



with Bob Cozzi

RPG IV at IBM "i" v7.1

© 2010 by Cozzi Productions, Inc. All rights reserved.

1

Summary of Enhancements



1. SORTA Supports Ascend and Descend Extenders
2. Data Structure Arrays Subfield Support
 - Sort Data Structure Arrays based on Subfields
 - Search Data Structure Arrays by subfield with Lookup Functions
3. Scan and Replace Built-in Function
4. Modification to %LEN
5. Long Externally Described Field Subfield Names (DDS ALIAS name support)
6. Improved Subprocedure Return Value Design
7. Parameter Ordinal Built-in Function
8. Prototypes Are Now Optional When Not Needed
9. More Granular Control on XML-INTO (PTF'd back to v6r1)
10. Teraspace Storage Model
 - RPG IV Storage Model Now Follows that of ILE CL
 - Teraspace Storage Option for %ALLOC Built-in Function via Header Spec ALLOC keyword
11. DBGVIEW source code can now be Encrypted
12. UCS-2 Parameters are Interchangeable with Character Parameters

© 2010 by Cozzi Productions, Inc. All rights reserved.

2

Array Enhancement Examples (2 of 2)

SORTA supports (D) and (A) extenders to sort Ascending or Descending
 Subfields may be specified when sorting an array with SORTA
 Subfields may be specified when searching an array with %LOOKUP



```
// Lookup Data Structure Array Subfield
D myStruct      DS              Dim(50) Qualified Inz
D Item          5A
D Qty          7P 0
D Price        7P 2
D index        S              10I 0
D itemCount    S              10I 0
/free
      index = %lookup('12345' : myStruct(*).item : 1 : itemCount);
/end-free
```

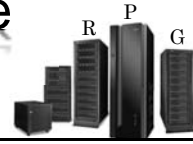
Scan and Replace Built-in Function



- Today introduced as a built-in function
 - %SCANRPL
- Case-sensitive scan/replace only
- Originally provided a decade ago
 - In RPG xTools
 - Now "RPGLIB" at www.RPGLIB.com
 - RPGLIB FindReplace Subprocedure Options:
 - Match Case (allows case-insensitive scan)
 - Find-First only (find/replace first occurrence only)
 - Find from a list of single-characters
 - Find any of "ABCDEFG" in search data.
 - Options are NOT support in %SCANRPL however

%SCANRPL Built-in Function Example

Scans a character string for a pattern.
Replaces the pattern, when found, with the replacement text
Finds/Replaces All Occurrences



```
/free  
    myName = %scanRpl('Bob' : 'Robert' : myName );  
/end-free
```

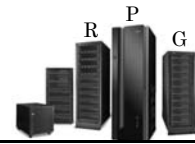
New %LEN Option



- Option to return the declared length of a character field with VARYING keyword
- %LEN(var-char-field [: *MAX])
 - If *MAX not specified
 - Returns the current length
 - if *MAX is specified
 - Returns the "maximum length"
 - AKA, the declared length
 - In other words, the value that %SIZE should be been returning but doesn't.

%LEN *MAX Parameter Example

Returns current length if *MAX not specified.
Returns declared length if *MAX is specified.



```
// %LEN *MAX parameter option
D company          S           50A   Varying

D curLen           S           10I 0
D maxLen           S           10I 0
D size             S           10I 0
/free
    company = 'RPG World';
    curLen = %len(company);           // curLen = 9
    maxLen = %len(company:*MAX);     // maxLen = 50
    size   = %size(company);         // size   = 52
/end-free
```

The smart money is on *MAX returning a compile-time error when a non-VARYING field is used, but we haven't yet tested this situation.

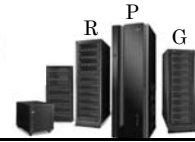
ALIAS Keyword Support for Data Structures



- If the ALIAS keyword is used in DDS files
 - RPG now has support to use those long field names in Data Structures
- Input/Output does not support ALIAS
- Confusing Implementation
- Probably won't be used much
- Highly desired/needed 20+ years ago
- Not so much today

Array Enhancement Examples (1 of 2)

SORTA supports (D) and (A) extenders to sort Ascending or Descending
 Subfields may be specified when sorting an array with SORTA
 Subfields may be specified when searching an array with %LOOKUP



DDS

```

R SALESREC
A      CUSTNO      7P 0      ALIAS(CUSTOMER_NUMBER)
A      CUSTNAME    32A      ALIAS(COMPANY_NAME)
A      MTHSALES    7P 2      ALIAS(MONTHLY_SALES_TOTAL)
A      K CUSTNO
  
```

Example 1

```

// Using the ALIAS keyword to access long field names
// Also see the CUSTSALES DDS source member
FCUSTSALES IF E          K DISK  ALIAS

D Sales1          DS          LikeRec(SaleRec) Inz
  
```

Example 2

```

// Using the ALIAS keyword to access long field names
// Also see the CUSTSALES DDS source member
FCUSTSALES IF E          K DISK  ALIAS PREFIX('S.')

D S              DS          LikeRec(SaleRec) Inz
  
```

Example 3

```

D Sales3          E DS          extName(CUSTSALES)
D                  Qualified Inz ALIAS
  
```

Improved Performance for Subprocedures



- Lengthy character return values
 - Creates a performance issue
- New v7.1 feature RTNPARM keyword
 - Improves performance by "hiding" return value
 - Returned value is passed as a hidden parameter
 - Parameters don't have the same architecture as *return values*

Subprocedure RTNPARM Keyword Example

Passes returned value through a hidden parameter
Greatly improves performance when lengthy parameters are returned



```
// RTNPARM option
D ToLower      PR      32766A  Varying  RTNPARM
D inValue      PR      32766A  Varying

P ToLower      B
D ToLower      PI      32766A  Varying  RTNPARM
D inValue      PI      32766A  Varying
D UPPER        C      'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
D lower        C      'abcdefghijklmnopqrstuvwxyz'
/free
      return %XLATE(upper:lower: inValue);
/end-free
P ToLower      E
```

Normally, lengthy return values cause a performance issue. But adding the new RTNPARM keyword fixes the performance issue.

New %PARMNUM Built-in Function



- Parameter Ordinal -- Heard of it?
- It's the numeric sequence number of a parameter as its declared in a subprocedure
 - Parm 1 is ordinal 1
 - Parm 2 is ordinal 2
 - And so on
- Until now, you had to explicitly identify the parameter number yourself

The %PARMNUM Built-in Function Example

```
// Using the %PARMNUM built-in function
P myProc          B          EXPORT
D myProc          PI          1024A  OPDESC RTNPARM
D firstName       256A       Const Varying
D lastName        256A       Const Varying
D Middle          256A       Const Varying OPTIONS(*NOPASS)
D nLen            S          10I 0
D maxRtnLen       S          10I 0
D dataType        S          10I 0
D curLen          S          10I 0
D maxLen          S          10I 0
/free
>>   if (%parms() >= %PARMNUM(firstName));
>>       ceegsi(%PARMNUM(firstName) : dataType : curLen : maxLen: *OMIT);
>>       rtnName = %subst(firstName:1:curLen);
>>   endif;
>>   if (%parms() >= %PARMNUM(middle));
>>       ceegsi(%PARMNUM(middle) : dataType : curLen : maxLen: *OMIT);
>>       rtnName += ' ' + %subst(Middle:1:curLen);
>>   endif;
>>   if (%parms() >= %PARMNUM(lastName));
>>       ceegsi(%PARMNUM(lastName) : dataType : curLen : maxLen: *OMIT);
>>       rtnName += ' ' + %subst(lastName:1:curLen);
>>   endif;
>>   return rtnName;
/end-free
```

Prototypes are Optional -- Sometimes



- Prototypes requirements have changed
- Required when calling a subprocedure not in "this" source member
- If source member A contains subproc's Q1 and S1
- And source member B contains subproc's P2 and R2
 - If Q1 calls P2, a prototype for P2 is required in A
 - If S1 calls P2, a prototype for P2 is required in A
 - If Q1 calls S1, no prototype for S1 is required in A
 - If P2 calls R2, no prototype for R2 is required in B
 - If R2 calls P2, no prototype for P2 is required in B
 - If S1 calls Q1, no prototype for Q1 is required in A

Where can Prototypes be Omitted?



- New v6r1 feature: MAIN on Header spec
 - Identifies a subprocedure that starts when the program is called
 - Avoids inserting the RPG Cycle into *PGM
- For subprocedures that exist in your source member
 - Regardless of whether or not they are called in "this" source member

XML-INTO Opcode Enhanced



- XML Attributes may be parsed
 - Attributes are within the brackets:
 - `<CUST ID=123>IBM Corp.</CUST>`
 - ID is an attribute
- Multiple XML Nodes may be parsed
- Repeating items may be stored in data structure arrays
- Count of items is magically stored in new user-supplied counter subfield

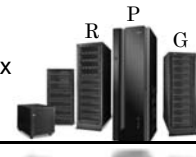
New XML-INTO OPTIONS



- **datasubf Option**
 - Names a subfield that will receive the data of an XML element's attributes
- **countprefix Option**
 - Specifies the prefix of subfield names that receive the count of the number of RPG array elements.
 - This value is calculated by XML-INTO

XML-INTO datasubf Example

datasubf identifies a subfield in the data structure into which the text of the XML tag is stored.



```
// XML-INTO options
D customer          DS              Qualified Inz
D  custno           ←              7P 0
D  compname         ←              50A
```

```
myXML = '<customer custno="5341">IBM Corp.</customer>';
xml-into customer %xml(myXML : 'datasubf=compname');
```

datasubf identifies a subfield in a data structure where the data in the XML tag should be stored. This is necessary because normally, the text would be copied to the "CUSTOMER" variable or subfield. Since the tag CUSTOMER contains both text data and an attribute, you have to use a data structure and identify the subfield into which the text data (i.e., "IBM Corp.") should be copied.

Results

```
customer.custno = 5341
customer.compname = 'IBM Corp.'
```

Summary of Other Features



- Teraspace Storage Model
 - RPG IV Storage Model Follows that of ILE CL
 - STGMDL (* SNGLVL | *TERASPACE | *INHERIT)
 - Teraspace Storage Option for %ALLOC
 - ALLOC(*STGMDL | *TERASPACE | *SNGLVL)
- DBGVIEW source code can be Encrypted
 - DBGVIEW(*LIST) Support Only
 - Use the DBGENCKEY keyword to set password
- UCS-2 Parameters are Interchangeable with Character Parameters on Subprocedures

Rational Open Access: RPG Edition



- This page intentionally left blank



with Bob Cozzi

RPG IV at IBM "i" v7.1 /EOF