Migration to Dyalog
Transforming APL+Win workspaces in proper Dyalog ones

Dr. Markos Mitsos
markos.mitsos@ergo.de

ERGO Group AG, Actuarial Department Health (DKV)

Dyalog 2024 — Glasgow

ERGO
A Munich Re company

About

Report on migration (and its progress):

- from APL+Win to Dyalog
- ongoing for years, moving really slow
- based on WS management framework

Targeting a better APL system:

- trying to create “true” Dyalog workspaces, not APL+Win copies
- using Dyalog specific features
- in fact combined with overhauling of old code / structures

ERGO

A Munich Re company

Outline

ERGO

A Munich Re company

Infrastructure for code management

- code in text files
  - simply use **Link**!
- versioning through TortoiseSVN
  - disentangle code and version management
  - one repository for each workspace
  - code publication in “official” checkout
- workspace build modes
  - build workspace from working copy
  - bi-directional link for coding
  - uni-directional link for debugging
  - use in parallel, do not save workspace

▸ Code and debug workspace

ERGO

A Munich Re company

Testing and deploying

- test infrastructure
  - test results too long to write down
  - transform into text (Array Notation)
  - save in database (DB2)
- automated tests
  - design tests and inspect results manually
  - fix target results in database
  - compare actual test results with targets
- deploy workspace
  - build workspace from repository
  - run all tests
  - save workspace in target folder

ERGO

A Munich Re company

Outline of section on structural changes

In this section we outline:

Workspaces structure and cooperation of workspaces

Objects structure of objects and use of globals

ERGO

A Munich Re company

Schematic workspace superstructure

High level WS structure conforming to framework

- sources
  - “own” ns primarily from working copy
  - “foreign” ns primarily from repository
- objects
  - no vars/fns/ops under #
  - only ns under #

▸ WS superstructure

ERGO

A Munich Re company

Workspace structure and cooperation

Contents proper in main ns <**nsmn**>

	APL+Win	Dyalog
structure	flat	2 ns levels
object distinction	name prefixes	main ns name
result	rather chaotic	clear
		collections not too big

▸ WS content structure (new) ▸ WS content structure (comparison)

ERGO
A Munich Re company

Exchange of code and data

Cooperation between workspaces

	APL+Win	Dyalog
code exchange	manual	automatically
	when necessary	during WS build
		fixed at deployment
data exchange	component files	

Streamline interfaces, use APL+Win cf as bridge between migrated and not-yet-migrated WS

▸ Exchange code during deployment ▸ Exchange of data through cf

ERGO
A Munich Re company

Schematic object structure

Coarse structure present in APL+Win, formalized/stringently enforced in Dyalog

1. header, fixed settings like Migration Level 1
2. activate error handling
3. process/control right argument and left parameters
4. main part
5. clean-up, errors ignored

ERGO
A Munich Re company

Header, parameters and result

Header most significant change

	APL+Win	Dyalog
	one-line	multi-line
long?	very	no!
object structure...	more or less...	reflected
namespaces		main data
		similar parameters
		[local]

Convention for passing namespaces

- as reference when only to be used
- as copy when to be modified

▸ Header in APL+Win ▸ Header in Dyalog

ERGO
A Munich Re company

Local, semi-global and global

Object as far as possible functional

	APL+Win	Dyalog
preferred	local variables	
problems	number of variables	
	amount of data	
deviations	ugly semi-globals	
exceptions	"save" objects globally	

Specific full ns paths reserved for globals, managed by special function[s]

▸ Global names function

ERGO

A Munich Re company

Outline of section on coding changes

In this section we outline:

Methods new/different Dyalog methods and capabilities

OOP Object Oriented Programming with APL syntax

ERGO

A Munich Re company

Local namespaces

Local namespaces widely used

- often a multitude of similar variables needed in function/operator
- define local ns as container
- reduces localisation / header significantly
- enhances readability and highlights connections

▸ Namespaces local to function

ERGO
A Munich Re company

Dfns and small algorithms

Dfns and dops (for small algorithms) as new tool

- APL+Win written out because too small for tradfn
- Dyalog dfn OK as separate object
- collect some small algorithms as utilities
- also useful inline

▸ Dfns as stand-alone objects ▸ Dfns as on-the-fly objects ▸ Dfns as inline code

ERGO
A Munich Re company

Trains (of thought)

Not many trains yet, but thinking about...

- still, a few have appeared!
- getting used to them...

▸ First trains in code...

ERGO

A Munich Re company

More/new[er] primitives

Reduction of own functions, new functionality, clarity of code

- Key ⌸ for structure and grouped operations, Index Of ⍸ for matrix search, replace self-written algorithms
- Where ⍸ reduces code and enhances readability
- At @ makes code clearer and avoids necessity to assign
- Power ⍣ for conditional application (and one "real" case in a check...)
- building of lists or statements more readable with modified assignment

▸ Number of representatives ▸ Grouped operation (sum) ▸ Index of matrix rows
▸ Check and index selection ▸ Selective replacement ▸ Case distinction

ERGO
A Munich Re company

More/other/new[er] system functions

Reduction of own functions, new functionality

- existence and erasure of files (fso objects versus □NEXISTS and □NDELETE, also □NPARTS)
- reading and writing small files (□NGET and □NPUT compact)
- date arithmetic (own algorithms versus □DT)
- use of regular expressions (□S and □R)

▸ Existence and erasure of files ▸ Reading and writing small files ▸ Date arithmetics
▸ Date filed checks ▸ Regular expressions

ERGO
A Munich Re company

Proper Classes

Running protocol as example

	APL+Win	Dyalog
Class	Windows Form variant	proper class
Instance		proper namespace
passed as	name	reference
“saved” as	“somewhere”	global reference
new methods		timestamp message
		reaction to decision

▸ Running protocol

ERGO
A Munich Re company

Main and parameter GUI

Schematic interaction with user through GUI

	APL+Win	Dyalog
main	similar functions	schematic, utility
parameter	schematic, utility	schematic, utility
based on	pages	subforms
new functionality		multiple Grids
		direct access to namespace

▸ Direct access to namespace ▸ Main GUI ▸ Parameter GUI

ERGO
A Munich Re company

COM as namespace

ADO (databases) or Excel

	APL+Win	Dyalog
objects	"Windows"	namespaces
syntax	some tricks, redirectional	direct access to namespace

Some new functions implemented by new developer!

▸ COM syntax

ERGO

A Munich Re company

Conclusion

Future

- utilities/infrastructure almost done
- simulations proper on the horizon
- also error handling, modularisation, tests...
- still long-time project...
- but the resulting Dyalog WS are better than the sources!

◂ introduction

ERGO

A Munich Re company

Overview of examples and illustrations

- Code and debug workspace

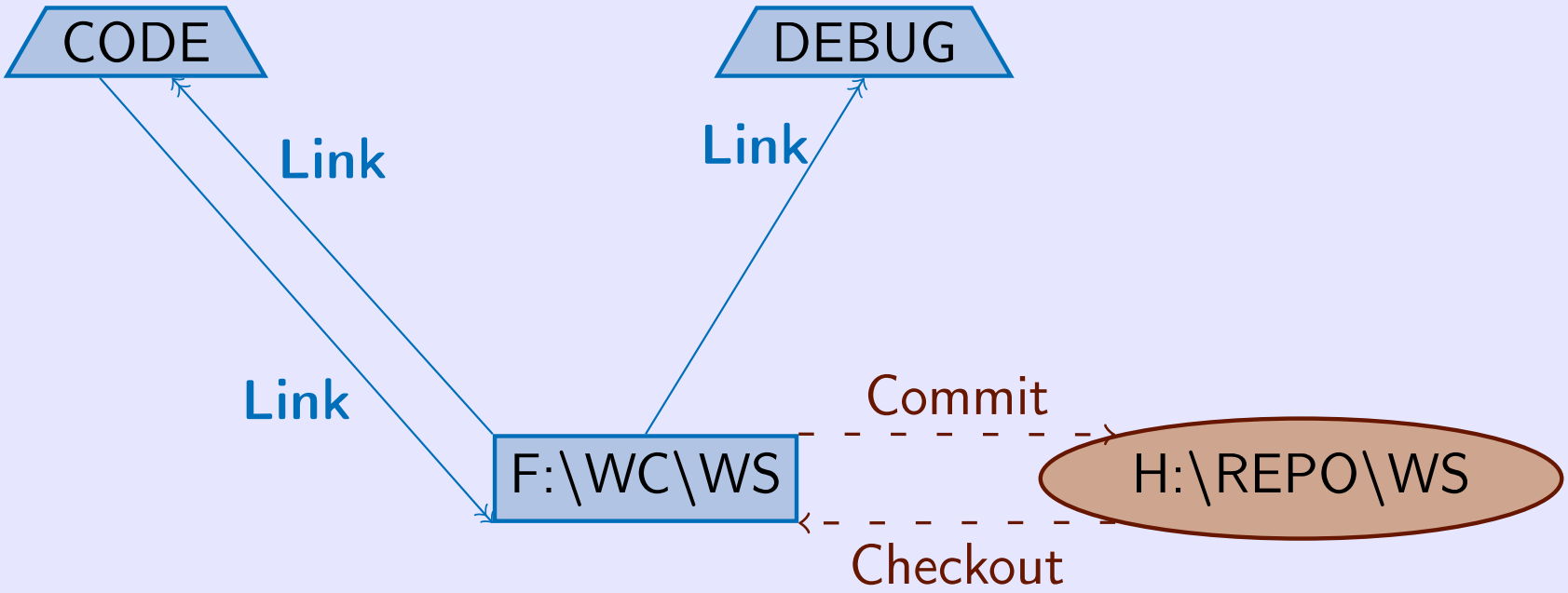
- WS superstructure
- WS content structure (new)
- WS content structure (comparison)
- Exchange of code during deployment
- Exchange of data through cf
- Header in APL+Win
- Header in Dyalog
- Global names function

- Namespaces local to function
- Dfns as stand-alone objects
- Dfns as on-the-fl objects
- Dfns as inline code
- First trains in code. . .
- Number of representatives
- Grouped operation (sum)
- Index of matrix rows
- Check and index selection
- Selective replacement
- Case distinction
- Existence and erasure of files
- Reading and writing small files
- Date arithmetics
- Date filed checks
- Regular expressions
- Running protocol
- Direct access to namespace
- Main GUI
- Parameter GUI
- COM syntax

ERGO
A Munich Re company

Code and debug workspace

Usual work setup, two WS in tandem:



◂ Infrastructure for code management

ERGO
A Munich Re company

WS superstructure

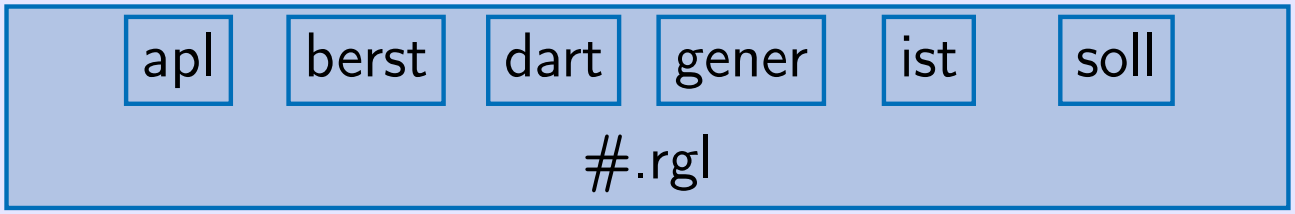
#.<nsmn>	code proper of workspace
#.build	building instructions and tests cases
#.test	alternatives, ideas,. . .
#.<nsfnX>	imported foreign namespace[s]

#.temp	temporary, for example during build
#.globals	global object, for example COM

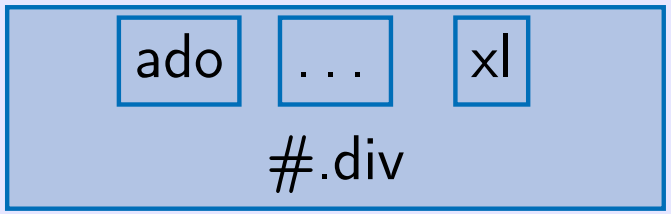
◂ Schematic workspace superstructure

ERGO
A Munich Re company

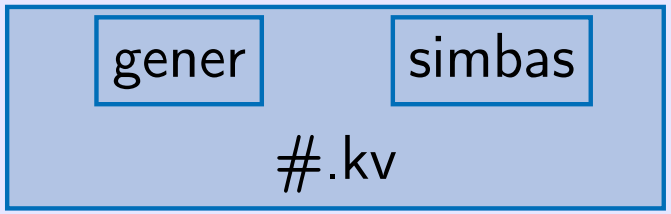
WS content structure (new)



own code of workspace



general utilities



health utilities

◂ Workspace structure and cooperation
▸ WS content structure (comparison)

ERGO
A Munich Re company

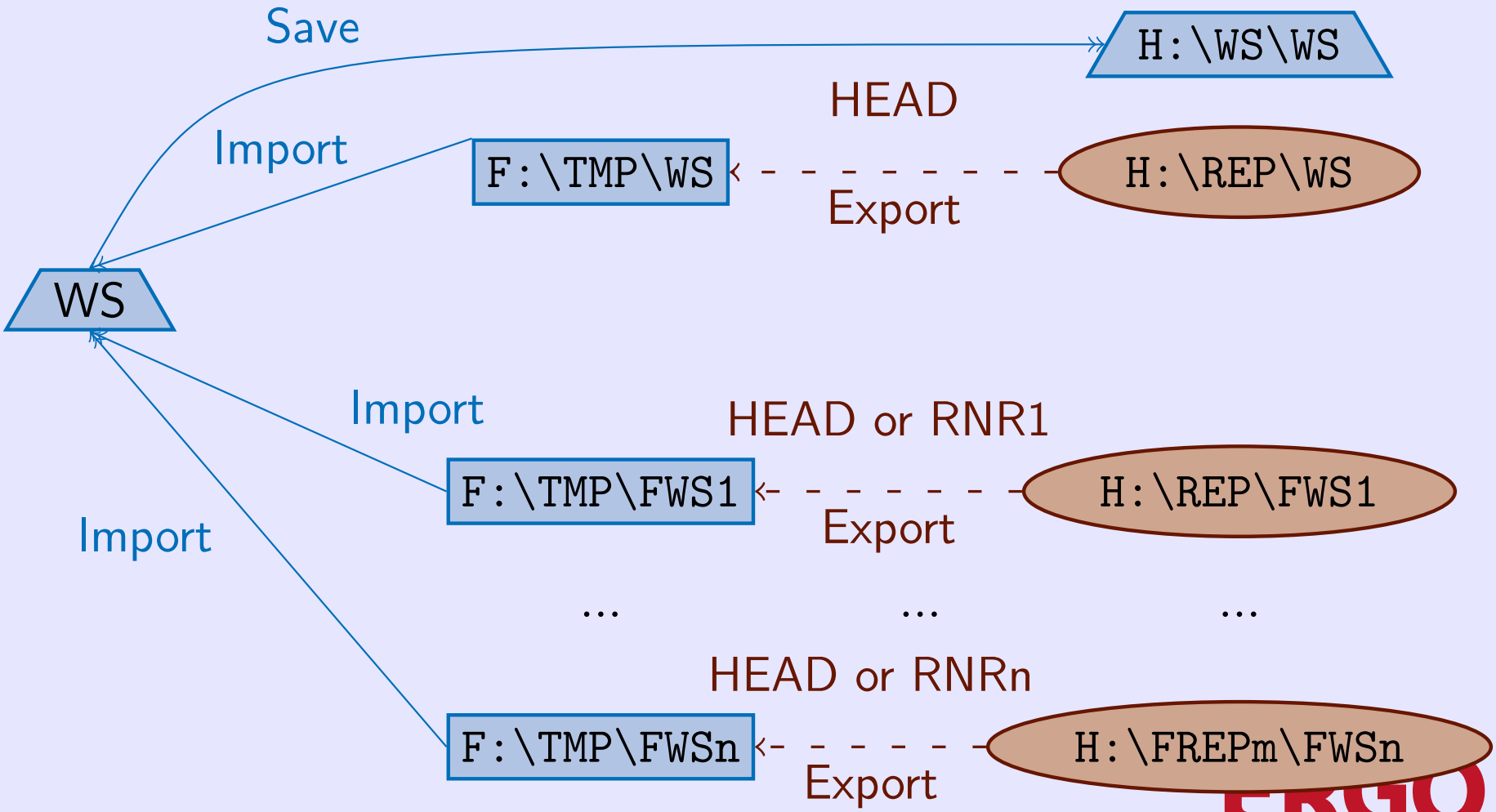
WS content structure (comparison)

APL+Win	Dyalog
actuarial data	
RGL_BEREITSTELLEN	#.rgl.gener.BEREITSTELLEN #.rgl.berst.BEREITSTELLEN
RGL_KALK_BEREITSTELLEN	#.rgl.soll.BEREITSTELLEN
RGL_KALK_DART_BEREITSTELLEN	#.rgl.dart.BEREITSTELLEN
RGL_KALK_DART_APL_BEREITSTELLEN	#.rgl.apl.BEREITSTELLEN
RGL_TM_BEREITSTELLEN	#.rgl.ist.BEREITSTELLEN
RGL_TM_BER_ST_KVTB*	#.rgl.ist.BER_ST_KVTB*
RGL_TM_HGB_BEREITSTELLEN	#.rgl.ist.HGB_BEREITSTELLEN
RGL_TM_HGB_BER_ST_KVTB*	#.rgl.ist.HGB_BER_ST_KVTB*
general utilities	
ADO_CONNECT	#.div.ado.CONNECT
ADO_SELECT	#.div.ado.SELECT
health utilities	
KV_GRD_BEREITSTELLEN	#.kv.simbas.GRD_BEREITSTELLEN

ERGO
A Munich Re company

◂ Workspace structure and cooperation ▸ WS content structure (new)

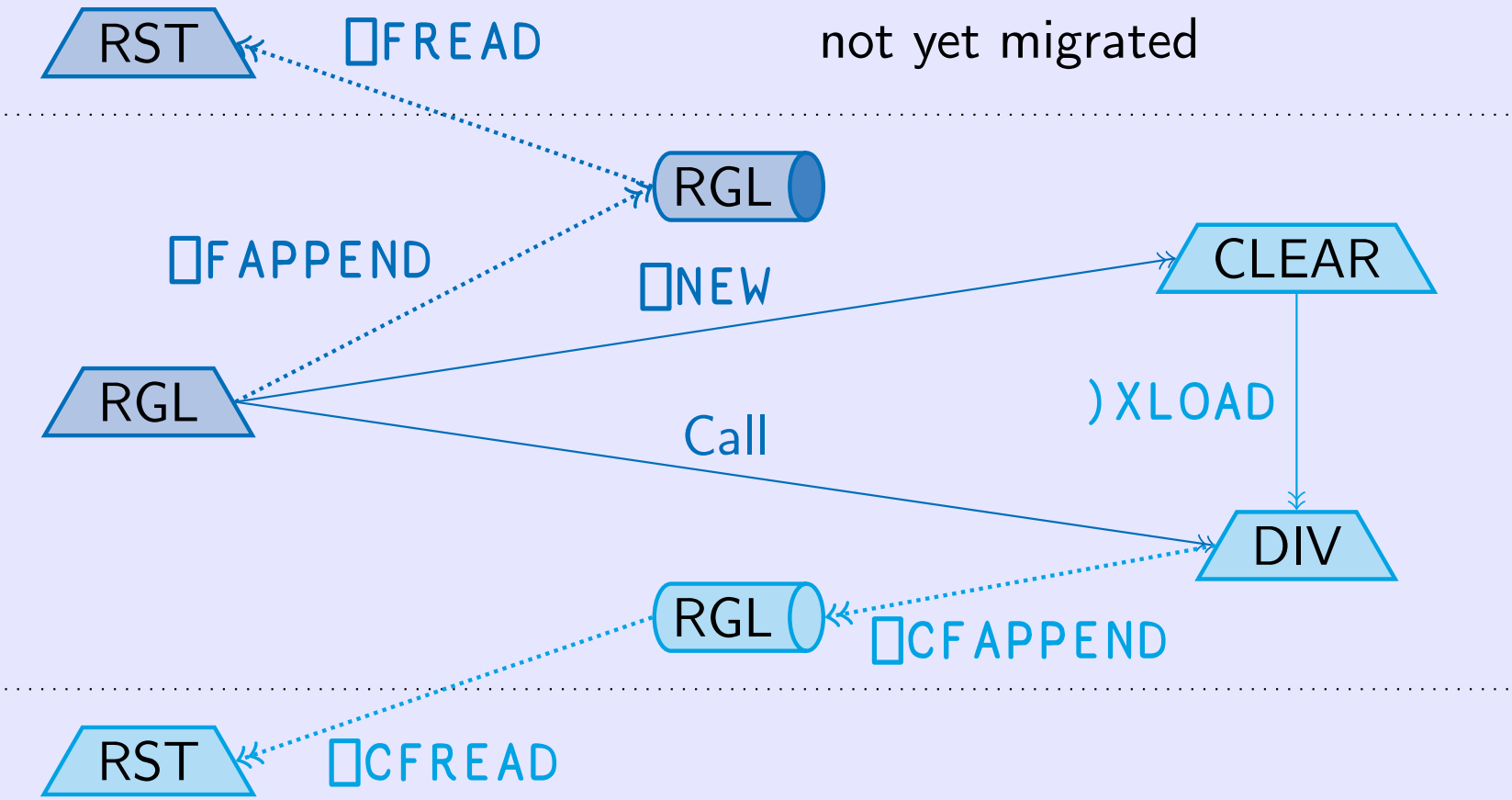
Exchange of code during deployment



◂ Exchange of code and data ▸ Exchange of data through cf

ERGO
A Munich Re company

Exchange of data through cf



◂ Exchange of code and data
▸ Exchange of code during deployment

ERGO
A Munich Re company

Header in APL+Win

Length ≈ 300 characters, some ≫ 1.000, variants for similar functions

```
ERG ← PRMS RGL_TM_BER_ST_KVTB0171 ARG;⎕elx ;TARIFE_TAID_DEF
;TARIFE_TAID ;HINW_UEB_DEF ;HINW_UEB          ...          ;V;VV;I
...
(TARIFE_TAID_DEF TARIFE_TAID HINW_UEB_DEF HINW_UEB
                 HINW_UEB_NR_MAX HINW_VOR_DEF HINW_VOR) ← 7 ↑ ARG
...
(BILANZ DATUM_VON DATUM_BIS STELL_KAV CONNR MSGBN PROT
                    WBNR XLN_PRMS RESP_MIN RESP_MAX) ← 11 ↑ PRMS
...
```

◂ Header, parameters and result ▸ Header in Dyalog

ERGO
A Munich Re company

Header in Dyalog

Identical for similar functions

```
(TARIFE_TAID rgl hinw) ← PRMS BER_ST_KVTB0171 ARG ;⎕ML;ERR_MSG
                                                 ;ERR_MSG_BEG ;⎕TRAP
 ;TARIFE_TAID_DEF;TARIFE_TAID;rgl;hinw          ⍝ Argument
 ;DATUM_VON;DATUM_BIS;regst;db;msgb;sondst      ⍝ Parameter
...
 ;V;VV;I;II;ns                                  ⍝ temporär
...
 (TARIFE_TAID_DEF TARIFE_TAID V V) ← ARG ← 4 ↑ ARG , ...
 'rgl' 'hinw' ⎕NS¨ ARG[3 4]
...
 (DATUM_VON DATUM_BIS V db msgb V) ← PRMS ← 6 ↑ PRMS , ...
 'regst' 'sondst' ⎕NS¨ PRMS[3 6]
...
```

◂ Header, parameters and result ▸ Header in APL+Win

ERGO
A Munich Re company

Global names function

```
    #.div.gener.GLOBAL_NAMES 'globals'
#.globals
    1 2 3 #.div.gener.GLOBAL_NAMES 'com' 'xl'
                                        'rng'
#.globals.com.xl.wbc.wb_01.sc.s_02.rng_03
    1 #.kv.gener.GLOBAL_NAMES 'simbas' 'rgl'
                                        'hgb'
#.globals.simbas.rgl.hgb_01
    #.rgl.gener.GLOBAL_NAMES 'path' 'ist'
H:\Aktuariat_Simulationsbasis\
            Rechnungsgrundlagen fachlich\
                    aus IST eingelesen\
```

◂ Local, semi-global and global

ERGO

A Munich Re company

Namespaces local to function

Comparison of actuarial data distinguishes many cases and collects statistics

```
'stat' 'hinw_ign' 'hinw_ber' 'vorg' ⎕NS¨ ⊂''
V ← (0 , ≠rgl_def.DEF) ⍴ 0
stat.BEARBL ← 'DELETE' 'UPDATE' 'INSERT'
stat.HINW_DEF ← ''
...
hinw_ign.(ALT_NUR_SOLL ALT_NUR_IST) ← ⊂V
...
hinw_ber.(F_DSP_SOLL F_DSP_IST) ← ⊂V
...
vorg.(RGL_D RGL_U RGL_I) ← ⊂V
```

A total of 7/24/14/3 variables collected in 4 namespaces

◂ Local namespaces

ERGO
A Munich Re company

Dfns as stand-alone objects

"Trim right" in APL+Win always inline, only whitespace

```
(- + / ⍲ \ ' ' = ⌽ V) ↓ (V ← VAR)
```

in Dyalog slightly more general function

```
TRIM_R ← {⍺ ← ' '
...
1 ≥ |≡⍵ : (- + /  ⍲ \  (⌽  ⍵) ∈   ⍺) ↓  ⍵
1 ≥ |≡⍺ : (- + /¨ ⍲ \¨ (⌽¨ ⍵) ∈¨ ⊂⍺) ↓¨ ⍵
          (- + /¨ ⍲ \¨ (⌽¨ ⍵) ∈¨  ⍺) ↓¨ ⍵
        }
```

◂ Dfns and small algorithms ▸ Dfns as on-the-fl objects ▸ Dfns as inline code

Dfns as on-the-fly objects

Preparation for grouped operation (scattering), in APL+Win unreadable, in Dyalog series of dfns

```
...
CODEL ,← '{((+ / ⍵ * 2) ÷ (≢⍵)) * ÷2}'
 '{((+ / (⍵ - (((+/)÷≢) ⍵)) * 2) ÷ (≢⍵)) * ÷2}'
...
CR ← ('↓' '{((×⍺) × (≢⍵) ⌊ |⍺) ↑ ⍵}') ⎕VV ← ...
CA ← (,¨ (⊂'G< AUSG[99999] >') ⎕FMT¨ V) ,¨ CR
CS ← ('' ' {⍵[⍋⍵]}' ' {⍵[⍒⍵]}') [1 + + ≠ ...
...
CODE ← ... (V ⍎¨ ⊂CODEL) ,¨ CA ,¨ CS ,¨ ...
FN ← ⍎ '{' , ((≢TR) ↓ CODE) , '}'
```

◂ Dfns and small algorithms ▸ Dfns as stand-alone objects ▸ Dfns as inline code

ERGO
A Munich Re company

Dfns as inline code

Preparation for GUI checks (“right data type?”), code used later

```
...
VVV ← '{α ← 0 ◇ ''?'' ≡ ω : 1 ◇ (0 ≥ α) ^ (I ←
~ (⎕DR ω) ∈ 80 160 320 326) : 0 ◇ 0 ≥ α : 1 ◇
I : (⊂ρω) ∈ ,¨ (0ρ0) 1 α ◇ (V ← ⎕VFI ω) [1] ∈
(,1) (αρ1) : 1 ◇ 0}'
...
CHECKT ←⍨ (,¨ (⊂ 'G<(9999⊃CnD.DATA_TYPE)>, < '
, VVV , ' >, G<(9999⊃CnD.VAL)>') ⎕FMT¨ ,[0.5]¨
2 /¨ ⍳I) , V ,[1.5] ((⊂'Nur alphanumerische
Einträge sind beim Control "') ,¨ VV ,¨ (⊂'"
erlaubt'))
```

◂ Dfns and small algorithms ▸ Dfns as stand-alone objects ▸ Dfns as on-the-fl objects

Trains (of thought)

Preparation for grouped operation (averages), in APL+Win unreadable, in Dyalog trains

```
...
OPER ,← '⌿' '⌽' '⍉'
CODER ,← '((+ /) ÷ ≢)'
         '((× /) * (÷ ≢))'
         '((+ / |) ÷ ≢)'
CODES ,← '((+ \) ÷ (⍳ ≢))'
         '((× \) * (÷ (⍳ ≢)))'
         '((+ \ |) ÷ (⍳ ≢))'
...
```

◂ D-fns and small algorithms

ERGO
A Munich Re company

Number of representatives

Count representatives of equivalence classes (groups), in APL+Win own algorithm

```
VV ← DATEN[⍋DATEN;]
I ← 1 , ∨ / (1 ↓[1] VV) ≠ (¯1 ↓[1] VV)
V ← I / ⍳ ↑ ⍴VV
L ← (1 ↓ V , (1 + 1 ↑ ⍴VV)) - V
(I ≠ VV) , L
```

in Dyalog Key

```
({⍺ , (≠⍵)} ⌸) DATEN
```

◂ More/new[er] primitives ▸ Grouped operation (sum) ▸ Index of matrix rows
▸ Check and index selection ▸ Selective replacement ▸ Case distinction

ERGO
A Munich Re company

Grouped operation (sum)

Sum data for each equivalence class (group), in APL+Win own algorithm

```
I ← 1 , ∨ / (1 ↓[1] DATEN) ≠ (¯1 ↓[1] DATEN)
V ← I / ⍳ ↑ ⍴DATEN
L ← (1 ↓ V , (1 + 1 ↑ ⍴DATEN)) - V
MAX ← ⌈ / L
II ← , L ∘.≥ ⍳MAX
VV ← ((⍴V) , MAX , (1 ↓ ⍴WERTE)) ⍴ II ⌿ WERTE
DATEN[V;] , (+ /[2] VV)
```

in Dyalog Key

```
DATEN ({⍺ , + ⌿ ⍵} ⌸) WERTE
```

◂ More/new[er] primitives ▸ Number of representatives ▸ Index of matrix rows
▸ Check and index selection ▸ Selective replacement ▸ Case distinction

ERGO
A Munich Re company

Index of matrix rows

Indexing matrices, in APL+Win own algorithm

```
IND ← (1 ↑ ρDATEN) ρ 1 ⋄ N_TR ← 1 + 1 ↑ ρREF
LISTE ← REF ,[1] DATEN
Z_NR ← (- ⍳ 1 ↑ ρREF) , (⍳ 1 ↑ ρDATEN)
LISTE ← LISTE[V ← ⍋LISTE;] ⋄ Z_NR ← Z_NR[V]
BEG ← (1 , ∨ / (1 ↓[1] LISTE)
                ≠ (¯1 ↓[1] LISTE)) / ⍳ 1 ↑ρLISTE
V ← (+ \ 0<Z_NR) [¯1 + 1 ↓ BEG , (1+ρZ_NR)]
V ← V - 0 , ¯1 ↓ V
IND[(0 < Z_NR) / Z_NR]
              ← V / N_TR - N_TR | 0 ⌊ Z_NR[BEG]
```

in Dyalog Index Of

```
IND ← REF ⍳ DATEN
```

◂ More/new[er] primitives ▸ Number of representatives ▸ Grouped operation (sum)
▸ Check and index selection ▸ Selective replacement ▸ Case distinction

ERGO
A Munich Re company

Check and index selection

Use of Where, `I / ⍳ ↑ ⍴I` changed to `⍸I`

Checks, for example passed arguments

```
:If ∨ / (I ← ~ BEARB ∈ V)
    ERR_MSG ← 'Unerlaubte Anweisung(en) in den
Spalten ' , (⍕ ⍸I) , ' "' , (⍕ I / BEARB) , '"
angegeben, erlaubt:' , (⎕UCS 13) , ⍕V
    →ENDE
:EndIf
```

Selection of relevant indices

```
:For NR :In ⍸REL_I
...
:EndFor
```

◂ More/new[er] primitives ▸ Number of representatives ▸ Grouped operation (sum)
▸ Index of matrix rows ▸ Selective replacement ▸ Case distinction

Selective replacement

Trivial use of At for replacement of line end characters with whitespace

```
FN ← {⎕UCS (32 @ (⍳ V ∈ 9 10 13 14 133))
                                    (V ← ⎕UCS ⍵)}
(I  / ,DATENS[;V]) ← FN¨ I  / ,DATENS[;V]
(II / ,DATENI[;V]) ← FN¨ II / ,DATENI[;V]
```

a case distinction

```
(VV VVV) ← (⊂¨ '<ignorieren>' '<kein>')
     ({⍺} @ (⍳ ~ LIST_DEF ∈ LISTS_DEF))¨ VV VVV
```

and a negation of a boolean column

```
IND ← TYPES ⍳ (⍱ (~ @ 4) ⍱TYPES)
```

◂ More/new[er] primitives ▸ Number of representatives ▸ Grouped operation (sum)
▸ Index of matrix rows ▸ Check and index selection ▸ Case distinction

ERGO
A Munich Re company

Case distinction

Use of Power is for trivial case distinctions during clean-up

```
V ← ('unbedingt abbauen' '' CnP.CONNR)
                  (##.ado.CONNECT ⍤ VERB_LOK_I) ''
```

or processing of passed parameters

```
(DEF STSP GBSP GESP) ← ,⍤⍤
 {(⊂ ⍤ (((⎕DR ⍵) ∈ 80 160 320) ∧ (0 < ≢⍵))) ⍵}⍤⍤
                                    DEF STSP GBSP GESP
ZSP ← , {(⊂ ⍤ ((⎕DR ⍵) ∈ 80 160 320)) ⍵} ZSP
```

◂ More/new[er] primitives ▸ Number of representatives ▸ Grouped operation (sum)
▸ Index of matrix rows ▸ Check and index selection ▸ Selective replacement

Existence and erasure of files

APL+Win use of Windows Object with Methods like

```
V ← 'fso' ⎕wi 'Create'
                    'Scripting.FileSystemObject'

EX_DAT ← 'fso' ⎕wi 'XFileExists'              DATN
PATH   ← 'fso' ⎕wi 'XGetParentFolderName' DATN
BASE   ← 'fso' ⎕wi 'XGetBaseName'             DATN
'fso' ⎕wi 'XDeleteFile'   OBJN
'fso' ⎕wi 'XDeleteFolder' OBJN
```

and cover functions for those replaced by calls like

```
⎕NEXISTS CFN
1 ⎕NDELETE XLN
V ← ⎕NPARTS DATN
```

◂ More/other/new[er] system functions ▸ Reading and writing small files ▸ Date arithmetics
▸ Date filed checks ▸ Regular expressions

ERGO
A Munich Re company

Reading and writing small files

APL+Win use of usual sequences like

```
NNR ← ¯1 + ⌊ / 0 , ⎕nnums , ⎕xnnums
DATN ⎕xntie NNR
→ (0 = (V ← ⎕nsize NNR)) / ENDE
DATEN ← ⎕nread (NNR , 82 , V)
⎕nuntie NNR
DATEN ← (  ((1 ↓ II , 0) ∨ (II ← (0 , ¯1 ↓
    ⎕tcnl = DATEN) ∧ (⎕tclf = DATEN)))) ⊂ DATEN
DATEN ← (- + /¨ ∧ \¨ ' ' =¨ ⌽¨ DATEN) ↓¨ DATEN
```

replaced for regular files by calls like

```
PROT ← ↑ 1 ⊃ ⎕NGET (DATN 1)
```

similarly for writing small regular files

◂ More/other/new[er] system functions ▸ Existence and erasure of files ▸ Date arithmetics
▸ Date filed checks ▸ Regular expressions

Date arithmetics

Arithmetic complicated by keeping dates in numerical format YYYYDDMM
APL+Win use of self-written functions like

```
DATUM_VOR ← ← 2 ⊃ 2 DATE ¯1 + (2 ⊃ DAYS DATUM)
```

replaced by

```
DATUM_VOR ← 100 ⊥ 3 ↑ ⊃ 1 ¯1 ⎕DT ¯1
                  + ¯1 1 ⎕DT ⊂ 1E4 1E2 1E2 ⊤ DATUM
```

◂ More/other/new[er] system functions ▸ Existence and erasure of files
▸ Reading and writing small files ▸ Date filed checks ▸ Regular expressions

ERGO
A Munich Re company

Date field checks

Date field checks would be easier/better if ⎕DT accepted ISO formatted dates and/or dates later than 4000-02-28

```
DATE_ISO ← {V ← (⊂ ⍤ (80 = ⎕DR ⍵)) ⍵
            I ← 80 = ⎕DR⍤ V
0 = ≢ (VV ← I / V) : I
            I ← I \ (⊂ ,0)
                ≡⍤ ('\A\d4-\d2-\d2\Z' ⎕S 0)⍤ VV
0 = ≢ (VV ← I / V) : I
            VV ← 2 ⊃⍤ ⎕VFI⍤ ('-' ⎕R ' ') VV
            I \ (0 < ⊃⍤ VV)
                ∧ (0 ⎕DT 1 + 2000 | ¯1 + VV)
          }
```

◂ More/other/new[er] system functions
▸ Existence and erasure of files
▸ Reading and writing small files
▸ Date arithmetics
▸ Regular expressions

ERGO
A Munich Re company

Regular expressions

Regular expressions new way of thinking about patterns

Trivial replacements

```
:For V :in 'SOLL' 'IST' 'SOLL-Duplikate' ...
    SNL ,← ⊂ ('<art>' ⎕R V) CnP.SNT
:EndFor
```

or more complicated, problems with empty arrays / performance

```
:For (V VV) :In ('\A\s*' '') ('\s*\z' '')
('\s+' ' ') ('\s*(?=[\(\)\=\+\-\<\>,])' '')
('[\(\)\=\+\-\<\>]\K\s*' '')
    ⎕EX 'FN' ⋄ FN ← V ⎕R VV
    (I  / ,DATS[;VVV]) ← FN¨ I  / ,DATS[;VVV]
    (II / ,DATI[;VVV]) ← FN¨ II / ,DATI[;VVV]
:EndFor
```

◂ More/other/new[er] system functions ▸ Existence and erasure of files
▸ Reading and writing small files ▸ Date arithmetics ▸ Date filed checks

ERGO
A Munich Re company

Running protocol

```
      ⎕NC ⊂'#.div.udc.MESSAGE_BOX'
9.4
      msgb ← ⎕NEW #.div.udc.MESSAGE_BOX
      msgb.Message ← ''
      msgb.TimeStampMessage 'What''s up?'
      msgb.Message
 2024-08-15 12:02:30  What's up?
      msgb.WaitOn 'Decision' 'Decide!' 3
      ↑ msgb.Message
2024-08-15 12:02:30
What's up?
2024-08-15 12:04:01
Fortfahren durch fehlende Benutzeraktion
                              nach Entscheidung
```

ERGO
A Munich Re company

◂ Proper Classes

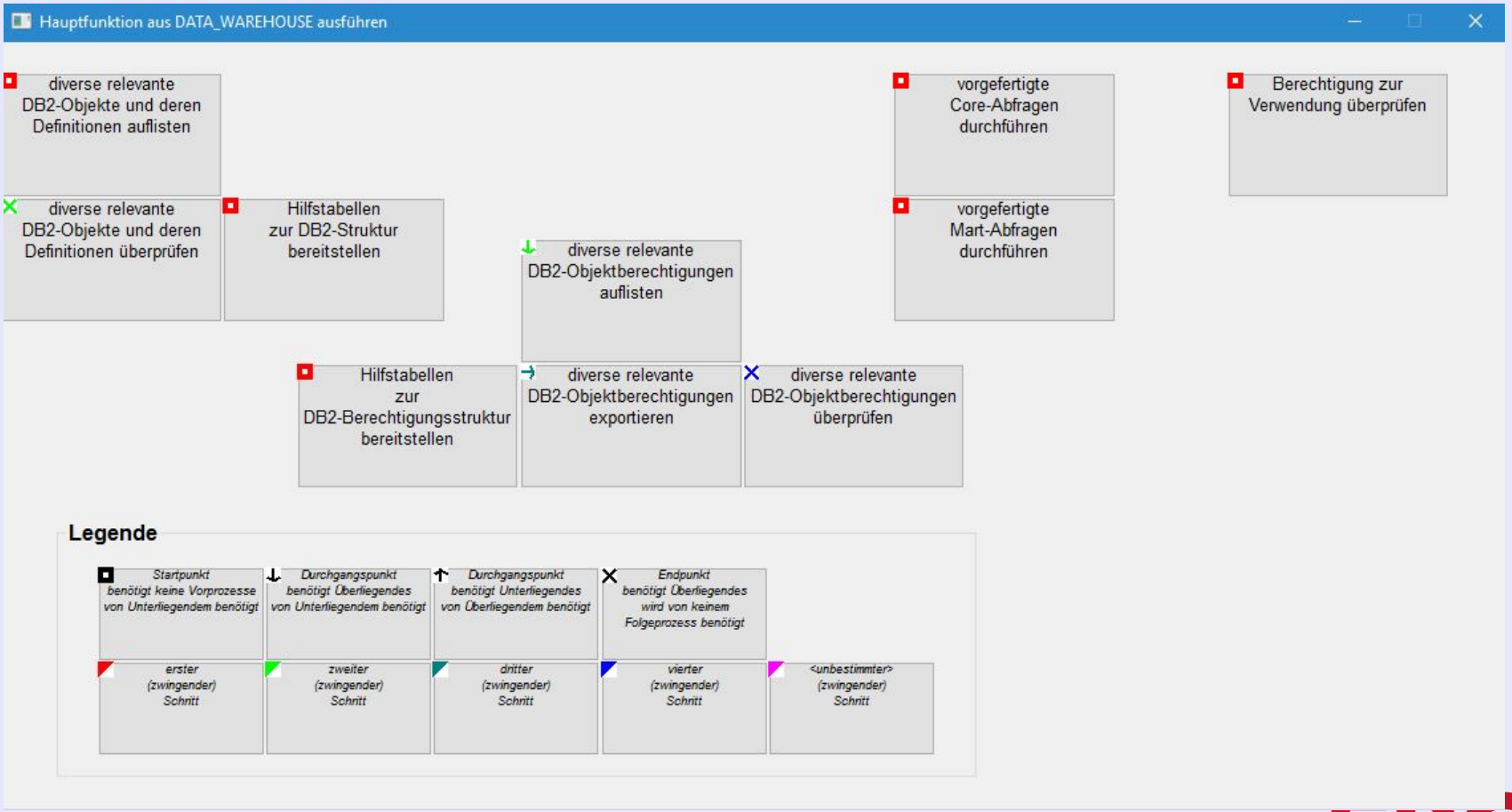
Direct access to namespace

```
obj ← F ← ⎕NEW 'Form' (('Sizeable' 1)
                       ('SysMenu' 1) ('MinButton' 1)
                       ('MaxButton' 1)) ⍝ only here!
obj.(Caption Coord CursorObj Moveable)
                         ← FD.Caption 'RealPixel'
                                                0 1
obj.onClose     ← 1
obj.onConfigure ← 'FD.Configure'
obj.on9999      ← 1

obj ← F.(TF ← ⎕NEW ⊂'TipField')

obj ← F.(SF ← ⎕NEW ⊂'SubForm')
obj.(Posn Size) ← (0 0)  (¯50 0 + F.Size)
```

ERGO
A Munich Re company

◂ Main and parameter GUI ▸ Main GUI ▸ Parameter GUI

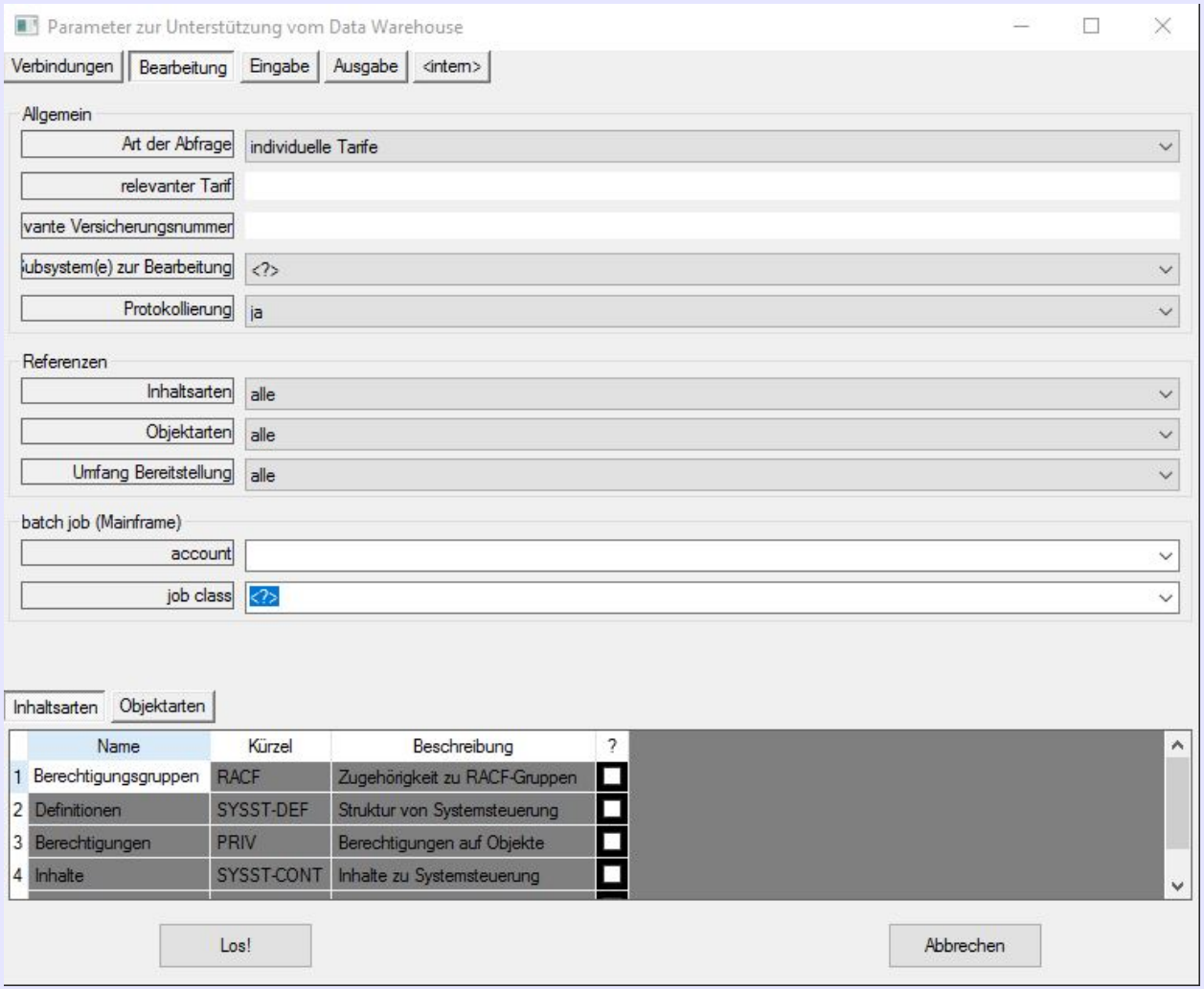
Main GUI



◂ Main and parameter GUI ▸ Direct access to namespace ▸ Parameter GUI

ERGO
A Munich Re company

Parameter GUI



◂ Main and parameter GUI ▸ Direct access to namespace ▸ Main GUI

ERGO
A Munich Re company

COM syntax

In APL+Win (redirectional) syntax cumbersome

```
⎕wi 'wb.xStyles > xl.wb.sts'
⎕wi 'wb.sts.xItem > xl.wb.sts.nrm' 'Normal'
⎕wi 'wb.sts.nrm.xFont > xl.wb.sts.nrm.fnt'
⎕wi 'wb.sts.nrm.fnt.xName' 'Arial'
⎕wi 'wb.sts.nrm.fnt.xSize' 10
```

in Dyalog namespace syntax much easier

```
wb.Styles.Item[⊂'Normal'].Font.(Name Size)
                              ← 'Arial' 10
```

◂ COM as namespace

ERGO
A Munich Re company