?

Re: New "base document" available

The annual cost to be a member of J4 is in the thousands. Attendance at meetings includes a per meeting fee plus the cost of accommodations, food, and transportation for each meeting. It all adds up! It would seem to require a commitment by a business or other organization with a vested interest in furthering the COBOL standard to absorb these costs.

And I guess they are getting thin on the ground as time goes by...OK, that's fair comment.

There are, currently, six types of intrinsic functions: alphanumeric, boolean, national, numeric, integer, and index. Each returns a temporary value to the point of activation. I don't know what you intend by "function"; but it would seem not to be related to intrinsic functions as in the standard.

If the current round of J4 endeavour is simply going to repeat the mistakes

of past J4 endeavours, what's the point? If the excellent people who ARE on

J4 (and I include you and Bill here) are simply going to run into the same old management and protocol issues, wading through molasses to get a date changed on a document when a typist could do it in 5 minutes, as if

nothing

was learned in the past, then why bother?

I am not now and have never been a member of J4! To support the COBOL standard, I have followed the activities of J4, commented on the standard, and worked with members of J4 on specific issues of my choosing.

I stand corrected. I misunderstood your role. I have great respect for your ability and it would be good for J4 if people like you WERE on it.

Re: New "base document" available

Re: New "base document" available

Hardly surprising that many of the brightest and best aren't queuing up  
for  
seats on the committee...

Still, a FUNCTION COMMAND-LINE  
would resolve these

Because other means are already in use, it seems more  
likely that using a function would confuse; not resolve.

Not that I really care, but I STRONGLY disagree with that position. Roger

is

right; a simple function would make the whole thing a lot simpler across

all

platforms. The "other means" are clumsy, ugly and need a lot more time  
and  
thought. The function could return the arguments and the number of them  
in

a

single collection. (In fact, the collection automatically holds the  
number  
of them as one of its attributes.) How hard is it to devise the syntax  
for

a

simple function that returns a collection? Why should it be such an

onerous

thing to do? It could be easily implemented on any platform, mainframe  
or  
Client/Server, so there is unlikely to be resistance from vendors.

I notice that Mr While did not provide any detail

Re: New "base document" available

Re: New "base document" available

for how FUNCTION COMMAND-LINE would work.

Forgive me; but "a simple function would make ..." seems to be a lot like "all you got to do is ...". <g>

Sometimes, "all you got to do is..." IS actually all you have to do. There is a subtle difference between over-simplifying ("All ya gotta do is..") and actually cutting to the real essentials (it worked for William of Ockham and many people since have derived benefit from following his example...:-)

Let me skip that for the moment and look at the "end" rather than the "means".

Good. You have my full attention :-)

It seems to me that the application programmer has no direct interest in the content of the command line; but is interested in how that content affects the behavior of the program.

Certainly, but that is another problem. What Roger is suggesting is an approach that allows the programmer to obtain these things he is interested in, in an easily assimilable and manipulable form. I favour a collection, but that may not have been Roger's intention. It is also important to realise that no-one is seriously suggesting dropping support for the XOpen approach, as currently implemented by some vendors. The FUNCTION COMMAND-LINE would be a language standard approach, giving people a choice, and not affecting the operation of existing programs.

Thus, it would seem that

the "end" is a record or object that may be queried to control the application. This suggests that each application would have a program or class to process the command line arguments (and environment variables) to create that record or object. (Here, I assume that command line arguments may override environment variables which override program default values and establishes a possible relationship between command line arguments and environment variables.)

As usual, you have perspicaciously identified some of the less obvious implications of any approach to command line parameters. :-)

Re: New "base document" available

Re: New "base document" available

But this is now muddying the water because there is already an established precedence for the items you mention. That should remain intact and we should confine ourselves to obtaining parameters from a command line and having them in a form that easily usable. No other considerations are important in the context of that. If there is a conflict with existing EVs or default values, that is something that must be handled exactly as it is now, but AFTER using FUNCTION COMMAND-LINE to determine exactly what the command line parameters are...

Returning now to the "means", there is the possibility that both command line arguments and environment variables need to be accessed and it would seem that accessing these in a similar manner would be beneficial. Given the request to add "FUNCTION COMMAND-LINE" to access all command line arguments, it then becomes reasonable to include "FUNCTION ENVIRONMENT-VARIABLES" as a complement to access all environment variables.

That is certainly a possibility, but for J4 one simple idea at a time might be best...:-)

We could also kill two birds with one stone and have FUNCTION GET-RUNTIME-PARAMETERS which would return two collections; one for command line and one for EVs.

(Though it would not be able to alter those variables.)  
The latter function would accept a list of environment variable names and return a collection of the corresponding values.

As presenting a list of EVs to it complicates it, why not just let it return all EVs in the context?

So, instead of one "simple" function, there are now two "simple" functions; both of which are used in a manner not consistent with other intrinsic functions.

And the second one is not really "simple" at all....:-)

I wouldn't be too worried about the incompatibility with other intrinsic functions; this one (or two, if you insist) would be a different function type, that's all.

Re: New "base document" available

Recall that intrinsic functions

return a temporary value to the point of activation,  
whereas these two functions would return object  
references that must be saved for further processing.

Yes, that's a fair comment. But, in my view at least, it is outweighed by  
the advantage of having a simple means to get run time parameters.

Perhaps,  
set command-line-arguments to command-line  
and  
set environment-values  
to environment-variables (environment-names)  
for example. Thus, these are not functions at all; but  
replace the factory method "new" to instantiate  
objects into otherwise non-OO programs. <g>

:~)

That being the case, why not just add them to the  
COBOL collection classes?

Since Micro Focus is on record as opposing the  
COBOL collection classes and has implemented  
the X/Open "means", your comment "so there is  
unlikely to be resistance from vendors" seems  
rather optimistic.

OK, I was unaware of that, although, now you mention it, I do recall Bill  
stating they were against Collections, in his capacity representing them. As  
we are not looking for trouble here and don't want to have to "sell" a  
solution to resistant vendors...

Your case above is a reasoned and persuasive one.

What about....

FUNCTION GET-RUNTIME-PARAMETERS (<pointer to an array of pointers for  
command-line>, <pointer to an array of pointers for EVs>)

Function Type: System interfacing.

Description:

## Re: New "base document" available

This function returns two sets of pointers to Key/Value pairs representing the parameters passed on the command line and the Environment Variables set in the program's context. It will initialize the pointer arrays each time it is called so, if you call it twice it will overwrite what was previously there. This may be important in checking if certain parameters have changed on certain events.

For command line parameters:

1. Parameters must be separated by commas and any number of spaces. Omitted parameters are indicated by comma with no text.
2. Keyword and positional parameters can be mixed in the same command line.
3. The function loads pointers for both keywords and values, even if no keywords are present and all command line parameters are positional.
4. Parameters are scanned left to right and if no keyword is found, a default positional keyword is entered into the keyword name by the function. This will be P1, P2, ...Pn. If a keyword is found, it is entered into the corresponding keyword name for the ordinal position in the command line where it is encountered. Keywords must be followed by = and there can be spaces between the = sign and the keyword or the value. If the keyword name has more than one word, it must be in quotes and likewise for the value.

For Environment Variable parameters:

1. The function loads both EV name and EV value for each EV in the context.

Example...

CONTEXT:

1. Environment Variables... EV1=X, EV2=Y, PARMX=Z
2. Command Line... myProg A, , C, PARMX=D, E

(The command line must cater for positional and keyword parameters and a mixture of both, see below.)

WORKING-STORAGE SECTION.

01 Command-line-parms.

12 clparms occurs max-clparms-for-program  
indexed by cl-x1.

15 clp-name usage pointer.

15 clp-value usage pointer.

01 Environment-Variables.

12 EVparms occurs max-EVparms-for-context

Re: New "base document" available

indexed by EV-x1.  
15 EV-name usage pointer.  
15 EV-value usage pointer.

PROCEDURE DIVISION.

```
....  
*> Load the pointer sets...  
FUNCTION GET-RUNTIME-PARAMETERS (Command-line-parms,  
Environment-Variables)  
....
```

What happens next depends on the requirements of the programmer... At this point, given the parameters in the example above, the two sets contain the following:

clp-name (1) points to a string containing "P1"  
clp-value (1) points to a string containing "A"  
clp-name (2) points to a string containing "P2"  
clp-value (2) points to a string containing "0x00" (null)  
clp-name (3) points to a string containing "P3"  
clp-value (3) points to a string containing "C"  
clp-name (4) points to a string containing "PARMX"  
clp-value (4) points to a string containing "D"  
clp-name (5) points to a string containing "P5"  
clp-value (5) points to a string containing "E"

EV-name (1) points to a string containing "EV1"

EV-value (1) points to a string containing "X"  
EV-name (2) points to a string containing "EV1"

EV-value (2) points to a string containing "Y"  
EV-name (3) points to a string containing "PARMX"

EV-value (3) points to a string containing "Z"

.... all other entries are null pointers.

It is pretty trivial to check both tables and find that the original EV value of PARMX ("Z") has been overridden on the command line to now be "D".

Similarly, searches can be done on either table using the names, to locate parameters the program may be interested in.

The "dummy" keyword names for the command line don't HAVE to be the ones I used (they are intended as an example); maybe a COBOL reserved word with a position concatenated to it would be better, and guarantee no possible confusion if an application had a keyword parameter named "P1"...

I realise I just wasted 40 minutes doing this, but it made a welcome relief from some papers I was reading, and I wanted to reassure myself that the

Re: New "base document" available

process of documenting a new feature in COBOL CAN be done in less than 17 years...:-)

I honestly believe the above would be a useful function (or whatever, if you REALLY don't like it being a function...) and I don't think it would be difficult to implement (the strings would be on the heap anyway). If I was still developing COBOL I'd go ahead and write it as a component...:-)

In any case, the X/Open standard is a sufficient "means" to accomplish the suggested "end".

Well, that is a matter of opinion...:-) I would agree it serves the purpose, but it is inelegant and inflexible.

Pete.

.