DYALOG

APL Germany e.V.

Mainz, April 30th 2024

Avoiding Execute

(System functions for getting and setting variable values)

Morten Kromberg, CTO

Dyalog – The Next Generation

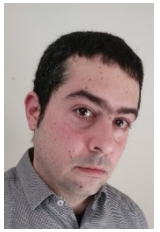
2010-2021





2022



2023





2024

2023 Summer Interns


Dyalog – The Next Generation

2010-2021

2022

2023

2024

2023 Summer Interns

Welcome to Dyalog Videos Page

dyalog.tv/Dyalog23/?v=eya2FlHinHE

Relaunch to update

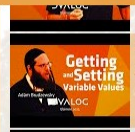
Apps | Link | APL | Flying & Sailing | Car | Dyalog | Cloud | SBO | Travel | Linux | Sport | Productivity | Ferie 2022

DYALOG

Elsinore 2023

Dyalog '23 | APL Seeds '23 | Dyalog '22 | Webinar | APL Seeds '22 | Dyalog '21 | APL Seeds '21 | Dyalog '20 | Dyalog '19 | Dyalog '18 | Dyalog '17 | Dyalog '16 | Dyalog '15 | Dyalog '14 | Dyalog '13 | Dyalog '12 | Dyalog '11 | APL Berlin 2010

Dyalog '09 | Dyalog '08



Setting and Getting Variable Values
Adám Brudzewsky

Revisiting SH and CMD
Peter Mikkelsen

Giving Key a Vocabulary
Adám Brudzewsky

An Implementation of APL Array Notation
Kamila Szewczyk

APL Problem Solving Competition

How I Won the APL Problem Solving Competition – Introduction and Prize Ceremony
Brian Becker

How I Won the APL Problem Solving Competition – Non-student Winner
Alexander Block

How I Won the APL Problem Solving Competition – Student Winner
Andrea Piseri

I want to...

- get the values of variables using an array of variable names
- set variables using arrays containing names and values
- set a default left argument for an ambivalent tradfn
- base a new namespace on two source namespaces
- query data objects, but some have missing values
- construct a namespace from names and values
- populate class fields from name–value pairs
- convert between tables and namespaces
- check the value of an optional global

GET values of names w/out ⍎

- New function ⎕NG (Name Get) takes a list of names and returns values:
- If we have:

```
(First Last)←'Max' 'Munstermann'
```

- Then:

```
      ⍴⍴ ⎕NG 'First' 'Last'
 3   11
```

- Also allows a single simple name, and a character matrix of names

GET with **default values**

- ⎕NG can take name value pairs, the values are returned if the name does not exist:

```
      ⎕NG ('First' '')('Last' '')('Height' ¯1)
Max   Munstermann   ¯1
```

- Not all names need a fallback value:

```
      ⎕NG 'First' 'Last' ('Height' ¯1)
Max   Munstermann   ¯1
```

GET with an array of namespaces

- By default, names are found in the current namespace
- The optional left argment of ⎕NG can be one or more namespaces
- If records is a vector of spaces representing people:

```
      ⊃records ⎕NG 'First' 'Last' ('Height' ¯1)
Max        Munstermann  182
Lieschen   Müller       167
John       Doe           ¯1
```

- The ability to provide defaults is particularly useful in this case, for example when processing JSON input with missing values

Default left argument of TRADFN

- Finally, a nice(er) way to do this:

```
∇ r←{left} Foo y
  life←⎕NG ⊂'left' 42
  …
∇
```

- Nicer than `:If 0=⎕NC 'left' …`
- Not all names need a fallback value:

```
   ⎕NG 'First' 'Last' ('Height' ¯1)
Max  Munstermann  ¯1
```

Merge Namespaces

- Create a `merged` namespace, where values from `input` overwrite `defaults`

```
merged←input ⎕NG defaults
```

- Merge a hierarchy of value sources (e.g. configuration):

```
new←⎕NG/namespaces
```

Get Names **AND** Values: ⎕NV

- Return names **and** values for given name classes

```
(names values)←⎕NV 2  ⍝ Positive: Name Matrix
        pairs←⎕NV ¯2 ⍝ Negative: Name Value Pairs
  pairs←source ⎕NV ¯2
```

SET names using name/value pairs

- The existing function ⎕NS is extended with name/value pairs:

```
ref←target ⎕NS ('First' 'Max')('Last' 'Munstermann')
⍝ Equivalent to:
target.(First Last)←'Max' 'Munstermann'
```

- NB: if no `target` is given, a new namespace is created and `ref` points to it
- `target` can be ⎕THIS or #, of course

SET using vectors of names & values

- If you have an array of values, name/value pairs may be inconvenient.
- You can use one array of names and one of values:

```
vars   ← 'First'    'Last'
values ← 'Lieschen' 'Müller'

target ⎕NS (↑vars) values
```

- The first element must be a matrix of names to distinguish this from the name/value pair case

Summary Examples

Get:

```
      [source] ⎕NG 'Name' ('Height' ¯1)  ⍝ Get values w/defaults
merged←input ⎕NG defaults                ⍝ Merge namespaces
```

List & Get:

```
(names values)←[source] ⎕NV 2      ⍝ Name Matrix and values
         pairs←[source] ⎕NV ¯2     ⍝ (Name Value) Pairs
```

Set:

```
ref←[target] ⎕NS ('First' 'Lieschen')('Last' 'Müller')
ref←[target] ⎕NS (2 5⍴'FirstLast ')('Max' 'Munstermann')
```

I want to...

- get the values of variables using an array of variable names
- set variables using arrays containing names and values
- set a default left argument for an ambivalent tradfn
- base a new namespace on two source namespaces
- query data objects, but some have missing values
- construct a namespace from names and values
- populate class fields from name–value pairs
- convert between tables and namespaces
- check the value of an optional global

Still only a proposal…

As I prepared these slides, it struck me that I would prefer:

- ⎕VGET for ⎕NG
- ⎕VSET for ⎕NS
  - With no left argument meaning “current space” rather than "create new space"
- ⎕NLV for ⎕NV (Name List with Values)

- … and leave ⎕NS unchanged, for creating NameSpaces and copying names into NameSpaces

We should have done this 20 years ago!