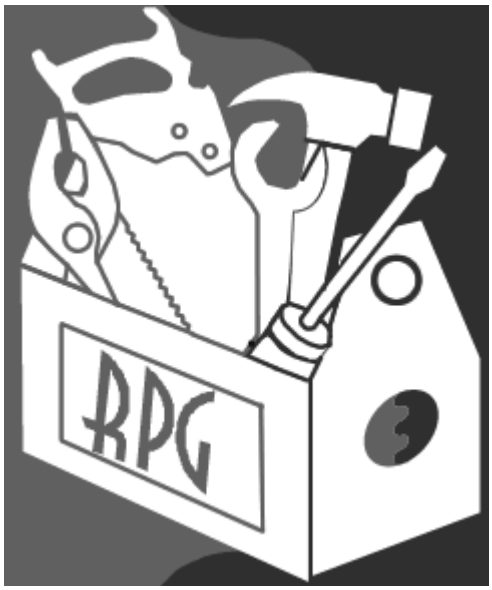
American Top Tools

Información del Distribuidor
Via Laietana 20
08003 Barcelona, Spain
93 319 16 12
www.att.es
email: att@att.es

Linoma Software's

RPG Toolbox

(Version 5.05)



LINOMA SOFTWARE is a division of LINOMA GROUP, Inc.

Publication date: **January 22nd, 2014**

Introduction

(The iSeries, System i and AS/400 platforms will be referred to as IBM i throughout this document)

Linoma's RPG Toolbox will greatly improve the productivity of developers who write and maintain applications on the IBM i. The Toolbox allows you to modernize your RPG programs, develop new applications faster and work with existing source code much more effectively.

The Toolbox was designed and developed by RPG programmers for RPG programmers.

Besides the many RPG specific productivity aids, the Toolbox was designed to also help you work with CL, COBOL, DDS and CMD source code more effectively.

RPG Toolbox Features

The Toolbox contains an abundance of easy-to-use productivity tools for everyday use. The primary capabilities of the Toolbox allow you to:

- Convert RPG III and RPG/400 source code to modernized RPG IV syntax.
- Rejuvenate existing RPG IV source code to take advantage of the most modern syntax available for your IBM i operating system release.
- Convert RPG fixed-format specifications into free-form syntax, with options to convert C specs only, or additionally convert H, F, D and P specs into free-form.
- Use the plugin for Rational Developer for IBM i (RDi) to convert RPG source into free-form and also indent existing free-form nested logic.
- More effectively work within IBM's Source Entry Utility (SEU) using over 70 new line commands provided by the Toolbox's SEU PLUS feature.
- Clean up RPG source code by removing unused work fields, subroutines, prototypes, key lists, parameter lists and tags.

Instructions

Please take the time to read this manual to gain a full understanding of all the Toolbox features available. As you will soon discover, the Toolbox is rich in functionality and can start saving you time right away.

Happy Computing and "LONG LIVE RPG!"

Linoma Software

The Company

Linoma Software is dedicated to providing innovative and useful products for the IBM i platform. Our goal is to help IBM i users take advantage of modern technologies and become more productive in their everyday work.

Since our start in 1994, we have built a large base of satisfied clients around the world. You are encouraged to review the many positive testimonials and product reviews on our web site. Customer references can also be supplied upon request.

Linoma's success has been built on being very responsive to our customer's requirements. So if you have any suggestions on how we can improve our products to make your job easier, please let us know.

We would like to hear from you

For comments, questions or to purchase a license to the Toolbox, you can contact us at:

Electronic

Sales:	sales@linoma.com
Support:	support@linoma.com
Web site:	www.linomasoftware.com

Phone numbers

Toll-free:	1-800-949-4696
Outside USA:	(402) 944-4242
Fax:	(402) 944-4243

Address

LINOMA SOFTWARE
1409 Silver Street
Ashland, NE 68003 USA

Trial Period

The trial version of the Toolbox will allow your organization to evaluate its value within your own environment. All the Toolbox features are available for use during the trial period, but with the following restrictions:

- The Toolbox expires in 30 days from the time of first use.
- Up to 10 RPG source members can be converted.
- RPG source members with more than 5000 records cannot be converted.
- Multiple RPG source members cannot be converted at one time.

The above restrictions are removed when a permanent license is purchased.

If you exceed the trial limits and need a temporary extension, please contact us.

Purchasing a License

The RPG Toolbox is licensed per IBM i partition. For pricing information, please visit our web site at www.linomasoftware.com or call us at (402) 944-4242 or 1-800-949-4696 (in the US).

How to Order

You will need to provide the following information when placing an order for the Toolbox:

- IBM i serial number
- Processor group (can be found by running the WRKLICINF command)
- Your Name
- Organization
- Address
- Country
- E-mail address
- Voice phone number
- Fax number

Use one of the following methods to place your order:

- **Internet:** Visit www.linomasoftware.com and place your order on-line using a credit card.
- **Fax:** Purchase orders can be faxed to (402) 944-4243.
- **Phone:** Call (402) 944-4242 or 1-800-949-4696 (in the US) and order using a credit card.
- **Mail:** Send the proper payment amount to Linoma Software, 1409 Silver St., Ashland NE 68003
- **Wire Transfer:** Bank wire transfers are also accepted. Call us for details.

Upon receipt of a valid payment, we will e-mail or fax you the permanent license key(s).

Please note that you may not use this Toolbox to modernize any RPG source code owned by another organization unless that organization has also purchased a Toolbox license.

Pre-Installation Notes

The Toolbox software restores onto the IBM i as a Licensed Program named 4RPGBOX. After you restore this licensed program, the software will be contained in the library named RPGTOOLBOX.

Requirements

To install the Toolbox, the following requirements must be met:

- IBM i release V5R1 or greater installed on your IBM i.
- Your user id must have authority to the RSTLICPGM command.
- If you need to use the Toolbox feature which converts RPG III or RPG/400 source code to RPG IV, you must have IBM's ILE RPG compiler installed on your IBM i (since IBM's CVTRPGSRC command is used during the preliminary conversion process).

RPG Toolbox Upgrades

Please read this section if you are upgrading from a prior release of the Toolbox.

IMPORTANT: A new product license key is required when upgrading from version 4.06 or lower of the RPG Toolbox. To view your current installed version, run the command DSPDTAARA RPGTOOLBOX/VERSION. Please contact Linoma Software at sales@linomasoftware.com to request a new product license key.

Warnings

You should not delete the existing Toolbox licensed program or library before upgrading, otherwise user-created data (i.e. custom SEU line commands) will be lost.

The SNIPPETS source file in the RPGTOOLBOX library will be replaced during an upgrade. If you have created or modified any source members within that source file, please backup those source members before the upgrade. You should always refrain from modifying or creating source members directly in the RPGTOOLBOX/SNIPPETS source file.

Check for Locks

Before installation, make sure there are no locks on the file RPGTOOLBOX/BXP030. If this file is locked, then inform the user(s) with the locks to either:

1) Sign Off or
2) From within SEU, run the Toolbox line command of RESET.

Retention of User Data

If you already have a permanent license key to the Toolbox, the key will be retained during an upgrade.

The following user-defined data will be retained through the Toolbox upgrade:

- User-defined SEU line commands.
- User-defined Snippet Index Records.
- User-defined Snippet Source Types and Categories used for searching.
- Any defaults which were specified on the CHGDFT (Change RPG Wizard defaults) command.

During the installation process, a copy of the existing Toolbox data will be saved into a library called RPGTBxxxxx, where xxxxx is a sequential number starting with 00001. This library will only be needed if an upgrade fails, in which case you should contact Linoma Software.

Follow the instructions on the next page to install the new version of *RPG Toolbox*.

Installation

Important: Make sure you have read the "Pre-Installation Notes" listed on the prior page if upgrading from a prior version of *RPG Toolbox*.

The *RPG Toolbox* software is contained in an IBM i Save file, which needs to be uploaded to your system through FTP. Perform the following steps to install (or upgrade) the *RPG Toolbox* product onto the system.

1. Create a temporary Save file on your system by executing the following IBM i command:
   CRTSAVF FILE(QGPL/RPGBOXSF)

2. Before proceeding, make sure the FTP server is running on the system. To start the FTP server on the system, execute the IBM i command:
   STRTCPSVR SERVER(*FTP)

3. FTP the file RPGBOX.SAVF from your PC to the Save file created in step 1. Listed below are instructions for a Windows user:
   - Open a DOS window.
   - Enter the DOS command **FTP <hostname>**, where <hostname> is the host name or IP address of your IBM i.
   - Login with your IBM i user id and password, then enter the following highlighted FTP commands on your PC:

ftp> **BINARY**	(switches the FTP session to binary mode)
ftp> **LCD \<tempdir>**	(<tempdir> is the PC directory containing the file RPGBOX.SAVF)
ftp> **CD qgpl**	(changes the remote directory to the QGPL library)
ftp> **PUT rpgbox.savf rpgboxsf**	(sends the PC file RPGBOX.SAVF to the IBM i file RPGBOXSF)
ftp> **QUIT**	(ends your FTP session)

4. Restore the licensed program from the Save file by running the IBM i command:
   RSTLICPGM LICPGM(4RPGBOX) DEV(*SAVF) SAVF(QGPL/RPGBOXSF)

5. Delete the temporary Save file by running the IBM i command:
   DLTF FILE(QGPL/RPGBOXSF)

6. Continue with the section labeled "Creating PDM Options".

Menu of Commands

All of the commands supplied within the Toolbox are accessible from the following menu. To access this menu, run the IBM i command of:

GO RPGTOOLBOX/MENU

```
                          Linoma's RPG Toolbox

 Select one of the following:

 Configuration
    1. Create PDM Options              (CRTOPT)
    2. Change RPG Wizard Defaults      (CHGDFT)
    3. Enable SEU PLUS System-Wide     (ADDEXITPGM)
    4. Add License Key                 (ADDKEY)
    5. Contact Linoma Software
 Tools
   10. RPG Wizard                      (RPGWIZ)
   11. Display Indented Source         (DSPIND)
   12. Document Nested Logic           (DOCNST)
   13. Indent Nested Free-Form Logic   (INDNST)
   14. Highlight Comment Lines         (HLTCMT)
   15. Clean RPG Source                (CLNRPG)

 Selection or command
===>_____________________________________________________________________________
_____________________________________________________________________________

 F3=Exit   F4=Prompt   F9=Retrieve   F12=Cancel
 F13=Information Assistant  F16=AS/400 main menu
```

To run the "Tools" commands (menu options 10 through 15) quickly over source members, it is recommended to create PDM option shortcuts.

Creating PDM Options

This step allows you to create PDM options for the Toolbox commands which act upon source members. Using PDM options is easier than having to key in the command each time. You can simply place the desired PDM option next to the source member process, then the member name, file and library will automatically fill in for you.

If PDM options already exist for the values you specify, they will be replaced with the new values. All other PDM options will not be affected.

You can create PDM options by first prompting (F4) the command **RPGTOOLBOX/CRTOPT**, which is shown below, then make any changes and press Enter:

```
                  Create RPG Toolbox PDM Options (CRTOPT)

Type choices, press Enter.

PDM options file . . . . . .   QAUOOPT      Name, QAUOOPT
  Library  . . . . . . . . .     QGPL       Name, QGPL, *LIBL
PDM options member . . . . . .  QAUOOPT      Name
RPGIV Wizard - Fixed Format  . .  RX         Character value, RX
RPGIV Wizard - Free Format . . .  RF         Character value, RF
Display Indented Source  . . . .  DI         Character value, DI
Document Nested Logic  . . . . .  DN         Character value, DN
Indent Nested Free-Form Logic  .  IN         Character value, IN
Highlight Comment Lines  . . . .  HC         Character value, HC
Clean RPG Source . . . . . .   CR            Character value, CR
```

PDM options file (FILE)

Specifies the name of the file and library. QAUOOPT is typically the default PDM options file.

PDM options member (MBR)

Specifies the name of the file member. QAUOOPT is typically the default PDM options member.

RPGIV Wizard – Fixed Format PDM option (OPTION1)

Specifies the PDM option to use for the RPG Wizard (RPGWIZ) command when converting to Fixed-Format specifications. The default is RX.

RPGIV Wizard – Free Format PDM option (OPTION2)

Specifies the PDM option to use for the RPG Wizard (RPGWIZ) command when converting to Free-Form specifications. The default is RF.

Display Indented Source PDM option (OPTION3)

Specifies the PDM option to use for the Display Indented Source (DSPIND) command.
The default is DI.

Document Nested Logic PDM option (OPTION4)

Specifies the PDM option to use for the Document Nested Logic (DOCNST) command.
The default is DN.

Indent Nested Logic PDM option (OPTION5)

Specifies the PDM option to use for the Indent Nested Logic (INDNST) command.
The default is IN.

Highlight Comment Lines PDM option (OPTION6)

Specifies the PDM option to use for the Highlight Comment Lines (HLTCMT) command.
The default is HC.

Clean RPG Source PDM option (OPTION7)

Specifies the PDM option to use for the Clean RPG Source (CLNRPG) command.
The default is CR.

Applying a Permanent License Key

If you receive a permanent license key for the RPG Toolbox, you can apply it by prompting the command **RPGTOOLBOX/ADDKEY.**

ADDKEY screen:

```
                  Add License Key - RPG Toolbox (ADDKEY)

Type choices, press Enter.

License key:
  Characters 1 - 6 . . . . . .   ______        Character value
  Characters 7 - 12  . . . . . .  ======        Character value
  Characters 13 - 18 . . . . . .  ======        Character value
Processor group  . . . . . . . .  _____         *ANY
Feature  . . . . . . . . . . . .  *BASE        *BASE, *RDI
Usage limit  . . . . . . . . . .  *NOMAX       0-999999, *NOMAX
```

Enter the following information on the screen:

- License key
- Your IBM i Processor group
- Feature (either *BASE or *RDI)

Leave the Usage limit at *NOMAX and the Expiration date at *NONE.

RPG Wizard

Introduction

Linoma's RPG Wizard will modernize your RPG source code with the most up-to-date syntax and style available for your IBM i operating system release. The RPG Wizard is included in Linoma's RPG Toolbox and is referred to as RPGWIZ throughout the remainder of this document.

There are several types of conversions which can be performed by RPGWIZ:

- Convert RPG III and RPG/400 source code to modernized RPG IV syntax.
- Update existing RPG IV source code to take advantage of the most modern syntax available for your IBM i release.
- Convert RPG IV fixed-format Calculation specifications to the free-form syntax.
- Optionally convert H, F, D and P specifications to the free-form syntax.

RPGWIZ makes it easy to learn RPG IV and free form

While RPGWIZ is very effective at RPG conversion, it also can serve as a valuable learning aid for programmers wanting to take advantage of the latest syntax and methods available in RPG IV, including the free-form syntax.

Simply convert a program you're familiar with and watch how the old operations are converted into the new syntax. There is no better way to learn than "by example".

Features

RPGWIZ's extensive modernization features are listed below in summarized format. Most of these features can be switched on/off during a conversion, allowing you to customize the modernization process.

Definition Specifications
- Redefine data structure fields; list fields in relative position order, convert from/to positions to field lengths, add OVERLAY keywords and indent sub-fields.
- Move Calculation-defined fields and lengths to the Definition specs.
- Move *LIKE defined fields from the Calculation specs to the Definition specs.
- Move *ENTRY parameters from the Calculation specs into the Definition specs.
- Create data structures from KLIST fields and change file I/O operations to use the %KDS (search arguments in data structure) BIF.

Assignment Operations
- Convert ADD, SUB, Z-ADD, Z-SUB, MULT and DIV operations to EVALs or free form.
- Convert MOVE and MOVEL operations to EVALs or free form.
- Optionally convert MOVE(L) of *BLANKs or *ZEROs to CLEAR operations.
- Convert MOVEA (move array) operations.
- Convert CAT operations to EVALs or free form.
- Convert SETON, SETOF and COMP operations to EVALs or free form.
- Convert the constants of '1' to *ON and '0' to *OFF in indicator operations.

Built-in functions (BIFs)
- Convert traditional operations to their corresponding BIFs. For example, SUBST becomes %SUBST.
- Insert file I/O BIFs, such as %FOUND and %EOF, under file operations.

Control Operations
- Move IFxx, DOxxx, WHxx, ANDxx and ORxx expressions to extended factor 2.
- Convert eligible GOTO operations
- Convert DO operations to FOR operations.
- Qualify END operations. For example, an END under an IF becomes an ENDIF.
- Convert most conditioning indicators (left hand) to IF operations.
- Document the beginning and ending of IF, DO, DOW, DOU, SELECT, CAS and FOR structures.

CALLs and CALLB Operations
- Convert CALL and CALLB operations to CALLP operations.
- Create prototypes for new CALLP operations.

Subroutines
- Convert CASxx operations to SELECT/WHEN or IF/ELSE operations.
- Convert subroutines to sub-procedures.

Free format
- Reformat fixed-format operations into free format style, with options to convert C, H, F, D and P specifications.
- Add semicolon delimiters to the end of free format operations.
- Convert asterisk (*) comment designators to slashes (//) comment designators.
- Indent nested logic.

Compression
- Compress expressions to fit into the minimum number of source lines needed.

Comments
- Remove specification types from comment lines.
- Highlight comment lines.
- Convert asterisk (*) comment designators to slashes (//) comment designators (available in V5R1).
- Remove comment designators (* or //) from blank comment lines.

Case Conversion
- Convert source logic to lower, upper or mixed case.
- Convert spec types to lower or upper case.
- Convert in-line source comments to either lower or upper case.
- Convert right-hand source comments to either lower or upper case.

Change Control
- Generate a summary report of converted source member(s), with any exception messages.
- Set the source date on modified lines to the current date or to zeros.

Modernization Examples

Fixed-format Conversion Example

The follow example demonstrates how RPGWIZ can convert RPG III source into fixed format RPG IV. In this example, notice how the subroutines were also converted to sub-procedures.

Before Conversion:

```
I..................................PFromTo++DFldnmeL1M1FrPlMnZr.
I            DS
I                                        1   7 FIRST6
I                                        4   6 PREFIX
I                                        1   3 AREACD
I                                        1  10 PHONE

CL0N01N02N03Factor1+++OpcdeFactor2+++ResultLenDHHiLoEq
C           *LIKE     DEFN AREACD    WAREA
C* Retrieve customer record
C           CNBR      CHAINCUSTMAS              99
C* Prepare values
C                     MOVE *ZEROS    XX
C  N99                MOVE AREACD    WAREA
C  N99                EXSR PROCESS
C* Calculate
C           PROCESS   BEGSR
C                     ADD  1         XX      30
C           WAREA     IFNE *BLANK
C           XX        MULT 100       NUMBER
C                     SETON                     61
C                     END
C                     ENDSR
```

After converting to RPG IV fixed format:

```
DName++++++++++ETDsFrom+++To/L+++IDc.Keywords++++++++++++++++++++++++++
D                 DS
D PHONE                          10
D  FIRST6                         7    OVERLAY(PHONE)
D   AREACD                        3    OVERLAY(FIRST6)
D   PREFIX                        3    OVERLAY(FIRST6:4)
 * Work fields
D WAREA           S                    LIKE(AREACD)
D XX              S               3  0
 * Prototypes
D PROCESS       PR

CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
 * Retrieve customer record
C     CNBR          CHAIN     CUSTMAS
C                   EVAL      *IN99 = NOT%FOUND(CUSTMAS)
 * Prepare values
C                   EVAL      XX = *ZEROS
C                   IF        *IN99 = *OFF
C                   EVAL      WAREA = AREACD
C                   CALLP     PROCESS
C                   ENDIF
 * Calculate
P PROCESS           B
C                   EVAL      XX = XX + 1
C                   IF        WAREA <> *BLANK
C                   EVAL      NUMBER = XX * 100
C                   EVAL      *IN61 = *ON
C                   ENDIF
P PROCESS           E
```

Free-format Conversion Example (C specs only)

The follow example demonstrates how RPGWIZ can convert RPG III source into free-format RPG IV. In this example, only the C specifications are converted to free form.

Before Conversion:

```
I..................................PFromTo++DFldnmeL1M1FrPlMnZr.
I            DS
I                                       1   7 FIRST6
I                                       4   6 PREFIX
I                                       1   3 AREACD
I                                       1  10 PHONE

CL0N01N02N03Factor1+++OpcdeFactor2+++ResultLenDHHiLoEq
C           *LIKE     DEFN AREACD    WAREA
C* Retrieve customer record
C           CNBR      CHAINCUSTMAS              99
C* Prepare values
C                     MOVE *ZEROS    XX
C  N99                MOVE AREACD    WAREA
C  N99                EXSR PROCESS
C* Calculate
C           PROCESS   BEGSR
C                     ADD  1         XX      30
C           WAREA     IFNE *BLANK
C           XX        MULT 100       NUMBER
C                     SETON                     61
C                     END
C                     ENDSR
```

After converting to Free Form:

```
DName++++++++++ETDsFrom+++To/L+++IDc.Keywords++++++++++++++++++++++++++
D                 DS
D PHONE                          10
D  FIRST6                         7    OVERLAY(PHONE)
D   AREACD                        3    OVERLAY(FIRST6)
D   PREFIX                        3    OVERLAY(FIRST6:4)
 //
D WAREA           S                    LIKE(AREACD)
D XX              S               3  0
 /FREE
  // Retrieve customer record
  CHAIN CNBR CUSTMAS;
  *IN99 = NOT%FOUND(CUSTMAS);
  // Prepare values
  XX = *ZEROS;
  IF *IN99 = *OFF;
      WAREA = AREACD;
      EXSR PROCESS;
  ENDIF;
  // Calculate
  BEGSR PROCESS;
      XX = XX + 1;
      IF WAREA <> *BLANK;
         NUMBER = XX * 100;
         *IN61 = *ON;
      ENDIF;
  ENDSR;
 /END-FREE
```

Free-format Conversion Example - Totally Free

The follow example demonstrates how RPGWIZ can convert RPG fixed-format source into totally free form style, which is available with IBM i V7R1 technology release 7. Notice how the H, F, D, C and P specifications are all converted to free form.

Before conversion:

```
H* Control Specifications
HDATFMT(*USA) COPYRIGHT('LINOMA SOFTWARE')

F* File Specifications
FCDLCM1    IF A E           K DISK    RENAME(CDRCM:CDRCM2)

D* Data Structure
DPhoneDS          DS
D AreaCode                1      3
D Prefix                  4      6
D Last4                   7     10

D* Stand-alone fields
D TotalTax        S              9  2
D INVTOTAL        S              7  2

C     *LIKE         DEFINE    CMCRLM        CRLIMIT
C* Chain to Customer file
C                   MOVE      '000001'      CMIDNO
C     CMIDNO        CHAIN     CDLCM1                              99
C* Found-> Process credit limit
C   99              MOVE      CMCRLM        CRLIMIT
C   99              EXSR      PROCESS
C* Return to caller
C                   SETON                                         LR
C                   RETURN
C* Process credit limit
C     PROCESS       BEGSR
C     CRLIMIT       IFGT      10000
C                   SETON                                         61
C                   MOVEL     'ABC COMPANY' CMNAME
C     CRLIMIT       SUB       INVTOTAL      REMAIN            9 2
C                   END
C                   ENDSR
```

(continued on next page)

After converting to Free Form:

```
// Control Specifications
CTL-OPT DATFMT(*USA) COPYRIGHT('LINOMA SOFTWARE');

// File Specifications
DCL-F CDLCM1 DISK(*EXT) USAGE(*INPUT : *OUTPUT)
      KEYED RENAME(CDRCM:CDRCM2);

// Data Structure
DCL-DS PhoneDS;
  AreaCode CHAR(3);
  Prefix CHAR(3);
  Last4 CHAR(4);
END-DS;

// Stand-alone fields
DCL-S TotalTax PACKED(9 : 2);
DCL-S INVTOTAL PACKED(7 : 2);
DCL-S CRLIMIT LIKE(CMCRLM);
DCL-S REMAIN PACKED(9 : 2);
// Prototype for PROCESS
DCL-PR PROCESS;
END-PR;

// Chain to Customer file
CMIDNO = '000001';
CHAIN CMIDNO CDLCM1;
*IN99 = NOT %FOUND;
// Found-> Process credit limit
IF *IN99 = *ON;
    CRLIMIT = CMCRLM;
    PROCESS();
ENDIF;
// Return to caller
*INLR = *ON;
RETURN;
// Process credit limit
DCL-PROC PROCESS;
  IF CRLIMIT > 10000;
    *IN61 = *ON;
    %SUBST(CMNAME:1:11) = 'ABC COMPANY';
    REMAIN = CRLIMIT - INVTOTAL;
  ENDIF;
END-PROC PROCESS;
```

Using the RPG Wizard

RPGWIZ can be launched through the following methods:

1. From Rational Developer for IBM i (RDi) using the optional plugin.
2. By placing a PDM option next to the source member.
3. From the command line.
4. From within SEU, using SEUPLUS line commands (included with the Toolbox).

RDi execution

With the optional RDi plugin from Linoma Software, you can convert an entire source member or a selected block of lines using a right-click menu option. Read the section labeled "RPG Toolbox RDi Plugin" to learn how to install and use the plugin.

PDM execution

To convert all the lines in a source member, one of the easiest methods is to use PDM options.

Go into PDM and place one of the RPGWIZ options next to the source member you wish to convert. Then press F4 to Prompt or press Enter to execute with the default settings.

These PDM options are created with the CRTOPT command, which was explained in the installation instructions. The default option for conversion to fixed-format is RX and the default option for conversion to free-format is RF.

Command Line execution

To convert multiple source members at a time or to perform the conversion in batch, execute RPGWIZ from the command line or from within a CL program.

Key the command **RPGTOOLBOX/RPGWIZ** and then press F4 to Prompt.

SEUPLUS execution

To just convert a single line or a block of lines within a source member, use one of the SEUPLUS line commands from within SEU. Those line commands are:

Z - RPG Wizard on Line (no prompt)
Converts a line using the default RPGWIZ settings.

ZZ - RPG Wizard on Block (no prompt)
Converts a block of lines using the default RPGWIZ settings. Key the ZZ line command at the start of the block and key it again at the end of the block to convert, then press Enter.

ZP - RPG Wizard on Line (prompt)
Converts a line. You will first be prompted for the RPGWIZ settings.

ZZP - RPG Wizard on Block (prompt)
Converts a block of lines. You will first be prompted for the RPGWIZ settings. Key the ZZP line command at the start of the block and key it again at the end of the block to convert, then press Enter.

Recommendations for RPGWIZ

Backup and Testing

Always keep a copy of your original source members for backup purposes.

It is your responsibility to review, compile and test all converted programs.

Source Files

When creating source files to store RPG IV source members, it is recommended to use the IBM standard name of QRPGLESRC.

RPG IV source files should also be created with a record length of 112. Otherwise the right hand comments in the source code are lost if you use the default length of 92.

Converting from RPG III or RPG/400

When converting from RPGIII or RPG/400 with the RPGWIZ command, IBM's CVTRPGSRC command will automatically be run to perform the preliminary conversion. CVTRPGSRC performs functions such as moving the RPG logic into the proper columns, renaming some of the operation codes (i.e. SELEC is renamed to SELECT) and converting most of the Input specifications to the Definition specifications.

IBM's CVTRPGSRC command also generates a report, which you should review when a conversion problem is indicated. See appendix "B" in IBM's "ILE RPG for AS/400 Programmers Guide" for a better understanding of IBM's CVTRPGSRC command.

Converting RPT and RPT38 Source Members

When converting an auto report source member (type RPT or RPT38), any /COPY members will always be expanded into the converted source, even if you specify *NO on the "Expand copy members" parameter. This is because auto report is not supported by RPG IV and specifically does not support mixed specification types within /COPY members.

For auto report source members, it is recommended that you first change the source member type to RPG before the conversion, which will not expand the /COPY members. For any /COPY members that contain mixed specification types, those members will need to be broken out by specification type after the conversion.

Additional Notes

RPGWIZ will not automatically recompile your converted source code.

RPGWIZ will always generate a summary report which indicates the source member(s) converted, with any exception messages.

RPGWIZ Command Parameters

When you prompt the RPGWIZ command, the following parameters will be displayed:

```
                          RPG Wizard by Linoma (RPGWIZ)

Type choices, press Enter.

From file  . . . . . .   QRPGLESRC_    Name, QRPGSRC, QRPGLESRC
  Library  . . . . . .     *LIBL_____  Name, *LIBL
From member  . . . . .   __________    Name, generic*, *ALL
From type  . . . . . .   *RPG4_____    RPG, RPT, RPG38, RPT38...
To file  . . . . . .     QRPGLESRC_    Name, QRPGLESRC
  Library  . . . . . .     *FROMLIB__  Name, *FROMLIB, *LIBL
To member  . . . . . .   *FROMMBR__    Name, *FROMMBR
Replace existing To member(s)  .   *NO_          *NO, *YES
Target IBM i release . . . . . .   *CURRENT      *CURRENT, VxRx
Format of specifications . . . .   *FIXED        *FREE, *FREE2, *FIXED
Examine field attributes . . . .   *YES          *YES, *NO
Expand copy members  . . . . . .   *NO_          *NO, *YES
Redefine data structures . . . .   *YES          *NO, *YES
Redefine *LIKE DEFN fields . . .   *YES          *NO, *YES
Redefine calc. defined fields  .   *YES          *NO, *YES
Convert left hand indicators . .   *YES          *NO, *YES
Convert opcodes to BIFs  . . . .   *YES          *NO, *YES
Convert key lists (KLIST)  . . .   *NO           *NO, *YES, *YES2
Insert file I/O BIFs . . . . . .   *NO_          *NO, *YES
Convert ADDs/SUBs to EVALs . . .   *YES_         *NO, *YES, *YES2
Convert Z-ADDs/Z-SUBs to EVALs .   *YES_         *NO, *YES, *YES2
Convert MULTs to EVALs . . . . .   *YES_         *NO, *YES, *YES2
Convert DIVs to EVALs  . . . . .   *YES_         *NO, *YES, *YES2
Convert MOVE(L)s having *BLANK .   *EVAL_        *NO, *EVAL, *CLEAR
Convert MOVE(L)s having *ZERO  .   *EVAL_        *NO, *EVAL, *CLEAR
Convert MOVEs having data  . . .   *EVAL_        *NO, *EVAL
Convert MOVELs having data . . .   *EVAL_        *NO, *EVAL
Convert MOVEA operations . . . .   *NO           *NO, *YES
Convert CASxx operations . . . .   *YES          *NO, *YES
Convert CATs operations. . . . .   *YES_         *NO, *YES, *YES2
Convert DOs to FORs  . . . . . .   *YES          *NO, *YES
Convert LOOKUP operations  . . .   *NO           *NO, *YES
Convert SCAN operations  . . . .   *NO           *NO, *YES
Convert *ENTRY PLIST   . . . . .   *NO_          *NO, *YES, *YES2
Convert Subroutines to Procs . .   *NO_          *NO, *YES
Convert CALLs and CALLBs . . . .   *NO_          *NO, *YES, *YES2
Convert GOTO operations  . . . .   *YES          *NO, *YES
Compress expressions . . . . . .   *NO           *NO, *YES
Fixed-form comment designator  .   *LEAVE__     *LEAVE, *ASTERISK, *SLASHES
Highlight comments . . . . . .     *YES          *YES, *NO
Comment specification types  . .   *REMOVE_     *LEAVE, *REMOVE
Comment designator on blanks . .   *LEAVE        *LEAVE, *REMOVE
Case for specification types . .   *LEAVE_       *LEAVE, *LOWER, *UPPER
Case for unchanged logic . . . .   *LEAVE_       *LEAVE, *LOWER, *UPPER...
Case for changed and new logic     *UPPER_       *LOWER, *UPPER, *MIXED...
Case for in-line comments  . . .   *LEAVE_       *LEAVE, *LOWER, *UPPER
Case for right-hand comments . .   *LEAVE_       *LEAVE, *LOWER, *UPPER
Document nested logic  . . . . .   *NO           *NO, *YES
Free-form indent nested logic  .   *LEAVE__     *LEAVE, *INDENT0, *INDENT1...
Source date on converted lines .   *KEEP_       *KEEP, *TODAY, *ZEROS
Save command settings for job  .   *NO_         *NO, *YES
```

For a complete description of the parameters, press F1 on the screen or read the following pages.

Default Values

RPGWIZ is installed with the safest parameter values as the default. By using these default values, the functionality of a converted program should not be compromised even though the syntax is modernized.

Review this document to understand the considerations when enabling any additional parameters.

From file (FROMFILE)

Specifies the name of the source file that contains the source code to be converted and the library where the source file is stored.

source-file-name
Enter the name of the source file that contains the source member(s) to be converted.

The possible library values are

***LIBL**
The system searches the library list to find the library where the source file is stored.

library-name
Enter the name of the library where the source file is stored.

From member (FROMMBR)

Specifies the name(s) of the source member(s) to be converted.

source-file-member-name
Enter the name of the source member to be converted.

***ALL**
The command converts all the members in the source file specified.

generic*-source-file-member-name
Enter the generic name of members having the same prefix in their names followed by an * (asterisk). The command converts all the members having the generic name in the source file specified. For example, specifying FROMMBR(PR*) will result in the conversion of all members whose names begin with 'PR'.

From type (FROMTYPE)

Specifies the source type of the source member(s) to be converted.

source-member-type
Enter the type of the source member(s) to be converted. Valid values include RPG, RPT, RPG38, RPT38, SQLRPG, RPGLE or SQLRPGLE.

***RPG3**
The source type is RPG, RPT, RPG38, RPT38 or SQLRPG.

***RPG4**
The source type is RPGLE or SQLRPGLE.

To file (TOFILE)

Specifies the name of the source file that contains the converted source members and the library where the source file is stored. The source file must exist and should have a record length of 112 characters: 12 for the sequence number and date, 80 for the logic and 20 for the comments.

QRPGLESRC
The default source file QRPGLESRC will contain the converted source member(s).

***FROMFILE**
The source file name specified in the "From file" will contain the converted source member(s).

source-file-name
Enter the name of the source file that will contain the converted source member(s).

The possible library values are

***FROMLIB**
The system uses the library specified in the 'From file' to find the library where the source file is stored.

***LIBL**
The system searches the library list to find the library where the source file is stored.

library-name
Enter the name of the library where the source file is stored.

To member (TOMBR)

Specifies the name(s) of the converted source member(s) in the converted source file. If the value specified on the FROMMBR parameter is (*ALL) or generic*, then TOMBR must be equal to *FROMMBR.

***FROMMBR**
The From member name(s) are used as the converted source member name(s).

source-file-member-name
Enter the name of the converted source member.

Replace existing To member(s) (REPLACE)

Specifies whether or not to replace the existing To member(s), if they exist.

***NO**
No "To" member(s) will be replaced if they exist.

***YES**
All "To" member(s) will be replaced if they exist.

Target IBM i release (TGTRLS)

Specifies the IBM i release in which the conversion will be targeted. Only the syntax, operations and built-in-functions (BIFs) available up to that target release will be available for conversion.

target-release
Enter the target release in VxRy format, where x is the version and y is release. For instance, enter V7R1 for a target release of IBM i version 7 Release 1.

***CURRENT**
The current release for the IBM i you are executing on will be the target release.

Format of calculation specs (FMTCALC)

Specifies the target format for the converted calculation specifications.

***FIXED**
No specifications will be converted to free-form syntax.

***FREE**
All eligible fixed-format calculation operations will be converted to their free-form equivalents. This option is only valid for IBM i release V5R1 and higher.

If an error result indicator is specified on an ACQ, CLOSE, COMMIT, DEALLOC, DSPLY, FEOD, IN, NEXT, OPEN, OUT, POST, REL, RESET, ROLBK, TEST or UNLOCK operation, this operation will be qualified with an (E) extender and the error indicator will be converted into an inserted %ERROR BIF.

Comment line asterisks (*) will be converted to slashes (//) within the new free-form logic. Any existing right-hand comments will be proceeded with slashes(//) notation starting in position 81.

The appropriate /FREE and /END-FREE tags will be inserted around the new free-form logic.

Operations without free-form equivalents will remain in fixed format. As of IBM i release V5R1, these unsupported opcodes include ADD, ADDDUR, ALLOC, ANDxx, BITxx, CABxx, CALL, CALLB, CASxx, CAT, CHECK, CHECKR, COMP, DEFINE, DIV, DO, DOUxx, DOWxx, END, EXTRCT, GOTO, IFxx, KFLD, KLIST, LOOKUP, MzzZO, MOVE, MOVEA, MOVEL, MULT, MVR, OCCUR, ORxx, PARM, PLIST, REALLOC, SCAN, SETON, SETOFF, SHTDN, SQRT, SUB, SUBDUR, SUBST, TAG, TESTB, TESTN, TESTZ, TIME, WHENxx, XFOOT, XLATE, Z-ADD and Z-SUB. Using RPGWIZ with the correct parameter settings, most of these operations can be converted to operations which are supported in free-form.

Operations with unconverted left-hand indicators, result indicators or result field lengths will not be converted to free-form.

To maximize your conversion to free-form, you should also consider activating these additional parameters:

- Redefine *LIKE DEFN fields (LIKEFLD)
- Redefine calc. defined fields (CALCFLD)
- Convert left hand indicators (CVTLEFT)
- Convert opcodes to BIFs (OPCODEBIF)
- Insert file I/O BIFs (FILEBIF)
- Convert ADDs/SUBs to EVALs (CVTADDSUB)
- Convert Z-ADDs/Z-SUBs to EVALs (CVTZADDSUB)
- Convert MULTs to EVALs (CVTMULT)
- Convert DIVs to EVALs (CVTDIV)
- Convert MOVE(L)s having *BLANK (CVTMOVEBL)
- Convert MOVE(L)s having *ZERO (CVTMOVEZR)
- Convert MOVEs having data (CVTMOVER)
- Convert MOVELs having data (CVTMOVEL)
- Convert MOVEA operations (CVTMOVEA)
- Convert CASxx operations (CVTCAS)
- Convert CAT operations (CVTCAT)
- Convert DOs to FORs (CVTDO)
- Convert LOOKUP operations (CVTLOOKUP)
- Convert SCAN operations (CVTSCAN)
- Convert *ENTRY PLIST (CVTENTRY)
- Convert CALLs and CALLBs (CVTCALL)
- Convert GOTO operations (CVTGOTO)

***FREE2**
Like the *FREE option, the *FREE2 will convert C specifications into their free-form equivalents. In addition, the *FREE2 option will also convert H, F, D and P specifications into free-form syntax. The *FREE2 option will additionally remove any /free and /end-free directives from the source.

The *FREE2 option is only valid when targeting IBM i release V7R1 and higher. This level of free-form syntax was made available with the IBM i Technology Refresh 7 in November 2013. To support this free-form level in V7R1, you will need IBM PTF SI51094. For SQLRPGLE member types, you will also need IBM DB2 group PTF SF99701 level 26.

Examine field attributes (EXAMINEFLD)

Specifies if RPGWIZ should retrieve the attributes (types and lengths) of the fields used within the source member(s) to convert.

If you request to convert MOVE, MOVEL, ADD, SUB, MULT, DIV, Z-ADD or Z-SUB operations, this attribute information is used to determine which statements can be converted safely. For instance, a Z-ADD of a factor 2 field with a length greater than the result field is not considered safe and therefore will not be converted to an EVAL.

With EXAMINEFLD(*YES) specified, RPGWIZ will take longer to execute since it first has to retrieve these attributes. Even so, it is still highly recommended for safe conversions of MOVE, MOVEL, ADD, SUB, MULT, DIV, Z-ADD and Z-SUB operations.

***NO**
Does not retrieve the field attributes used within the source member(s) to convert.

***YES**
Retrieves the attributes of the fields used within the source member(s) to convert. RPGWIZ performs a *NOGEN compile using the CRTRPGMOD command for each source member to convert, which does not actually create an object. It then reads the compile listing(s) to retrieve the field attributes.

Before executing RPGWIZ, make sure your library list is set correctly so all externally described field attributes can be located.

If any RNF7030 errors (field definition not found) are encountered within the compile listing, any MOVE, MOVEL, ADD, SUB, MULT, DIV, Z-ADD or Z-SUB operations containing the unknown fields will not be converted.

Expand copy members (EXPCPY)

Specifies whether /COPY member(s) are expanded into the converted source member. This parameter is only valid when converting from RPG III or RPG/400 source members. EXPCPY(*YES) should be specified only if you are having conversion problems pertaining to /COPY members.

***YES**
Expand the /COPY file member(s) into the converted source.

***NO**
Do not expand the /COPY file member(s) into the converted source.

Considerations

When converting an auto report source member (type RPT or RPT38), any /COPY members will always be expanded into the converted source, even if you specify EXPCPY(*NO). This is because auto report is not supported by RPG IV and specifically does not support mixed specification types within /COPY members.

For auto report source members, it is recommended that you first change the source member type to RPG before the conversion, which will not automatically expand the /COPY members. For any /COPY members that contain mixed specification types, those members will need to be broken out by specification type.

Redefine data structures (REDEFINEDS)

Specifies whether or not to redefine the data structure fields.

***NO**
The data structures fields will not be redefined.

***YES**
The fields within eligible data structures will be redefined.

Data structure fields will be re-ordered into their natural sequence.

From/to positions will be converted to the actual length for any data structure field that either overlays or immediately follows (no positions in between) another field in the data structure.

Any data structure field that overlays another field will be indented under the field it overlays and the OVERLAY keyword will be added.

Example Before:

```
IDsname....NODsExt-file++.............OccrLen+...................
ICUST        DS
I.....................................PFromTo++DFldnmeL1M1FrPlMnZr.
I                                       57  66 PHN
I                                       57  59 PHNARE
I                                       60  62 PHNPFX
I                                       63  66 PHNLST
I                                        1  56 BILLTO
I                                        1  30 STREET
I                                       31  45 CITY
I                                       46  47 STATE
I                                       48  56 ZIP
I                                       48  52 ZIP5
I                                       53  56 ZIPEXT
```

Example After:

```
DName++++++++++ETDsFrom+++To/L+++IDc.Keywords++++++++++++++++++++++++++
D cust            ds
D billto                        56
D  street                       30    overlay(billto)
D  city                         15    overlay(billto:31)
D  state                         2    overlay(billto:46)
D  zip                           9    overlay(billto:48)
D   zip5                         5    overlay(zip)
D   zipext                       4    overlay(zip:6)
D phn                           10
D  phnare                        3    overlay(phn)
D  phnpfx                        3    overlay(phn:4)
D  phnlst                        4    overlay(phn:7)
```

Considerations

A data structure will not be redefined if any of its fields have lengths (versus from/to positions) already specified or if any of its fields have the DIM keyword specified.

Redefine *LIKE DEFN fields (LIKEFLD)

Specifies whether or not to redefine the *LIKE DEFN (or DEFINE) fields from the C specifications into the D specifications.

***NO**
*LIKE DEFN (or DEFINE) fields are not redefined.

***YES**
*LIKE DEFN (or DEFINE) fields are redefined in the Definition specifications and their corresponding Calculation specification lines are removed.

Multiple definitions of the same work field are only listed once in the D specifications.

All newly created D specification fields will be sorted alphabetically.

Example Before:

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
C     *LIKE         DEFINE    FIELDX        FIELD1
C     *LIKE         DEFINE    FIELDY        FIELD2          + 2
```

Example After:

```
DName++++++++++++ETDsFrom+++To/L+++IDc.Keywords++++++++++++++++++++++++
D FIELD1          S                   LIKE(FIELDX)
D FIELD2          S            + 2    LIKE(FIELDY)
```

Considerations

A traditional *LIKE defined numeric field in the C specifications is automatically set to a packed data type, even though it may be based on a field with a different numeric type (i.e. zoned). In contrast, LIKE defined fields in the D specifications will always adopt the type of the base field (i.e. zoned to zoned). An error will occur if you attempt to call another program with a redefined non-packed field as a parameter and if the called program is still expecting a packed field. To resolve this problem, convert the called program using RPGWIZ or change the type on the parameter field back to packed.

Redefine calc. defined fields (CALCFLD)

Specifies whether or not to redefine result fields with lengths or decimal positions from the C specifications into the D specifications.

***NO**
Result fields are not redefined.

***YES**
Result fields with lengths or decimal positions in the C specifications are redefined in the D specifications.

Lengths and decimal positions for those fields are removed from the C specification lines.

Calculation-defined data areas are also redefined into the D specifications.

Multiple definitions of the same work field are only listed once in the D specifications.

All newly created D specification fields will be sorted alphabetically.

Example Before:

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
C                   MOVE      'A'           FIELD3            1
C                   Z-ADD     123.56        FIELD4           10 2
C     *DTAARA       DEFINE    ORDNBR        ORDERN            4 0
```

Example After:

```
DName++++++++++ETDsFrom+++To/L+++IDc.Keywords++++++++++++++++++++++++
D FIELD3          s              1
D FIELD4          s             10  2
D ORDERN          s              4  0 DTAARA(ORDNBR)

CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
C                   MOVE      'A'           FIELD3
C                   Z-ADD     123.56        FIELD4
```

Convert left hand indicators (CVTLEFT)

Specifies whether or not to convert left hand (conditioning) indicators to IF structures.

***NO**
No left hand indicators will be converted to IF structures.

***YES**
Eligible left hand indicators will be converted to IF structures.

If the left hand indicators repeat themselves, their corresponding operations will be grouped under the same IF structure, which will improve the performance of your program. The exception is if their corresponding operations may turn on or off indicators (such as SETONs, SETOFs, COMPs, EXSRs and CALLPs), in which case the following operation will be placed under a new IF structure.

AN and OR continuations will be recognized and incorporated into the IF statement.

RPG/400 Example Before:

```
CL0N01N02N03Factor1+++OpcdeFactor2+++ResultLenDHHiLoEq
C   01
COR 02
CAN 03N04 05          MOVE *BLANKS   FIELD1
C*
C   02                MOVE *ZEROS    FIELD2
C   02                MOVE *ZEROS    FIELD3
C*
C   03                SETOF                     03
C   03                MOVE *ZEROS    FIELD4
```

RPGIV Example After:

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
C                   IF        *IN01 = *ON
C                             OR *IN02 = *ON
C                             AND *IN03 = *ON
C                             AND *IN04 = *OFF
C                             AND *IN05 = *ON
C                   MOVE      *BLANKS       FIELD1
C                   endif
 *
C                   IF        *IN02 = *ON
C                   MOVE      *ZEROS        FIELD2
C                   MOVE      *ZEROS        FIELD3
C                   endif
*
C                   IF        *IN03 = *ON
C                   EVAL      *IN03 = *OFF
C                   ENDIF
C                   IF        *IN03 = *ON
C                   MOVE      *ZEROS        FIELD4
C                   ENDIF
```

Considerations

Left hand indicators for CASxx, DOxxx, ENDxx, IFxx and SELEC operations are never converted to IF structures, due to the complexity and grouping nature of these operations.

Convert opcodes to BIFs (OPCODEBIF)

Specifies whether or not to convert eligible Calculation operation codes into their corresponding built-in-functions (BIFs) within EVAL operations. This is especially recommended when converting to free-form RPG since some operations are not supported in free-form logic.

***NO**
Operation codes will not be converted to built-in-functions (BIFs)

***YES**
Eligible operation codes will be converted to their corresponding built-in-functions (BIFs) available for the target release (TGTRLS). Operations will not be converted if:

- Result indicators are specified.
- Result field lengths are specified and you chose not to redefine the fields into the D specifications with CALCFLD(*NO).
- An error indicator is specified or an (E)rror extender is specified within the operation code.

BIFs available in TGTRLS(V4R2) and higher

SUBST (Substring)

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
 * Before (with padded blanks)
C     length        subst(p)  base          result
 * After
C                   eval      result = %subst(base:1:length)
 * Before (without padded blanks)
C     length        subst     base:start    result
 * After
C                   eval      %subst(result:1:length) =
C                              %subst(base:start:length)
 * Before (without padded blanks)
C                   subst     base:start    result
 * After
C                   eval      %subst(result:1
C                             :%size(base)-start+1) =
C                              %subst(base:start)
```

BIFs available in TGTRLS(V5R1) and higher

ADDDUR (Add a Duration to a Date/Time)

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
 * Before
C     date1         adddur    1:*year       resulta
C                   adddur    tmin:*mn      resultb
 * After
C                   eval      resulta = date1 + %year(1)
C                   eval      resultb = resultb + %minutes(tmin)
```

ALLOC (Allocate Memory)

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
 * Before
C                   alloc     length        result
 * After
C                   eval      result = %alloc(length)
```

BIFs available in TGTRLS(V5R1) and higher (continued)

CHECK (Check Characters)

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
 * Before
C     compare       check     base:start    result
C     compare       check     base          result1
 * After
C                   eval      result = %check(compare:base:start)
C                   eval      result1 = %check(compare:base)
```

- A result field must be present for conversion.

CHECKR (Check Reverse)

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
 * Before
C     compare       checkr    base:start    result1
C     compare       checkr    base          result1
 * After
C                   eval      result1 = %checkr(compare:base:start)
C                   eval      result1 = %checkr(compare:base)
```

- A result field must be present for conversion.

EXTRCT (Extract a Portion of a Date/Time)

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
 * Before
C                   extrct    date1:*y      yy1
C                   extrct    date2:*months mm2
 * After
C                   eval      yy1 = %subdt(date1:*y)
C                   eval      mm2 = %subdt(date2:*months)
```

- The result field must be numeric and an (E)rror operation extender must not be specified.

OCCUR (Set or Get an Occurrence)

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
 * Before
C     occur1        occur     ds
 * After
C                   eval      %occur(ds) = occur1
 * Before
C                   occur     ds            occur2
 * After
C                   eval      occur2 = %occur(ds)
 * Before
C     occur1        occur     ds            occur2
 * After
C                   eval      %occur(ds) = occur1
C                   eval      occur2 = %occur(ds)
```

- If a Factor1 field is specified, then the new BIF operation sets the occurance.
- If a Result field is specified, then the new BIF operation gets the occurance.
- If both fields are specified, then two statements are created; one BIF to set the occurance and another to get the occurance.

BIFs available in TGTRLS(V5R1) and higher (continued)

REALLOC (Reallocate Memory)

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
 * Before
C                   realloc   length        result
 * After
C                   eval      result = %realloc(length)
```

SQRT (Square Root)

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
 * Before
C                   sqrt      factor2       result
 * After
C                   eval      result = %sqrt(factor2)
```

SHTDN (Shutdown)

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
 * Before
C                   shtdn                                          01
 * After
C                   eval      *in01 = %shtdn
```

SUBDUR (Subtract a Duration from a Date/Time)

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
 * Before
C     date1         subdur    1:*year       resulta
C                   subdur    tmin:*mn      resultb
 * After
C                   eval      resulta = date1 - %year(1)
C                   eval      resultb = resultb - %minutes(tmin)
```

SUBDUR (Find the Difference between two Dates or Times)

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
 * Before
C     date1         subdur    date2         yydif:*years
C     time1         subdur    time2         hhdif:*hours
 * After
C                   eval      yydif = %diff(date1:date2:*years)
C                   eval      hhdif = %diff(time1:time2:*hours)
```

BIFs available in TGTRLS(V5R1) and higher (continued)

TIME (Retrieve Time and Date)

```
DName++++++++++ETDsFrom+++To/L+++IDc.Keywords++++++++++++++++++++++++++
D aptime          s               t
D apdate          s               d
D aptstamp        s               z

CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
 * Before
C                   time                    aptime
C                   time                    apdate
C                   time                    aptstamp

 * After
C                   eval      aptime = %time
C                   eval      apdate = %date
C                   eval      aptstamp = %timestamp
```

- EXAMINEFLD(*YES) must also be turned on.
- Only converts TIME operations which have result field types of T (time), D (date) or Z (timestamp).

XFOOT (Sum the Elements of an Array)

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
 * Before
C                   xfoot     array         sum
 * After
C                   eval      sum = %xfoot(array)
```

XLATE (Translate)

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
 * Before
C     lo:hi         xlate     source:start  result1
C     lo:hi         xlate     source        result2
 * After
C                   eval      result1 = %xlate(lo:hi:source:start)
C                   eval      result2 = %xlate(lo:hi:source)
```

Convert key lists (CVTKLIST)

Specifies whether or not to convert Key Lists into Data Structures or free-form search arguments within file I/O operations. This feature is valid in IBM i release V5R2 and higher within the free-form file I/O operations of CHAIN, DELETE, READE, READPE, SETLL and SETGT.

FMTCALC(*FREE) must also be specified on the RPGWIZ command since this style of search arguments is only valid in free-form I/O operations.

FILEBIF(*YES) must also be specified on the RPGWIZ command to remove result indicators from file I/O operations so they can be successfully converted to free-form operations.

***NO**
Key Lists will not be converted.

***YES**
Key Lists will be converted into Data Structures in the D specifications. File I/O operations will be modified to use these data structures (instead of key lists) with the %KDS built-in function.

Example Before:

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
C     CUSTKEY       KLIST
C                   PARM                    COMPNBR           2
C                   PARM                    CUSTNBR           5 0
 *
C     CUSTKEY       CHAIN     CUSTMAST
```

Example After:

```
DName++++++++++ETDsFrom+++To/L+++IDc.Keywords++++++++++++++++++++++++

D CUSTKEY         DS
D COMPNBR                       2
D CUSTNBR                       5  0

 /FREE
   CHAIN %KDS(CUSTKEY) CUSTMAS
 /END=FREE
```

A Data Structure will be created in the D specifications for each eligible KLIST. The new Data Structure names will be the same as the original KLIST names. Fields defined within each new Data Structure will contain the same field names, lengths and types as the original KFLD fields.

EXAMINEFLD(*YES) must also be specified on the RPGWIZ command so the KFLD field types and lengths can be found and specified within the new Data Structures.

Considerations

A data structure will not be created for a KLIST if the any of its KFLD fields lengths and types cannot be determined.

A key list will not be converted if one of the key fields contains an indicator value in factor 2.

The new data structure fields may have already been defined in the D specifications, so you may have to manually remove any duplicate field definitions after the conversion.

Convert key lists (continued)

***YES2**

Key list fields will be embedded as search arguments within the new free-form file I/O operations.

Example Before:

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
C     CUSTKEY       KLIST
C                   PARM                    COMPNBR           2
C                   PARM                    CUSTNBR           5 0
 *
C     CUSTKEY       CHAIN     CUSTMAST
```

Example After:

```
/FREE
  CHAIN ( COMPNBR : CUSTNBR ) CUSTMAS
/END=FREE
```

Tip: Existing KLIST structures will not be removed from the source member, but can be removed after the conversion using the CLNRPG (Clean RPG Source) command.

Insert file I/O BIFs (FILEBIF)

Specifies whether or not to insert file I/O built-in-functions (BIFs) under file operations, based on the result indicators specified. This is highly recommended when converting to free-format since result indicators are not allowed in free-form logic.

***NO**
File I/O built-in-functions (BIFs) will not be inserted under file operations.

***YES**
Appropriate file I/O built-in-functions (BIFs) will be inserted under file operations.

BIFs will only be inserted under file operations which have resulting indicators. These indicators will be moved into the BIFs.

When an error indicator is specified, an (E)rror extender is placed next to the opcode and an %Error BIF is inserted under the file operation.

***YES2**
Performs the same functionality as the *YES option and will additionally qualify any new file I/O BIFs with the name found in factor 2 of their corresponding file I/O operations.

Considerations when specifying *YES2
A record format name cannot be specified in a file I/O BIF (only a file name can be used). Therefore if any file I/O operations contain a record format name in factor 2, then you should either specify the *YES option on this parameter (for unqualified BIFs) OR first manually change any I/O operations to use file names versus record format names.

CHAIN example

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
 * Before
C     cmkey         chain     custmast                          0102
 * After
C     cmkey         chain(e)  custmast
C                   eval      *in02 = %error
C                   eval      *in01 = not %found(custmast)
```

DELETE example

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
 * Before
C     cmkey         delete    custrec                           0102
 * After
C     cmkey         delete(e) custrec
C                   eval      *in02 = %error
C                   eval      *in01 = not %found
```

EXFMT example

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
 * Before
C                   exfmt     dspfmt                                 02
 * After
C                   exfmt(e)  dspfmt
C                   eval      *in02 = %error
```

SETGT example

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
 * Before
C     cmkey         setgt     custmast                             0102
 * After
C     cmkey         setgt(e)  custmast
C                   eval      *in02 = %error
C                   eval      *in01 = not %found(custmast)
```

SETLL example

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
 * Before
C     cmkey         setll     custmast                             010203
 * After
C     cmkey         setll(e)  custmast
C                   eval      *in02 = %error
C                   eval      *in01 = not %found(custmast)
C                   eval      *in03 = %equal(custmast)
```

READE example (also applies to READ, READP, READC and READPE)

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
 * Before
C     cmkey         reade     custmast                               0202
 * After
C     cmkey         reade(e)  custmast
C                   eval      *in02 = %error or %eof(custmast)
```

UPDATE example

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
 * Before
C                   update    custrec                                02
 * After
C                   update(e) custrec
C                   eval      *in02 = %error
```

WRITE example

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
 * Before
C                   write     custrec                                02
 * After
C                   write(e)  custrec
C                   eval      *in02 = %error
```

Convert ADDs / SUBs to EVALs (CVTADDSUB)

Specifies whether or not to convert eligible ADD and SUB operations to their corresponding EVAL operations.

***NO**
ADD and SUB operations are not converted.

***YES**
Eligible ADD and SUB operations are converted to EVAL operations.

Existing half-adjust requests are incorporated into the EVAL operation.

ADD examples

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
 * Before
C                   ADD       2             FIELD1
C     FIELD1        ADD       900           FIELD2
C                   ADD(H)    1             FIELD3
 * After
C                   EVAL      FIELD1 = FIELD1 + 2
C                   EVAL      FIELD2 = FIELD1 + 900
C                   EVAL(H)   FIELD3 = FIELD3 + 1
```

SUB examples

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
 * Before
C                   SUB       2             FIELD1
C     FIELD1        SUB       900           FIELD2
C                   SUB(H)    1             FIELD3
* After
C                   EVAL      FIELD1 = FIELD1 - 2
C                   EVAL      FIELD2 = FIELD1 - 900
C                   EVAL(H)   FIELD3 = FIELD3 - 1
```

Considerations
RPGWIZ will never convert an arithmetic operation containing a resulting indicator since there is no equivalent EVAL operation.

If you chose not to redefine the calculation defined fields into the D specifications, RPGWIZ will not convert an arithmetic operation to an EVAL if it contains a result field length or decimal positions.

When converting ADD or SUB operations, it is recommended to turn on "Examine Field Attributes" with EXAMINEFLD(*YES), which will retrieve the field types and lengths for the source member. With this information, RPGWIZ will not convert an ADD or SUB operation if:

- The data types and lengths cannot be determined for factor 1, factor 2 or the result field.
- The significant portion of the result field length is not large enough to hold the result.

Otherwise if you specified EXAMINEFLD(*NO), then the above criteria will be ignored and you may get unpredictable results.

***YES2**
Performs the same functionality as the *YES option, but does not verify that the result field length is large enough to prevent potential overflow problems (even if EXAMINEFLD(*YES) is specified).

Convert Z-ADDs / Z-SUBs to EVALs (CVTZADDSUB)

Specifies whether or not to convert eligible Z-ADD and Z-SUB operations to their corresponding EVAL operations.

***NO**
Z-ADD and Z-SUB operations are not converted.

***YES**
Eligible Z-ADD and Z-SUB operations are converted to EVAL operations. Existing half-adjust requests are incorporated into the EVAL operation.

Z-ADD examples

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
 * Before
C                   Z-ADD     1             FIELD1
C                   Z-ADD(H)  AR(1)         FIELD2
C                   Z-ADD     1             FIELD3                   03
 * After
C                   EVAL      FIELD1 = 1
C                   EVAL(H)   FIELD2 = AR(1)
C                   Z-ADD     1             FIELD3                   03
```

Z-SUB Examples

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
 * Before
C                   Z-SUB     1             FIELD1
C                   Z-SUB(H)  AR(1)         FIELD2
C                   Z-SUB     1             FIELD3                   03
 * After
C                   EVAL      FIELD1 = 0 - 1
C                   EVAL(H)   FIELD2 = 0 - AR(1)
C                   Z-SUB     1             field3                   03
```

Considerations
RPGWIZ will never convert an arithmetic operation containing a resulting indicator since there is no equivalent EVAL operation.

If you chose not to redefine the calculation defined fields into the D specifications, RPGWIZ will not convert an arithmetic operation to an EVAL if it contains a result field length or decimal positions.

When converting Z-ADD or Z-SUB operations, it is recommended to turn on "Examine Field Attributes" with EXAMINEFLD(*YES), which will retrieve the field types and lengths for the source member. With this information, RPGWIZ will not convert a Z-ADD or Z-SUB operation if:

- The data types and lengths cannot be determined for factor 2 or the result field.
- The result field length or decimal positions are less than those of factor 2.

Otherwise if you specified EXAMINEFLD(*NO), then the above criteria will be ignored and you may get unpredictable results.

***YES2**
Performs the same functionality as the *YES option, but does not verify that the result field length is large enough to prevent potential overflow problems (even if EXAMINEFLD(*YES) is specified).

Convert MULTs to EVALs (CVTMULT)

Specifies whether or not to convert eligible MULT operations to EVAL operations.

***NO**
MULT operations are not converted.

***YES**
Eligible MULT operations are converted to EVAL operations.

Existing half-adjust requests are incorporated into the EVAL operation.

Examples

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
 * Before
C                   MULT      2             FIELD1
C     FIELD1        MULT      3             FIELD2
C     FIELD1        MULT(H)   4             FIELD3
C                   MULT      5             FIELD4                    03
 * After
C                   EVAL      FIELD1 = FIELD1 * 2
C                   EVAL      FIELD2 = FIELD1 * 3
C                   EVAL(H)   FIELD3 = FIELD1 * 4
C                   MULT      5             FIELD4                    03
```

Considerations
RPGWIZ will never convert a MULT operation containing a resulting indicator since there is no equivalent EVAL operation.

If you chose not to redefine the calculation defined fields into the D specifications, RPGWIZ will not convert a MULT operation to an EVAL if it contains a result field length or decimal positions.

RPGWIZ will never convert a MULT operation containing the date conversion values of 10000.01, 100.0001 or 10000.0001 in Factor 1 or Factor 2 since it would overflow with an EVAL operation.

When converting MULT operations, it is recommended to turn on "Examine Field Attributes" with EXAMINEFLD(*YES), which will retrieve the field types and lengths for the source member. With this information, RPGWIZ will not convert a MULT operation if:

- The data types and lengths cannot be determined for factor 1, factor 2 or the result field.
- The significant portion of the result field length is not large enough to hold the result.

Otherwise if you specified EXAMINEFLD(*NO), then the above criteria will be ignored and you may get unpredictable results.

***YES2**
Performs the same functionality as the *YES option, but does not verify that the result field length is large enough to prevent potential overflow problems (even if EXAMINEFLD(*YES) is specified).

Convert DIVs to EVALs (CVTDIV)

Specifies whether or not to convert eligible DIV operations to EVAL operations.

***NO**
DIV operations are not converted.

***YES**
Eligible DIV operations are converted to EVAL operations.

Existing half-adjust requests are incorporated into the EVAL operation.

Corresponding MVR operations are converted to %REM built-in-functions for a target release of V4R4 and higher.

Examples

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
 * Before
C                   DIV       2             FIELD1
C     FIELD1        DIV       3             FIELD2
C     FIELD1        DIV(H)    4             FIELD3
C                   DIV(H)    5             FIELD4                    03
C     FIELD4        DIV       FIELD3        FIELD5
C                   MVR                     FIELD6
 * After
C                   EVAL      FIELD1 = FIELD1 / 2
C                   EVAL      FIELD2 = FIELD1 / 3
C                   EVAL(H)   FIELD3 = FIELD1 / 4
C                   DIV(H)    5             FIELD4                    03
C                   EVAL      FIELD5 = FIELD4 / FIELD3
C                   EVAL      FIELD6 = %REM( FIELD4 : FIELD3 )
```

Considerations

RPGWIZ will never convert a DIV operation containing a resulting indicator since there is no equivalent EVAL operation.

If you chose not to redefine the calculation defined fields into the D specifications, RPGWIZ will not convert a DIV operation to an EVAL if it contains a result field length or decimal positions.

When converting DIV operations, it is recommended to turn on "Examine Field Attributes" with EXAMINEFLD(*YES), which will retrieve the field types and lengths for the source member. With this information, RPGWIZ will not convert a DIV operation if:

- The data types and lengths cannot be determined for factor 1, factor 2 or the result field.
- The significant portion of the result field length is not large enough to hold the result.

Otherwise if you specified EXAMINEFLD(*NO), then the above criteria will be ignored and you may get unpredictable results.

***YES2**
Performs the same functionality as the *YES option, but does not verify that the result field length is large enough to prevent potential overflow problems (even if EXAMINEFLD(*YES) is specified).

Convert MOVE(L)s having *BLANK (CVTMOVEBL)

Specifies whether or not eligible MOVE and MOVEL operations with *BLANK(S) in factor 2 should be converted to either EVAL or CLEAR operations. It is recommended to also specify EXAMINEFLD(*YES) and CALCFLD(*YES) to ensure a successful conversion.

***NO**
MOVEs and MOVELs with *BLANK(S) in factor 2 will not be converted.

***EVAL**
Eligible MOVE and MOVEL operations with *BLANK(S) in factor 2 will be converted to EVAL operations.

If you also specify EXAMINEFLD(*YES), then RPGWIZ will:
- Not convert the MOVE or MOVEL if the data type cannot be determined for the result field.
- Substitute in the special value of *ZEROS if the MOVE(L) was assigning *BLANK(S) to a numeric field, since you cannot assign *BLANK(S) to a numeric field within an EVAL.

If you specify EXAMINEFLD(*NO), then RPGWIZ will not convert any MOVE(L) which contains resulting indicator(s).

If you also specify CALCFLD(*NO), then RPGWIZ will not convert any MOVE(L) which contains a result field length or decimal positions.

*EVAL Example:

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
 * Before
C                   MOVE      *BLANK        Alpha5
C                   MOVEL     *BLANKS       Alpha2                   03
C                   MOVE      *BLANKS       Num2
 * After
C                   EVAL      Alpha5 = *BLANK
C                   EVAL      Alpha2 = *BLANKS
C                   EVAL      *IN03 = (Alpha2 = *BLANKS)
C                   EVAL      Num2 = *ZEROS
```

***CLEAR**
Eligible MOVE and MOVEL operations with *BLANK(S) in factor 2 will be converted to CLEAR operations.

*CLEAR Example:

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
 * Before
C                   MOVE      *BLANK        Alpha5
C                   MOVE      *BLANKS       Alpha2
C                   MOVE      *BLANKS       Num2
 * After
C                   CLEAR                   Alpha5
C                   CLEAR                   Alpha2
C                   CLEAR                   Num2
```

Convert MOVE(L)s having *ZERO (CVTMOVEZR)

Specifies whether or not eligible MOVE and MOVEL operations with *ZERO(S) in factor 2 should be converted to either EVAL or CLEAR operations. It is recommended to also specify EXAMINEFLD(*YES) and CALCFLD(*YES) to ensure a successful conversion.

***NO**
MOVEs and MOVELs with *ZERO(S) in factor 2 will not be converted.

***EVAL**
Eligible MOVE and MOVEL operations with *ZERO(S) in factor 2 will be converted to EVAL operations.

If you also specify EXAMINEFLD(*YES), then RPGWIZ will:
- Not convert the MOVE or MOVEL if the data type cannot be determined for the result field.
- Substitute in a string of zeros if the MOVE(L) was assigning *ZERO(S) to an alphanumeric field.

If you also specify EXAMINEFLD(*NO), then RPGWIZ will not convert any MOVE(L) which contains resulting indicator(s).

If you also specify CALCFLD(*NO), then RPGWIZ will not convert any MOVE(L) which contains a result field length or decimal positions.

*EVAL Example:

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
 * Before
C                   MOVE      *ZERO         Num2
C                   MOVEL     *ZEROS        Num5                   03
C                   MOVE      *ZEROS        Alpha5
 * After
C                   EVAL      Num2 = *ZERO
C                   EVAL      Num5 = *ZEROS
C                   EVAL      *IN03 = (Num5 = *ZEROS)
C                   EVAL      Alpha5 = '00000'
```

***CLEAR**
Eligible MOVE and MOVEL operations with *ZERO(S) in factor 2 will be converted to CLEAR operations.

Warning! A MOVE or MOVEL of *ZEROS to an alpha field will fill the field with zeros, whereas a CLEAR operation will fill the field with blanks. The conversion of this type of MOVE operation to a CLEAR may therefore create unpredictable results in your program. If you specify EXAMINEFLD(*YES), then RPGWIZ will only convert a MOVE *ZERO(S) to a CLEAR operation when the result field is numeric.

*CLEAR Example:

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
* Before
C                   MOVE      *ZERO         Num2
C                   MOVE      *ZEROS        Num5
C                   MOVE      *ZEROS        Alpha5
 * After
C                   CLEAR                   Num2
C                   CLEAR                   Num5
C                   MOVE      *ZEROS        Alpha5
```

Convert MOVEs having data (CVTMOVER)

Specifies whether or not MOVE operations having data in Factor 2 (something other than *BLANKs or *ZEROs) should be converted to EVAL operations. It is recommended to also specify EXAMINEFLD(*YES) and CALCFLD(*YES) to ensure a successful conversion.

***NO**
MOVE operations having data in factor 2 will not be converted.

***EVAL**
Eligible MOVE operations having data in factor 2 will be converted to EVAL operations within the source member.

Convert MOVELs having data (CVTMOVEL)

Specifies whether or not MOVEL operations having data in Factor 2 (something other than *BLANKs or *ZEROs) should be converted to EVAL operations. It is recommended to also specify EXAMINEFLD(*YES) and CALCFLD(*YES) to ensure a successful conversion.

***NO**
MOVEL operations having data in factor 2 will not be converted.

***EVAL**
Eligible MOVEL operations having data in factor 2 will be converted to EVAL operations within the source member.

Considerations when converting MOVE and MOVEL operations

If you also specify EXAMINEFLD(*YES), then RPGWIZ will not convert the MOVE(L) operation if:

- The data types and lengths cannot be determined for factor 2 or the result field.
- Factor 2 is alphanumeric and the result field is numeric.
- Factor 2 and the result field are both numeric and if:
  (a) the operation is a MOVE and the result field's length or decimals are less than those of factor 2.
  (b) the operation is a MOVE and the result field's length or decimals are greater than those of factor 2 and a (P) operation extender is not specified on the MOVE.
  (c) the operation is a MOVEL and the lengths of factor 2 and the result field don't match.

If you also specify EXAMINEFLD(*NO), then RPGWIZ will not convert any MOVE(L) which contains resulting indicator(s).

If you also specify CALCFLD(*NO), then RPGWIZ will not convert any MOVE(L) which contain a result field length or decimal positions.

Examples of converting MOVE and MOVEL operations when EXAMINEFLD(*YES) is specified

MOVE or MOVEL - Result field is the same size as factor 2 and data types are compatible:

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
 * Before
C                   MOVE      Num2          Work2N
C                   MOVEL     Alpha5        Work5A
 * After
C                   EVAL      Work2N = Num2
C                   EVAL      Work5A = Alpha5
```

MOVE and MOVEL examples (continued)

MOVE Alpha to Alpha - Result field is larger than factor 2:

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
 * Before
C                   MOVE      Alpha2        Alpha5
 * After
C                   EVAL      %SUBST(Alpha5:4:2) = Alpha2
```

MOVE Alpha to Alpha - Result field is larger than factor 2 and is padded with blanks:

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
 * Before
C                   MOVE(P)   Alpha2        Alpha5
 * After
C                   EVAL      Alpha5 = *BLANKS
C                   EVAL      %SUBST(Alpha5:4:2) = Alpha2
```

MOVE Alpha to Alpha - Result field is smaller than factor 2:

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
 * Before
C                   MOVE      Alpha5        Alpha2
 * After
C                   EVAL      Alpha2 = %SUBST(Alpha5:4:2)
```

MOVEL Alpha to Alpha - Result field is larger than factor 2:

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
 * Before
C                   MOVEL     Alpha2        Alpha5
 * After
C                   EVAL      %SUBST(Alpha5:1:2) = Alpha2
```

MOVEL Alpha to Alpha - Result field is larger than factor 2 and is padded with blanks:

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
 * Before
C                   MOVEL(P)  Alpha2        Alpha5
 * After
C                   EVAL      Alpha5 = Alpha2
```

MOVEL Alpha to Alpha - Result field is smaller than factor 2:

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
 * Before
C                   MOVEL     Alpha5        Alpha2
 * After
C                   EVAL      Alpha2 = Alpha5
```

MOVE and MOVEL examples (continued)

MOVE or MOVEL Numeric to Alpha – Result field is same size as factor 2:

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
 * Before
C                   MOVE      Num5          Alpha5
 * After
C                   EVAL      Alpha5 = %EDITC(Num5:'X')
```

MOVE Numeric to Alpha – Result field is smaller than factor 2:

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
 * Before
C                   MOVE      Num5          Alpha2
 * After
C                   EVAL      Alpha2 = %SUBST(%EDITC(Num5:'X'):4:2)
```

MOVEL Numeric to Alpha – Result field is larger than factor 2:

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
 * Before
C                   MOVEL     Num2          Alpha5
 * After
C                   EVAL      %SUBST(Alpha5:1:2) = %EDITC(Num2:'X')
```

MOVE or MOVEL other field types to Date, Time and Timestamp field types:

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
 * Before
C     *USA          MOVE      Alpha10       OrdDate
C                   MOVE      Num6          OrdTime
C                   MOVE      Alpha26       OrdTimeStp
 * After
C                   EVAL      OrdDate = %DATE(Alpha10:'*USA')
C                   EVAL      OrdTime = %TIME(Num6)
C                   EVAL      OrdTimeStp = %TIMESTAMP(Alpha26)
```

Resulting indicators are specified:

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
 * Before
C                   MOVE      Num2          Num5                 010102
 * After
C                   EVAL      Num5 = Num2
C                   EVAL      *IN01 = (Num5 <> *ZEROS)
C                   EVAL      *IN02 = (Num5 = *ZEROS)
```

Convert MOVEA operations (CVTMOVEA)

Specifies whether or not MOVEA operations should be converted.

***NO**
MOVEA operations will not be converted.

***YES**
Eligible MOVEA operations will be converted within the source member.

The following RPGWIZ keyword values must also be specified when converting MOVEA operations:

- EXAMINEFLD(*YES) to examine field types and lengths.
- TGTRLS of V4R4 or higher, since the FOR operation may be used to convert complex MOVEA operations.

Examples of converting MOVEA operations

MOVEA of special values to all array elements:

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
 * Before
C                   MOVEA     *ON           *IN
C                   MOVEA     *OFF          Array1
C                   MOVEA     *ZEROS        Array2
C                   MOVEA     *BLANKS       Array3

 * After
C                   EVAL      *IN = *ON
C                   EVAL      Array1 = *OFF
C                   EVAL      Array2 = *ZEROS
C                   EVAL      Array3 = *BLANKS
```

MOVEA of constant across multiple array elements:

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
 * Before
C                   MOVEA     '101'         *IN(50)
 * After
C                   EVAL      *IN(50) = *ON
C                   EVAL      *IN(51) = *OFF
C                   EVAL      *IN(52) = *ON
```

MOVEA of constant to all array elements:

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
 * Before
C                   MOVEA     '1'           *IN
 * After
C                   EVAL      *IN = *ON
```

(continued)

Convert MOVEA operations (continued)

MOVEA of entire array to another entire array:

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
 * Before
C                   MOVEA     Array1        Array2
 * After
C                   EVAL      Array2 = Array1
```

MOVEA *ALL of constant to partial array:

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
 * Before
C                   MOVEA     *all'A'       Array1(10)
 * After
C                   FOR       ma_x = 10 to %ELEM(Array1)
C                   EVAL      Array1(ma_x) = 'A'
C                   ENDFOR
```

MOVEA variable to partial array:

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
 * Before
C                   MOVEA     TestFld       Array1(N)
 * After
C                   FOR       ma_x = N to %ELEM(Array1)
C                   EVAL      Array1(ma_x) = TestFld
C                   ENDFOR
```

Considerations when converting MOVEA operations

RPGWIZ will not convert a MOVEA operation if:

- The data types and lengths cannot be determined for factor 2 or the result array.
- The data type and length of factor 2 does not match the data type and length of the result array elements.
- The MOVEA is moving an entire array to another entire array and the total array elements do not match.

Convert CASxx operations (CVTCAS)

Specifies whether or not CASxx operations should be converted to EXSR or CALLP operations. This is recommended when converting to free-form RPG (since CASxx operations are not supported in free-form) or when converting subroutines to sub-procedures (since you cannot call out a sub-procedure using a CASxx operation).

***NO**
CASxx operations will not be converted.

***YES**
CASxx operations will be converted to their corresponding EXSR or CALLP operations. EXSR operations are used when the "Convert Subroutines to procs" parameter CVTSUBR(*NO) is specified and CALLP operations are used when CVTSUBR(*YES) is specified.

CASxx structures containing left-hand indicators will not be converted, due to the complexity and grouping nature of left-hand indicators.

CASxx structures without right-hand indicators will be converted to SELECT/WHEN structures, otherwise they will be converted to IF/ELSE structures.

Example of converting to EXSRs

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
 * Before
C     FIELD1         CASEQ     FIELD2         $EQUAL
C     FIELD1         CASLT     FIELD2         $LESS
C                    CAS                      $OTHER
C                    ENDCS
* After
C                    SELECT
C                    WHEN      FIELD1 = FIELD2
C                    EXSR      $EQUAL
C                    WHEN      FIELD1 < FIELD2
C                    EXSR      $LESS
C                    OTHER
C                    EXSR      $OTHER
C                    ENDSL
```

Convert CAT operations (CVTCAT)

Specifies whether or not to convert eligible CAT (concatenate two strings) operations to EVAL operations.

***NO**
CAT operations are not converted.

***YES**
Eligible CAT operations will be converted to EVAL operations when either:

- The CAT operation contains a padding (P) operation extender OR
- The length of the result field is less than or equal to the total length of factor 1, factor 2 and the number of blanks (if specified).

If the number of blanks is specified within factor 2 (after the colon), then factor 1 will be trimmed of trailing blanks using the %TRIMR built-in-function. If the number of blanks specified is greater than 0, then a corresponding empty string will be inserted within the resulting EVAL operation.

***YES2**
Performs the same functionality as the *YES option and will additionally convert CAT operations which may generate a value which is smaller than the result field length.

Warning! An EVAL operation will automatically pad the right side of the resulting field with blanks, but a CAT operation without a (P) operation extender will retain the rightmost portion of the result field (if the result field is larger than the new value). Specifying *YES2 for CVTCAT may therefore cause unpredictable results if the program expects to carry forward the rightmost portion of the result from a prior operation.

Examples:

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
 * Before
C     FIRST         CAT(P)    LAST:1        FULLNAME
C                   CAT(P)    ZIP           ADDRESS
C     LEFT          CAT       RIGHT:0       JOINED
 * After
C                   EVAL      FULLNAME = %TRIMR(FIRST) + ' ' + LAST
C                   EVAL      ADDRESS = ADDRESS + ZIP
C                   EVAL      JOINED = %TRIMR(LEFT) + RIGHT
```

Considerations

When converting CAT operations, it is recommended to turn on "Examine Field Attributes" with EXAMINEFLD(*YES), which will retrieve the field types and lengths for the source member, allowing for a more intelligent conversion of CAT operations.

If you chose not to redefine the calculation defined fields into the D specifications, RPGWIZ will not convert a CAT operation to an EVAL if it contains a result field length.

If the number of blanks is specified, this value cannot be a variable name. The number of blanks specified must be a constant numeric value since RPGWIZ needs to know how large of empty string to concatenate within the resulting EVAL operation. This constant value can be a number from 0 to 25. If the CAT operation does not meet this criteria, then it will not be converted.

Convert DOs to FORs (CVTDO)

Specifies whether or not to convert eligible DO operations to FOR operations.

***NO**
DO operations are not converted.

***YES**
Eligible DO operations will be converted to FOR operations. Their corresponding END or ENDDO operations are converted to ENDFOR operations.

The starting, limit and index values within a DO operation will be incorporated into the new FOR operation.

If a starting value is not specified within factor 1 of the DO operation, then the starting value will be defaulted to the number 1 within the FOR operation.

If a limit value is not specified within factor 2 of the DO operation, then the limit value will be defaulted to the number 1 within the FOR operation.

If an index variable is not specified within the result field of the DO operation, then the index variable will be defaulted to the name of DO_X within the FOR operation.

If an increment value is specified in the DO's corresponding END or ENDDO operation, then that increment value will be moved into the new FOR statement.

Any existing ITER or LEAVE operations within converted DO loops will retain the same behavior.

Example Before:

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
 * Loop with an increment value
C     START         DO        LIMIT         INDEX
C                   ' '
C                   ENDDO     XXX
 * Loop without a start value
C                   DO        LIMIT         INDEX
C                   ' '
C                   END
```

Example After:

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
 * Loop with an increment value
C                   FOR       INDEX = START BY XXX TO LIMIT
C                   ' '
C                   ENDFOR
 * Loop without a start value
C                   FOR       INDEX = 1 to LIMIT
C                   ' '
C                   ENDFOR
```

Considerations

FOR operations are only supported in IBM i release of V4R4 and higher. Therefore DO operations will not be converted if you specify a target release lower than V4R4.

If you chose not to redefine the calculation defined fields into the D specifications with CALCFLD(*NO), RPGWIZ will not convert a DO operation if it contains a result field length.

Convert LOOKUP operations (CVTLOOKUP)

Specifies whether or not to convert eligible LOOKUP operations to %LOOKUP and %TLOOKUP built-in functions.

***NO**
LOOKUP operations are not converted.

***YES**
Eligible LOOKUP operations will be converted to %LOOKUP and %TLOOKUP built-in functions.

If the variable name in factor 2 starts with "TAB", then the LOOKUP will be converted to a %TLOOKUP (Look Up a Table Element). Otherwise the LOOKUP will be converted to a %LOOKUP (Look Up an Array Element).

Example Before:

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
 * Lookup an Array element
C     SEARCH          LOOKUP   ARRAY1(INDEX)                          99
 * Lookup a Table element
C     SEARCH          LOOKUP   TAB1          TAB2                 99  99
```

Example After:

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
 * Lookup an Array element
C                   EVAL      INDEX = %LOOKUP(SEARCH:ARRAY1:INDEX)
C                   IF        INDEX > 0
C                   EVAL      *IN99 = *ON
C                   ELSE
C                   EVAL      *IN99 = *OFF
C                   EVAL      INDEX = 1
C                   ENDIF
 * Lookup a Table element
C                   EVAL      *IN99 = %TLOOKUPGE(SEARCH:TAB1:TAB2)
```

* Notice after the new %LOOKUP that the variable INDEX is set to 1 if the element is not found, which preserves the behavior of the old LOOKUP operation.

Considerations

The %LOOKUP and %TLOOKUP built-in functions are only supported in IBM i release of V5R1 and higher. Therefore LOOKUP operations will not be converted if you specify a target release lower than V5R1.

If both the High and Equal result indicators are specified within a LOOKUP, then RPGWIZ will only convert this LOOKUP if both indicators match. The Low and Equal indicators must also match if they are both specified.

An index variable is required as the result to a %LOOKUP operation. Therefore an Array LOOKUP operation will only be converted if an index value is specified within it's factor 2 (within parenthesis). This index value must be a variable name.

The %LOOKUP and %TLOOKUP built-in functions do not turn on the %EQUAL or %FOUND built-in functions. If %ERROR or %FOUND was used to test the success of a converted LOOKUP operation, then unpredictable results may occur in your converted logic.

Convert SCAN operations (CVTSCAN)

Specifies whether or not to convert eligible SCAN operations to %SCAN built-in functions.

***NO**
SCAN operations are not converted.

***YES**
Eligible SCAN operations will be converted to %SCAN built-in functions.

Example:

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq

 * Before
C     COMPARE       SCAN      BASE:START    RESULT

 * After
C                   EVAL      RESULT = %SCAN(COMPARE:BASE:START)
```

Considerations

The %SCAN built-in function is only supported in IBM i release of V4R1 and higher. Therefore SCAN operations will not be converted if you specify a target release lower than V4R1.

A SCAN operation will not be converted if it contains a resulting indicator.

A SCAN operation will not be converted if it does not contain a result variable.

A SCAN operation will not be converted if the result variable is an array.

A SCAN operation will not be converted if it contains a compare string length in factor 1.

Warning! The %FOUND built-in-function is not supported for the %SCAN built-in-function. Therefore if the %FOUND built-in-function was used in the logic to check the success of a SCAN operation, then you'll need to modify your logic to instead check the result value of the new %SCAN.

Convert *ENTRY PLIST (CVTENTRY)

Specifies whether or not the *ENTRY PLIST structure should be moved from the C specifications into the D specifications.

***NO**
The *ENTRY PLIST will not be converted.

***YES**
An eligible *ENTRY PLIST will be converted.

Both a Procedure Interface (PI) and a Prototype (PR) will be created in the D specifications that corresponds with the *ENTRY PLIST. The existing *ENTRY PLIST will be removed from the C specifications.

The names of the Procedure Interface and Prototype will match the program name, which will provide the same behavior as an *ENTRY PLIST.

Fields defined within the new Procedure Interface will contain the same names, lengths and types as the result fields which were defined within the *ENTRY PLIST.

Fields defined within the new Prototype will be appended with an underscore character and will refer to field name definitions within the Procedure Interface (using the LIKE keyword).

Example Before (for a program named "OER001"):

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
C     *ENTRY        PLIST
C                   PARM                    CUSTNO            8
C                   PARM                    ORDERNO           5 0
C                   PARM                    STATUS            1
```

Example After:

```
DName++++++++++ETDsFrom+++To/L+++IDc.Keywords++++++++++++++++++++++++++
 *  Prototype for OER001
D OER001          PR
D CUSTNO_                               LIKE(CUSTNO)
D ORDERNO_                              LIKE(ORDERNO)
D STATUS_                               LIKE(STATUS)

 * *ENTRY Interface for Main Procedure
D OER001          PI
D CUSTNO                         8
D ORDERNO                        5  0
D STATUS                         1
```

Notice that the program name OER001 was used for both the Prototype and Procedure Interface names, which will provide the same behavior as an *ENTRY PLIST.

***YES2**
Performs the same functionality as the *YES option, but will place the field types/lengths into the created Prototype versus using the LIKE keyword. You must also specify the option EXAMINEFLD(*YES) so the field types/lengths are known.

(continued)

Considerations

It is recommended that you also specify EXAMINEFLD(*YES) on the RPGWIZ command so the *ENTRY parameter result field types and lengths can be found and specified within the Procedure Interface (even if those lengths were not directly coded on the *ENTRY parameter lines).

It is recommended that you also specify CALCFLD(*YES) on the RPGWIZ command, which allows RPGWIZ to remove the field lengths and decimal positions from the *ENTRY parameter fields.

An *ENTRY PLIST will not be converted if:

- You chose not to redefine the calculation defined fields with CALCFLD(*NO) and any of the *ENTRY parameters are coded with a result field length or decimal positions.
- The length and type of any *ENTRY parameter field cannot be determined.
- Any of the *ENTRY parameters contain a result field which is defined as a data structure (since a data structure cannot be defined within a procedure interface).
- Any of the *ENTRY parameters contain a value in either factor 1 or factor 2.

The RPG compiler does not allow a Procedure Interface parameter name to duplicate a field name already defined in the D specifications (ie standalone fields), so you may have to manually find and remove any of these duplicated field names from the other D specifications.

The Prototype name created for the *ENTRY PLIST will match the program name. The RPG compiler does not allow a Prototype name to duplicate another field name or another Prototype name. If for some reason this new Prototype name is duplicated, then you will have to manually find and change the other field or Prototype name.

Convert Subroutines to Procs (CVTSUBR)

Specifies whether or not to convert subroutines to sub-procedures.

***NO**
Subroutines are not converted to sub-procedures.

***YES**
Subroutines are converted to internal sub-procedures.

EXSR operations are converted to CALLP operations for converted subroutines.

Prototypes are created in the Definition specifications for their corresponding sub-procedures.

LEAVESR operations are converted to RETURN operations within converted subroutines.

If Factor1 of a ENDSR contains a tag value, then a tag operation is inserted just above the end of the new sub-procedure (so GOTO and CABxx statements will still operate correctly).

Example Before:

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
C                   EXSR      $EDITCHK
C                   ' '
C     $EDITCHK      BEGSR
C                   ' '
C                   LEAVESR
C                   ' '
C                   ENDSR
```

Example After:

```
DName++++++++++ETDsFrom+++To/L+++IDc.Keywords++++++++++++++++++++++++++++
D $EDITCHK        PR

CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
C                   CALLP     $EDITCHK
C                   ' '
P $EDITCHK        B
C                   ' '
C                   RETURN
C                   ' '
P $EDITCHK        E
```

Considerations

The IBM special subroutines of *INZSR and *PSSR are not converted to sub-procedures.

When converting subroutines, you should also convert CASxx operations with CVTCAS(*YES).

Warning! A RETURN operation within a subroutine will end the program, but a RETURN within a sub-procedure will return control to the statement after the CALLP operation which called it. To avoid unpredicatable runtime results with existing RETURN operations within your converted subroutines, RPGWIZ will change their specification type to an invalid value of "X" so the converted program cannot compile. For those invalidated RETURN operations, you will need to manually modify your logic to get the same results as the original program.

(continued)

Mainline specifications or unconverted subroutines (i.e. *INZSR) cannot follow sub-procedures. If the new sub-procedures are not at the bottom of your program, you need to manually move them there.

Due to restrictions in the ILE model, a program which contains sub-procedure(s) cannot be compiled into the default activation group. Therefore you must specify a named activation group on the ACTGRP parameter when compiling a program with sub-procedures.

Convert CALLs and CALLBs (CVTCALL)

Specifies whether or not CALL and CALLB operations should be converted to CALLP operations.

***NO**
CALL and CALLB operations will not be converted.

***YES** or ***YES2**
Eligible CALL and CALLB operations will be converted to CALLP operations. Prototype definitions will also be created in the D specifications for converted CALL and CALLB operations.

CALL or CALLB operations will be eligible for conversion which:

- Do not contain parameters OR
- Have parameters defined immediately after the CALL or CALLB operation OR
- Have a PLIST name specified in the result field and you also specified EXAMINEFLD(*YES).

Example Before:

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
 * Execute the DSPMSG command
C                   CALL      'QCMDEXC'                             99
C                   PARM      'DSPMSG'      COMMAND          80
C                   PARM      6             CMDLENGTH        15 5
```

Example After:

```
DName++++++++++ETDsFrom+++To/L+++IDc.Keywords++++++++++++++++++++++++++
 * Standalone work fields
D COMMAND         S             80
D CMDLENGTH       S             15  5

 * Prototype for 'QCMDEXC'
D QCMDEXC         PR                  EXTPGM('QCMDEXC')
D COMMAND_                            LIKE(COMMAND)
D CMDLENGTH_                          LIKE(CMDLENGTH)

 * Execute the DSPMSG command
C                   EVAL      COMMAND = 'DSPMSG'
C                   EVAL      CMDLENGTH = 6
C                   CALLP     QCMDEXC ( COMMAND : CMDLENGTH )
C                   EVAL      *IN99 = %ERROR
```

Prototype details

RPGWIZ will attempt to name the Prototype definition with the value found in factor 2 of the existing CALL or CALLB operation, minus any quotes. If RPGWIZ determines that this name is already used by another field or another Prototype name, then a sequential number will be appended to the new Prototype name.

The Prototype will use the keyword of EXTPGM for a converted CALL operation and EXTPROC for a converted CALLB operation.

Any parameter fields defined within the new Prototype will be appended with an underscore character.

(continued)

If you specify the *YES option on the CVTCALL keyword, these new Prototype fields will refer to the original parameter result field's definition using the LIKE keyword. If you specify the *YES2 option, the field types/lengths will be placed directly into the created Prototype (you must also specify the option EXAMINEFLD(*YES) so the field types/lengths are known).

For CALL or CALLB operations that use parameter lists (PLIST), only one Prototype will be created for each unique combination of the program/module name and parameter list (PLIST) name.

CALLP details

An (E) operation extender will be appended to the new CALLP operation if any of the following conditions are met:

- An (E) operation extender was specified on the CALL or CALLB operation.
- An error result indicator was specified in positions 73 and 74 on the CALL or CALLB operation.
- Any of the CALL or CALLB parameters had a variable specified in factor 1 (so the result can be moved into the factor 1 variable only after a successful call).

Any parameters that were specified within a CALL or CALLB operation will be placed within parenthesis after the CALLP and will be separated by colons.

If a value was specified in factor 2 of the original PARM statement, then RPGWIZ will insert a line above the CALLP to move this factor 2 value into the result field using an EVAL operation.

If a variable was specified in factor 1 of the original PARM statement, then RPGWIZ will insert a line below the CALLP to move the result field into this factor 1 variable using an EVAL operation. This line will only execute if the CALLP was successful (IF %ERROR = *OFF).

Warning! In a traditional CALL or CALLB operation, the result field variable is copied into factor 1 after the called program is initialized, even if the called program errors out. The new logic will only move the result field into factor 1 if the CALLP completes with no errors. If you have factor 1 specified in any of your PARM statements, then you need to be aware of this different behavior.

Considerations

It is recommended that you also specify EXAMINEFLD(*YES) on the RPGWIZ command so 1) all the PLISTs can be located within both the source member and its /COPY books and 2) all existing field names are identified so newly created Prototypes can be uniquely named.

A CALL or CALLB operation will not be converted if:

- You chose not to redefine the calculation defined fields with CALCFLD(*NO) and any of the parameters are coded with a result field length or decimal positions.
- The CALL or CALLB operation has a resulting indicator specified in positions 75 and 76 (which turns on if the called program returns with the LR indicator on). The CALLP operation does not accommodate this resulting indicator.

The RPG compiler does not allow a Prototype name to duplicate another field name or another Prototype name. RPGWIZ will attempt to create unique names, but if for some reason a new Prototype name is duplicated, then you will have to manually change the duplicated name.

Tip: Existing PLIST structures will not be removed from the source member, but can be removed after the conversion using the CLNRPG (Clean RPG Source) command.

Convert GOTO operations (CVTGOTO)

Specifies whether or not GOTO operations should be converted.

***NO**
GOTO operations will not be converted.

***YES**
GOTO operations which jump to the end of subroutines (tag in a ENDSR operation) will be converted. These GOTOs will be converted to:

LEAVESR operations if the target IBM i release is V4R4 or higher and if subroutines are not converted to sub-procedures with CVTSUBR(*NO).

RETURN operations if subroutines are also converted to sub-procedures with CVTSUBR(*YES).

The tag value will not be removed from the ENDSR operation (to accommodate any CABxx operations).

Example Before:

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
C     $EDITCHK      BEGSR
C                   ' '
C                   GOTO       @END
C                   ' '
C     @END          ENDSR
```

Example After:

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
C     $EDITCHK      BEGSR
C                   ' '
C                   LEAVESR
C                   ' '
C     @END          ENDSR
```

Compress Expressions (CMPEXP)

Specifies whether or not expressions should be compressed into the minimum amount of space needed.

***NO**
Expressions will not be compressed.

***YES**
Eligible expressions will be compressed.

This option will remove any extra spaces from each eligible expression, leaving 1 space between variables and operators. Each reformatted expression will then be fitted into the minimum number of lines needed.

Expressions within both extended factor 2 and free-form RPG are eligible for compression.

Example Before:

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
C                   IF     STATE  =  'NE'  OR
C                          STATE  =  'IA'
C                   EVAL   TAXES =  STATETAX +
C                          COUNTYTAX +
C                          CITYTAX
C                   ENDIF
```

Example After:

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
C                   IF     STATE = 'NE' OR STATE = 'IA'
C                   EVAL   TAXES = STATETAX + COUNTYTAX + CITYTAX
C                   ENDIF
```

Considerations

Only the right hand comments (in positions 81-100) for the first line in each expression will be retained. The right hand comments for all subsequent lines of a compressed expression will be removed.

Any in-line comments contained within an eligible expression will be removed.

Any free-form RPG expressions with trailing // comments in the logic area (before position 81) will not be compressed.

Highlight comments (HIGHCOMM)

Specifies whether or not to highlight the comment lines in the source member.

RPGWIZ regards a line as a comment if it has an asterisk in position 7, if it has proceeding slashes (//), if its blank or if it contains the compiler directives of /TITLE, /SPACE or /EJECT.

***NO**
No comment lines will be highlighted. Comment lines which are already highlighted will not be affected.

***YES**
All comment lines will be highlighted. This is achieved by placing a hexadecimal 22 in position 6 of the comment line if the specification type is removed, or position 5 of the comment line if the specification type is not removed, or position 5 if it has a // comment designator.

Example

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
Before
C*
C* Clear the customer number
C*
C                   CLEAR                   CUSTNO

After
C*
C* Clear the customer number
C*
C                   CLEAR                   CUSTNO
```

Considerations

You can optionally use the separate HLTCMT command to highlight all in-line comment lines within a source member. Read about the HLTCMT command later in this document for more information.

The SEUPLUS line command of HC will also allow you to highlight a block of comment lines while in SEU and the SEUPLUS line command of RH will allow you to remove highlighting from a block of comment lines.

Fixed-form comment designator (CMTDESIG)

Specifies the designator to use for indicating an in-line comment within fixed-format specifications. This parameter is only valid in a target release (TGTRLS) of V5R1 and higher.

***LEAVE**
Comment designators are not converted.

***SLASHES**
Comment designators are converted from asterisks (*) to slashes (//).

*SLASHES example:

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
Before
C*
C* Clear the customer number
C*
C                   CLEAR                   CUSTNO
After
 //
 // Clear the customer number
 //
C                   CLEAR                   CUSTNO
```

***ASTERISK**
Comment designators are converted from slashes (//) to asterisks (*).

*ASTERISK example:

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
Before
 //
 // Clear the customer number
 //
C                   CLEAR                   CUSTNO
After
 *
 * Clear the customer number
 *
C                   CLEAR                   CUSTNO
```

Comment specification types (CMTSPECTP)

Specifies whether or not to remove the specification types from comment lines in fixed-format specifications.

Eligible comment lines have either an asterisk in position 7 or contain the compiler directives of /TITLE, /SPACE or /EJECT.

***LEAVE**
The specification types will not be removed from fixed-format comment lines, unless converting to free-form RPG.

***REMOVE**
The specification types will be removed from comment lines.

Example:

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
Before
C*
C* Clear the customer number
C*
C                   CLEAR                   CUSTNO

After
 *
 * Clear the customer number
 *
C                   CLEAR                   CUSTNO
```

Comment designator on blanks (CMTBLANK)

Specifies whether or not to remove the comment designator from blank comment lines. RPGWIZ will recognize both asterisk (*) and slashes (//) comment designators within both fixed-format and free-form RPG.

***LEAVE**
The comment designator will not be removed from blank comment lines.

***REMOVE**
The comment designator will be removed from blank comment lines.

Example before:

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
 *
 * Clear the customer number
 *
C                   CLEAR                   CUSTNO
 *
 * Display the screen
 *
C                   EXFMT     DSP01
```

Example after:

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
 * Clear the customer number

C                   CLEAR                   CUSTNO

 * Display the screen

C                   EXFMT     DSP01
```

Case for specification types (CASESPECTP)

Specifies what type of case conversion to perform on the specification types (position 6) within the source member.

***LEAVE**
The specification types will be left in their current case.

***LOWER**
The specification types will be changed to lower case.

Example:

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
 * Before
C* Clear the customer number
C                   CLEAR                   CUSTNO
C                   MOVE      *BLANKS       FIELD3
 * After
c* Clear the customer number
c                   CLEAR                   CUSTNO
c                   MOVE      *BLANKS       FIELD3
```

***UPPER**
The specification types will be changed to upper case.

Example:

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
 * Before
c* Clear the customer number
c                   CLEAR                   CUSTNO
c                   MOVE      *BLANKS       FIELD3                 03
 * After
C* Clear the customer number
C                   CLEAR                   CUSTNO
C                   MOVE      *BLANKS       FIELD3                 03
```

Case for unchanged logic (CASELOGICO)

Specifies what type of case conversion to perform on any logic (from positions 7-80) that was not changed by RPGWIZ.

The case in constants, comments and array data is never modified.

***LEAVE**
Unchanged logic will be left in its current case.

***LOWER**
Unchanged logic will be converted to lower case.

***UPPER**
Unchanged logic will be converted to upper case.

***MIXED**
Unchanged user-defined file, record and field names will be converted to mixed case (1st letter is capitalized). Unchanged IBM keywords and operations will be converted to upper case.

***MIXED2**
Unchanged user-defined file, record and field names will be converted to mixed case (1st letter is capitalized). Unchanged IBM keywords and operations will also be converted to mixed case, unless they're preceded by an * or % (such as *IN01 or %SUBST).

Example

Before:

```
I..............Namedconstant+++++++++C.........Fldnme......
I              'NEBRASKA'             C         FIELD1

CL0N01N02N03Factor1+++OpcdeFactor2+++ResultLenDHHiLoEq
C* Set the Field Defaults
C                     MOVE 'A'       FIELD2
C                     MOVE FIELD1    FIELD3

...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8
** Array Data
NE
IA
```

*LOWER after:

```
DName++++++++++ETDsFrom+++To/L+++IDc.Keywords++++++++++++++++++++++++
D field1          c                   const('NEBRASKA')

CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
 * Set the Field Defaults
C                   move      'A'           field2
C                   move      field1        field3

...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+...
** Array Data
NE
IA
```

Case for changed and new logic (CASELOGICN)

Specifies what type of case conversion to perform on any logic (from positions 7-80) that was changed or added by RPGWIZ.

The case in constants, comments and array data is not modified.

***LOWER**
Changed and new logic will be converted to lower case.

***UPPER**
Changed and new logic will be converted to upper case.

***MIXED**
Changed and new user-defined file, record and field names will be converted to mixed case(1st letter is capitalized). Changed and new IBM keywords and operations will be converted to upper case.

***MIXED2**
Changed and new user-defined file, record and field names will be converted to mixed case(1st letter is capitalized). Changed and new IBM keywords, built-in functions and operations will also be converted to mixed case.

Case for in-line comments (CASECMTIN)

Specifies what type of case conversion to perform on any in-line comments.

RPGWIZ regards a line as a comment line if it has an asterisk in position 7, if it has proceeding slashes(//), if its blank or if it contains the compiler directives of /TITLE, /SPACE or /EJECT.

***LEAVE**
In-line comments will be left in their current case.

***LOWER**
In-line comments will be converted to lower case.

***UPPER**
In-line comments will be converted to upper case.

Case for right-hand comments (CASECMTRH)

Specifies what type of case conversion to perform on any right-hand comments (positions 81-100).

***LEAVE**
Right-hand comments will be left in their current case.

***LOWER**
Right-hand comments will be converted to lower case.

***UPPER**
Right-hand comments will be converted to upper case.

Document nested logic (DOCNEST)

Specifies whether or not to document the nested logic within the RPG IV source member. Source code positions 1 through 4 will be used to mark the beginning and ending of IF, DO, SELECT, CASxx and FOR structures. Read about the DOCNST command later in this document for more information.

***YES**
All logic structures within the converted source member will be documented.

***NO**
No logic structures within the converted source member will be documented.

Indent nested logic (INDNEST)

Specifies whether or not to indent logic within free-form RPG. This includes indenting logic under IF, DOU, DOW, SELECT, CAS and FOR structures. Read about the INDNST command later in this document for more information.

***INDENT1 to *INDENT9**
Specifies to indent the free-form nested logic from 1 to 9 spaces.

***INDENT0**
Specifies to remove indentation from free form logic.

Source date on converted lines (SRCDAT)

Specifies what value to place in the source date for the converted lines of source code.

The date on a source code line will not be modified when the only change to that line was highlighting the comment, changing it to upper/lower case or documenting the nested logic.

***KEEP**
The source date on converted lines is kept at the original date.

***TODAY**
The source date on converted lines is set to today's date.

***ZEROS**
The source date on converted lines is set to zeros.

Save command settings for job (SAVECMD)

Specifies whether or not to temporarily save the current RPGWIZ parameter values for future conversions in the current job. This is useful when performing multiple conversions and you don't want to permanently change the command defaults.

***NO**
The RPGWIZ parameter values will not be saved for future conversions.

***YES**
The command settings will be saved for the job. The RPGWIZ command will be created in your job's QTEMP library with the defaults set to the current parameter values. To use these saved command settings within the current job, you can prompt the command QTEMP/RPGWIZ from the command line.

Automatic Features

The following conversion features are automatic.

Comparison operations

RPGWIZ moves the IFxx, WHxx, DOxxx, ANDxx and ORxx comparisons to extended factor 2. All comparison codes are converted to their corresponding expression codes (i.e. EQ becomes =, NE becomes <> and so on).

IFxx example

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
 * Before
C     FIELD1         IFEQ      1
C     FIELD2         ANDNE     2
C     FIELD3         ORGE      3
C                    MOVE      *ZEROS         FIELD4
C                    ENDIF
 * After
C                    IF        FIELD1 = 1
C                              AND FIELD2 <> 2
C                              OR FIELD3 >= 3
C                    MOVE      *ZEROS         FIELD4
C                    ENDIF
```

WHENxx (also applies to **WH**xx in RPG/400) example

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
 * Before
C                    SELECT
C*
C     FIELD1         WHENEQ    'A'
C     FIELD2         ORLT      2
C     FIELD3         ANDLE     3
C                    MOVE      *ZEROS         FIELD4
C*
C     FIELD1         WHENEQ    'B'
C                    MOVE      1              FIELD4
C                    ENDSL
 * After
C                    SELECT
C*
C                    WHEN      FIELD1 = 'A'
C                              OR FIELD2 < 2
C                              AND FIELD3 <= 3
C                    MOVE      *ZEROS         FIELD4
C*
C                    WHEN      FIELD1 = 'B'
C                    MOVE      1              FIELD4
C                    ENDSL
```

DOxxx example

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
 * Before
C     FIELD1         DOUGT     10
C     FIELD2         ORLT      2
C*
C     FIELD3         DOWEQ     *ON
C                    MOVE      *OFF          FIELD3
C                    ADD       1             FIELD1
C                    ENDDO
C                    ENDDO
 * After
C                    DOU       FIELD1 > 10
C                              OR FIELD2 < 2
 *
C                    DOW       FIELD3 = *ON
C                    MOVE      *OFF          FIELD3
C                    ADD       1             FIELD1
C                    ENDDO
C                    ENDDO
```

END operations

RPGWIZ converts END operations to their corresponding ENDxx operations.

For instance, an END operation for an IF structure will be converted to an ENDIF or converted to ENDDO for a DO loop.

END examples

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
 * Before
C                   IF        FIELD1 < 10
C                   DOW       FIELD2 = *ON
C                   MOVE      *OFF          FIELD2
C                   END
C                   END
 * After
C                   IF        FIELD1 < 10
C                   DOW       FIELD2 = *ON
C                   MOVE      *OFF          FIELD2
C                   ENDDO
C                   ENDIF
```

SETON, SETOF and COMP operations

RPGWIZ converts SETON, SETOF (or SETOFF) and COMP operations to their corresponding EVAL operations.

If left hand indicators are not requested to be converted; A SETON, SETOF or COMP operation will not be converted to an EVAL if it has a left hand indicator and if it turns on/off more than one indicator.

SETON example

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
 * Before
C                   SETON                                        010203
 * After
C                   EVAL      *IN01 = *ON
C                   EVAL      *IN02 = *ON
C                   EVAL      *IN03 = *ON
```

SETOFF example

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
 * Before
C                   SETOFF                                       010203
 * After
C                   EVAL      *IN01 = *OFF
C                   EVAL      *IN02 = *OFF
C                   EVAL      *IN03 = *OFF
```

COMP examples

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
 * Before
C     FIELD1        COMP      FIELD2                             010203
C*
C     FIELD3        COMP      FIELD4                             70  70
 * After
C                   EVAL      *IN01 = (FIELD1 > FIELD2)
C                   EVAL      *IN02 = (FIELD1 < FIELD2)
C                   EVAL      *IN03 = (FIELD1 = FIELD2)
 *
C                   EVAL      *IN70 = (FIELD3 >= FIELD4)
```

Indicator constants

RPGWIZ converts the constant of '1' to *ON and '0' to *OFF in indicator operations.

The constants of '1' and '0' will not be converted in MOVEA operations.

Example:

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
 * Before
C                   IF        *IN02 = '0'
C                   MOVE      '1'           *IN01
C                   ENDIF
 * After
C                   IF        *IN02 = *OFF
C                   MOVE      *ON           *IN01
C                   ENDIF
```

Other features

RPGWIZ removes the outdated subroutine designator of 'SR' from positions 7 and 8 of the source code.

In the "F" specs, if the "file designation entry" in position 18 is a "C" or "D", which is no longer valid, RPGWIZ will change it to a "F".

In the "F" specs, if the "mode of processing entry" in position 28 is a "R", which is no longer valid, RPGWIZ will clear it out.

Converted source members under 10,000 lines are resequenced by a value of 1. Source members with 10,000 or more lines are resequenced by a value of .10.

Changing RPGWIZ Defaults

The defaults settings for the RPGWIZ command can be modified using the CHGDFT command. Follow these instructions to permanently change these defaults:

RDi users: the default settings for converting to free form are configured in the preferences section within RDi.

1. Prompt (F4) the IBM i command **RPGTOOLBOX/CHGDFT.**
2. Listed on the prompted screen will be the same parameter values as on the RPGWIZ command.
3. Optionally press F1 on any parameter for help.
4. Make the desired changes to the parameter defaults and press Enter.
5. The RPGWIZ command defaults have now been changed.

Note: For any active job(s) which ran the RPGWIZ command with *YES specified on the "Save command settings for job" parameter, a copy of the RPGWIZ command is still in these job(s) QTEMP library. These job(s) will continue to use the settings of their respective QTEMP/RPGWIZ commands (assuming QTEMP is above the RPGTOOLBOX library in the library list) until those jobs terminate or when the RPGWIZ command is deleted from their QTEMP library.

Display Indented Source

The command DSPIND will produce a source member listing for RPGLE and SQLRPGLE member types. Any nested logic will be shown in indented fashion. This report can be viewed or printed.

Both fixed-format and free-form RPG logic is supported. The beginning and ending of control structures (IFs, DOs, FORs, etc.) will be visually connected and the logic within them will be indented. The level of nesting will also be shown on the far right side of the report.

Example Output

```
SEQNBR*...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ..LEVL

  1.00     C                   dou       stoplp = 'DONE'                       B001
  2.00     C                   |         or xx >= 10                              001
  4.00     C                   |  if        ppcode = 'C'                       B002
  5.00     C                   |  |  eval      payment = 'CREDIT CARD'           002
  6.00     C                   |  |  eval      xx = xx + 1                       002
  7.00     C                   |  else                                         X002
  8.00     C                   |  |  eval      payment = 'CASH'                  002
  9.00     C                   |  endif                                        E002
 11.00     C                   enddo                                         E001
```

To produce this report, go into PDM and place the PDM option for DSPIND next to the source member, then press Enter or F4 to prompt. This PDM option was created when you ran the CRTOPT command (explained in the installation section). The default option was "DI". Otherwise prompt the command RPGTOOLBOX/DSPIND.

Prompted Screen

```
                    Display Indented Source (DSPIND)

Type choices, press Enter.

File . . . . . .   QRPGLESRC     Name, QRPGLESRC
  Library  . . . . . .   *LIBL       Name, *LIBL
Member . . . . . .                  Name
Spaces to indent . . . . . .   *INDENT2     *INDENT1, *INDENT2...
Indention character  . . . . . .            Character value
In-line comments . . . . . .   *INTERSECT   *LEAVE, *INTERSECT, *REMOVE
Right-hand comments  . . . . . .   *REMOVE   *LEAVE, *REMOVE
Page headings  . . . . . .   *NO           *NO, *YES
Output . . . . . .   *                     *, *PRINT
```

- Make any necessary changes and press Enter.

Command Parameters

The parameters for the DSPIND command are described below.

File
Specify the name of the source file of the member.

Library
Specify the name of the library where the source file is stored.

Member
Specify the name of the source file member to display.

(continued)

Spaces to indent
Specify the number of spaces to indent any nested logic. Use the special values of *INDENT1 through *INDENT5.

Indention character
Specify the character to use to visually connect the beginning and ending of any control structures (IFs, DOs, FORs, etc.). Recommended to keep the default of a pipe | character.

In-line comments
Specify the operation to perform on in-line comments.

***LEAVE**
In-line comments will be shown on the report and the indention character will NOT intersect these comments.

***INTERSECT**
In-line comments will be shown on the report and the indention character will intersect through them.

***REMOVE**
In-line comments will be removed from the report and only source logic will be shown.

Right-hand comments
Specify the operation to perform on right-hand comments. You may wish to remove right-hand comments from the report so there is more room available on the report for indention.

***LEAVE**
Right-hand comments will be shown on the report.

***REMOVE**
Right-hand comments will be removed from the report.

Page headings
Specify whether or not to show page headings on any pages following the first page. You may choose to suppress these headings so the indented source is easier to follow across pages.

***NO**
Page headings will only be shown on the first page and not on subsequent pages.

***YES**
Page headings will be shown on all pages.

Output
Specify whether to display the report on the screen or print it.

*****
The report will be displayed on the screen.

***PRINT**
The report will be produced into your default output queue.

SEUPLUS

The SEUPLUS block line commands of **DI** (for viewing indented source) and **LI** (for printing indented source) can also be used within SEU for just creating the report for a selected block of source.

Document Nested Logic

The command DOCNST documents the beginning and ending of your IF, DOxx, SELECT, CAS, MONITOR and FOR structures within your RPG IV program. Both fixed-format and free-form logic is documented. This documentation is recorded in positions 1 through 4 of your source code and can be viewed through SEU (Source Entry Utility).

The DOCNST command is located in the RPGTOOLBOX library. The command can be run from a command line or PDM.

The default PDM option for DOCNST is "DN".

Prompted Screen

```
                  Document Nested Logic - Linoma (DOCNST)

Type choices, press Enter.

File . . . . . . . . . . . . . .   QRPGLESRC     Name, QRPGLESRC
  Library  . . . . . . . . . . .     *LIBL       Name, *LIBL
Member . . . . . . . . . . . . .                 Name
Documentation action . . . . . .   *ADD          *ADD, *REMOVE
```

- Specify *ADD to add the documentation and specify *REMOVE to remove the documentation.
- Make any necessary changes and press Enter.

Details

The beginning of a structure is denoted by a "B". An "else" operation is denoted by a "X". The end of a structure is denoted by an "E". The remaining number indicates the level of the structure. For instance, a value of "B002" indicates the beginning of a structure at the second nested level.

Example

```
B001 C                   dou       stoplp = 'DONE'
 001 C                             or xx >= 10
B002 C                   if        ppcode = 'C'
 002 C                   eval      payment = 'CREDIT CARD'
X002 C                   else
 002 C                   eval      payment = 'CASH'
E002 C                   endif
E001 C                   enddo
```

Viewing the Documentation

To view this nested logic documentation, load the source code in SEU (Source Entry Utility) and press F19 to window left. This documentation will be shown on the left side of your calculation specifications. Enter W6 (to window 6) on a sequence number to hide this documentation.

SEUPLUS

The SEUPLUS line command of DN can be used to document a portion of the source code.

Indent Nested Logic

The command INDNST will indent logic within free-form RPG the requested number of spaces. This includes indenting logic under IF, DOU, DOW, SELECT, MONITOR, CAS and FOR structures.

Right-hand comments (in positions 81-100) will be preserved in their current location.

The INDNST command is located in the RPGTOOLBOX library. The command can be run from a command line or PDM.

The default PDM option for INDNST is "IN".

Prompted Screen

```
                  Indent Nested Logic by Linoma (INDNST)

Type choices, press Enter.

 File . . . . . . . . . . . . . . . > QRPGLESRC_    Name, QRPGLESRC
   Library  . . . . . . . . . . . . >   BXTEST____  Name, *LIBL
 Member . . . . . . . . . . . . . . > SUBST _____   Name
 Starting column number . . . . .   *COL8          *COL8, *COL9, *COL10...
 Spaces to indent . . . . . . . .   *INDENT4       *INDENT0, *INDENT1…
```

- The value of *INDENT0 will remove indentation.
- Make any necessary changes and press Enter.

Example

Before:

```
/free
 dou stoplp = 'DONE' or xx >= 10
 if ppcode = 'C'
 eval payment = 'CREDIT CARD'
 else
 eval payment = 'CASH'
 endif
 enddo
/end-free
```

After:

```
/free
 dou stoplp = 'DONE' or xx >= 10
     if ppcode = 'C'
         eval payment = 'CREDIT CARD'
     else
         eval payment = 'CASH'
     endif
 enddo
/end-free
```

SEUPLUS

The SEUPLUS line commands of IN1-IN9 can be used to indent a portion of the source code.

Highlight Comment Lines

The command HLTCMT will highlight in-line comments found within a source member. This command can be used to highlight comments in RPG, DDS, CL and CMD source members.

Highlighting is achieved by placing a hexadecimal 22 in the source line. When highlighting comments in RPG and DDS source code, this hex character will be placed in position 6 if an asterisk comment line does not have a specification type. Otherwise it will be placed in position 6. When highlighting CL and CMD comment lines, the hex character will be placed after the /* comment designator.

To highlight comments in a source member, go into PDM and place the PDM option for HLTCMT next to the source member, then press enter or F4 to prompt. This PDM option was created when you ran the CRTOPT command (explained in the installation section). The default option was "HC".

Otherwise prompt the command RPGTOOLBOX/HLTCMT.

Prompted Screen

```
                    Highlight Comment Lines (HLTCMT)

Type choices, press Enter.

File . . . . . .   QRPGLESRC     Name
  Library  . . . .   *LIBL       Name, *LIBL
Member . . . . . .               Name
Highlight action .   *ADD        *ADD, *REMOVE
```

- Specify *ADD to add highlighting to comment lines and specify *REMOVE to remove the highlighting from comment lines.
- Make any necessary changes and press Enter.

RPG Example after highlighting the comment lines

```
     C*
     C* Set the payment code
     C*
     C                   if        ppcode = 'C'
     C                   eval      payment = 'CREDIT CARD'
     C                   endif
```

Viewing the Highlighted Comment Lines

To view the highlighted comments lines, load the source code in IBM's SEU (Source Entry Utility).

When opening RPGLE and SQLRPGLE source members, SEU will default the display area to start at position 6. Since the hex character may be placed in position 5, you may have to enter W5 (to window 5) on a sequence number to see the highlighting of the comments.

SEUPLUS

The SEUPLUS line command of HC can also be used to highlight a block of comment lines and RH can be used to remove highlighting from a block of comment lines.

Clean RPG Source

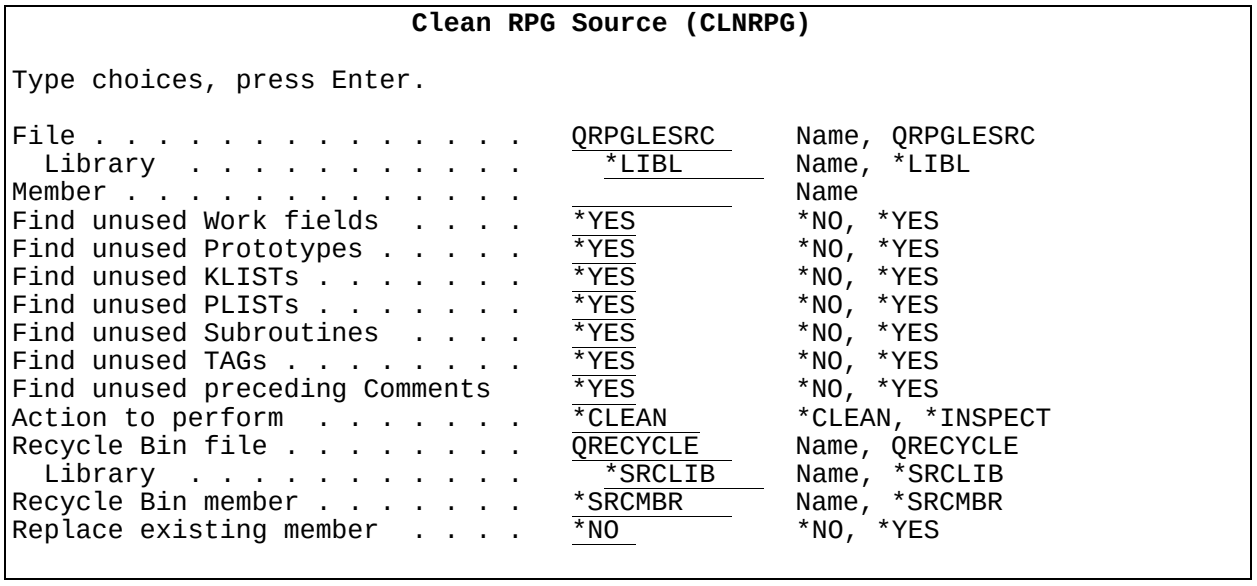
The Clean RPG Source (CLNRPG) command will find, and optionally remove, unused definitions and logic found within RPG IV (RPGLE or SQLRPGLE) source code. CLNRPG can be configured to find unused stand-alone work fields, subroutines, prototypes, key lists, parameter lists and tags.

By removing unused definitions and logic from your RPG programs, these programs will compile faster and run more efficiently. The programs will also be easier to maintain and test since programmers will not have to work with unused source code.

The CLNRPG command has two different actions. The first action called *INSPECT will generate a report of unused definitions and logic in the program, but will not remove any source code. The second action called *CLEAN will generate a report and will additionally remove the unused definitions and logic from the source code. The removed definitions and logic will be placed into another source file (Recycle Bin) for backup purposes.

The CLNRPG command is located in the RPGTOOLBOX library. The command can be run from a command line or PDM.

The default PDM option for CLNRPG is "CR".



Clean RPG Source (CLNRPG)

Type choices, press Enter.

File	QRPGLESRC	Name, QRPGLESRC
Library	*LIBL	Name, *LIBL
Member		Name
Find unused Work fields	*YES	*NO, *YES
Find unused Prototypes	*YES	*NO, *YES
Find unused KLISTs	*YES	*NO, *YES
Find unused PLISTs	*YES	*NO, *YES
Find unused Subroutines	*YES	*NO, *YES
Find unused TAGs	*YES	*NO, *YES
Find unused preceding Comments	*YES	*NO, *YES
Action to perform	*CLEAN	*CLEAN, *INSPECT
Recycle Bin file	QRECYCLE	Name, QRECYCLE
Library	*SRCLIB	Name, *SRCLIB
Recycle Bin member	*SRCMBR	Name, *SRCMBR
Replace existing member	*NO	*NO, *YES

- Press F1 on any parameter for additional help.
- Make any necessary changes and press Enter.

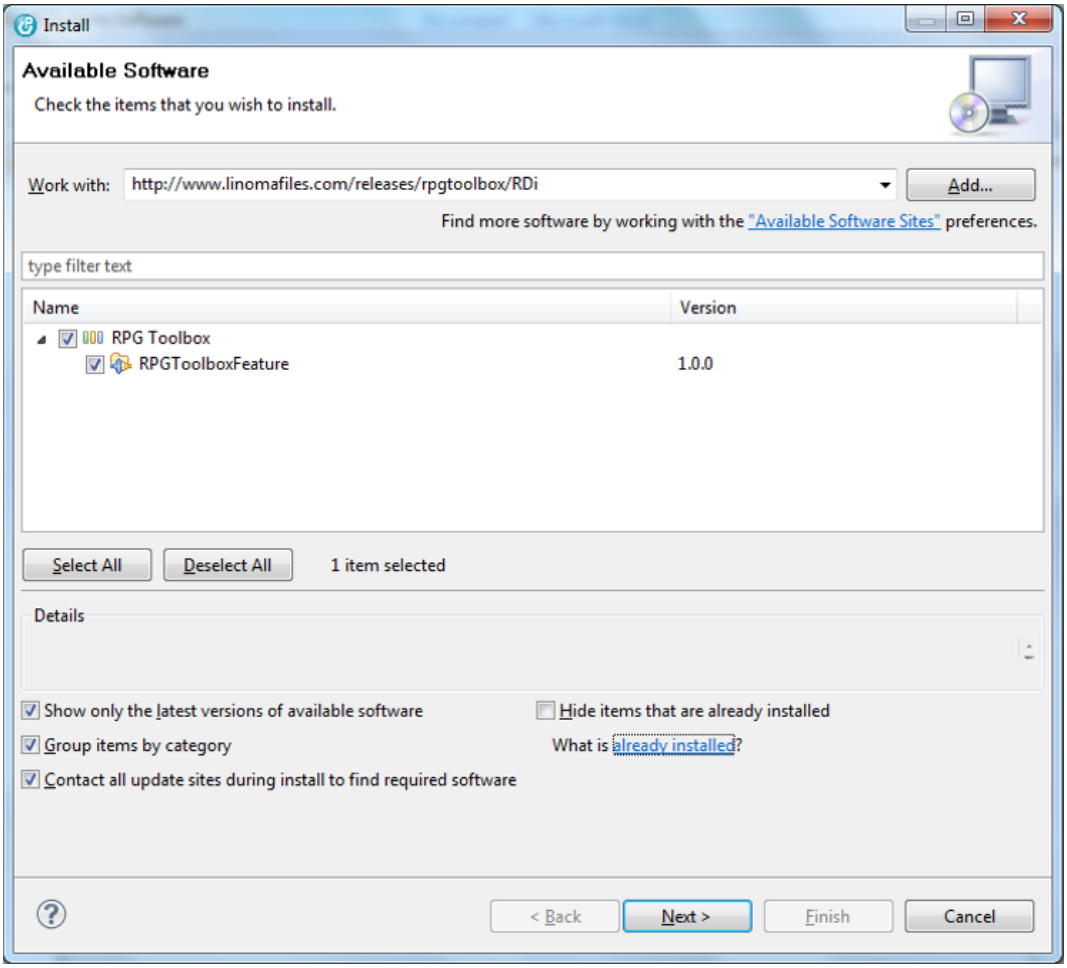
RPG Toolbox RDi Plugin

Linoma's RPG Toolbox RDi Plugin allows you to convert RPG source into free form syntax, as well as indent nested logic within the Rational Developer for IBM i (RDi) graphical development environment. The plugin adds new items to the RDi context menus, allowing you to take action on entire RPG source members or just blocks of selected lines of source code.

RDi Plugin - Installation

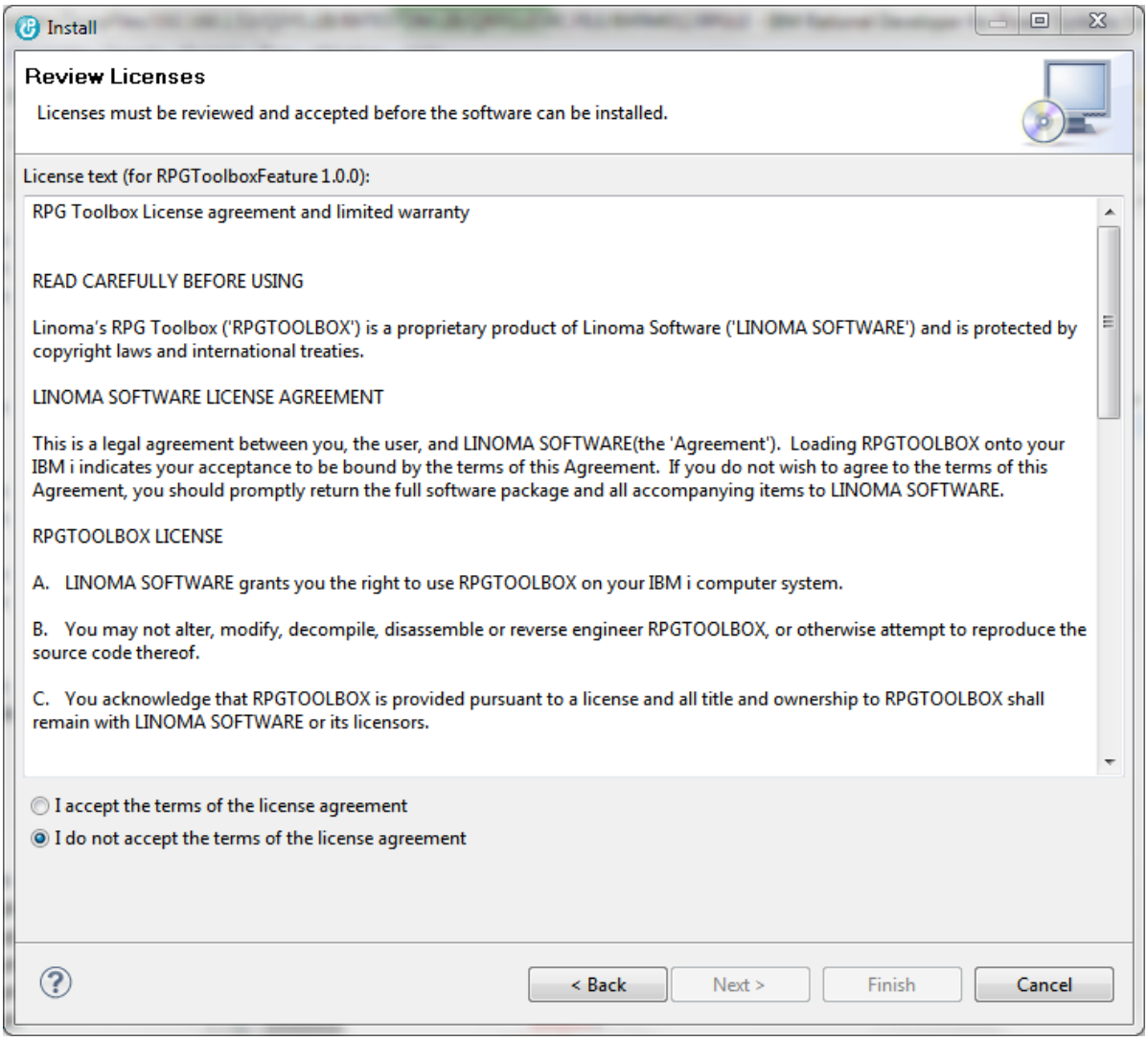
Use the following procedures to install the RPG Toolbox Plugin for RDi versions 8.0 or 9.0:

1. Change the RDi shortcut to run as an administrator by following these steps:
   a. Right-click on the RDi shortcut and choose properties.
   b. On the Shortcut tab, select the Advanced button.
   c. Select the checkbox of **Run as administrator** and click OK.
2. Run the RDi shortcut.
3. From the **Help** menu, select **Install New Software**.
4. In the "Work with" field, specify the url to the RDi install package and then click Add. The install package is located at: **http://www.linomafiles.com/releases/rpgtoolbox/RDi**
5. The **RPG Toolbox** plugin appears. Click on the **Select All** button or expand the **RPG Toolbox,** select the **RPGToolboxFeature,** and then click **Next**.

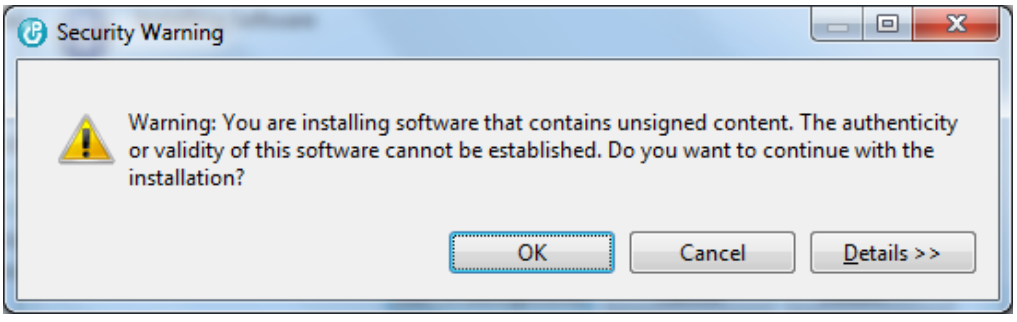


6. The **Install Details** screen appears. Click **Next**.

7. The **Review Licenses** screen appears. If the license agreement is satisfactory, then accept the agreement and click **Finish**.



8. A **Security Warning** message will appear. Click **Ok**.

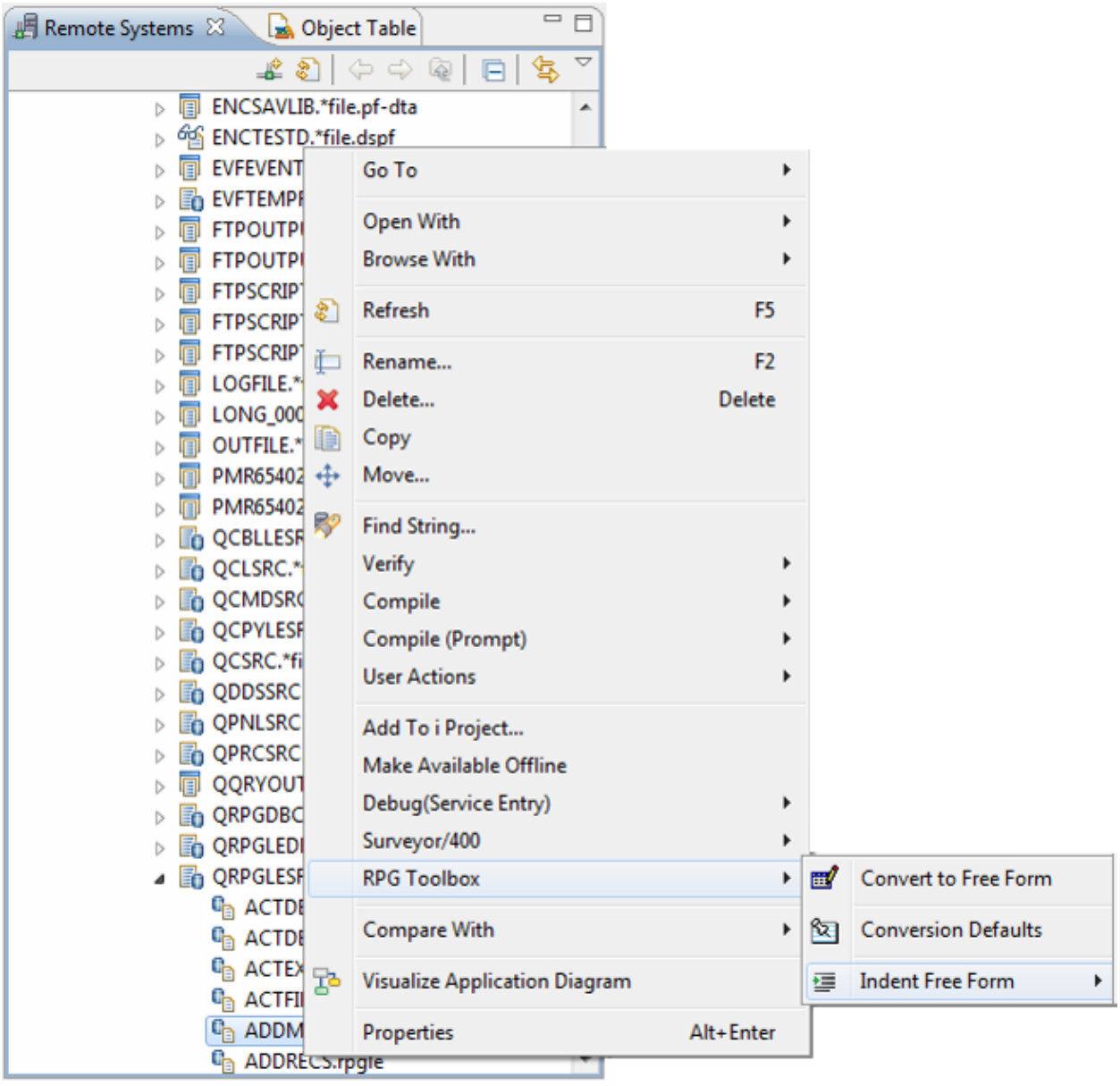


The plugin is now installed. Read the instructions on the following pages to learn how to use the plugin.

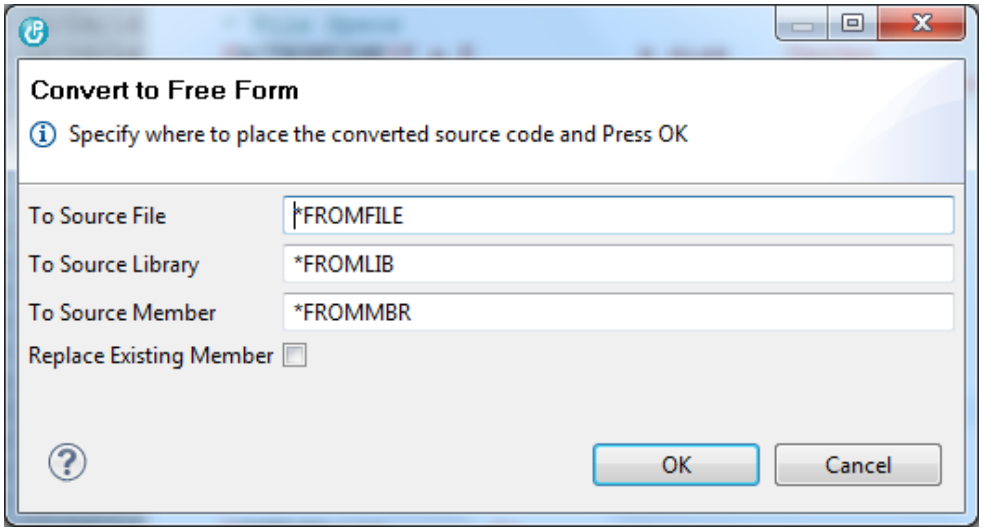
RDi Plugin - Converting a RPG Source Member to Free Form

The RPG Toolbox RDi Plugin can be used to convert RPG source members to free form syntax. The selected member must be one of the following types: RPG, RPT, RPG38, RPGLE, SQLRPG or SQLRPGLE. Perform the following steps to convert an entire RPG source member:

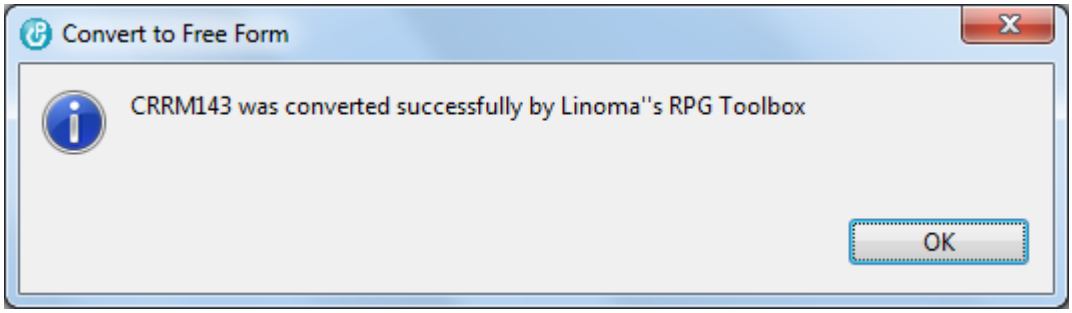
1. From the **Remote Systems Explorer**, right-click on the RPG source member to convert.
2. Select **RPG Toolbox** and then choose **Convert to Free Form**.



3. The **Convert to Free Form** window appears. Specify the source member to place the converted source into, then click **OK**. Important: If you choose to replace the existing member, make sure to have a backup of that source member before proceeding.



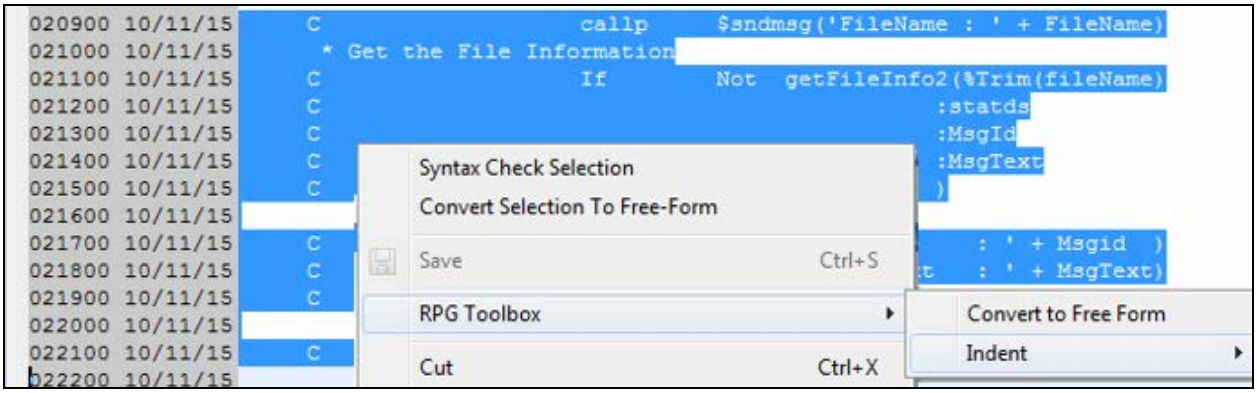
4. A summary window will show the details of the conversion process. Click **OK** to close the dialog.



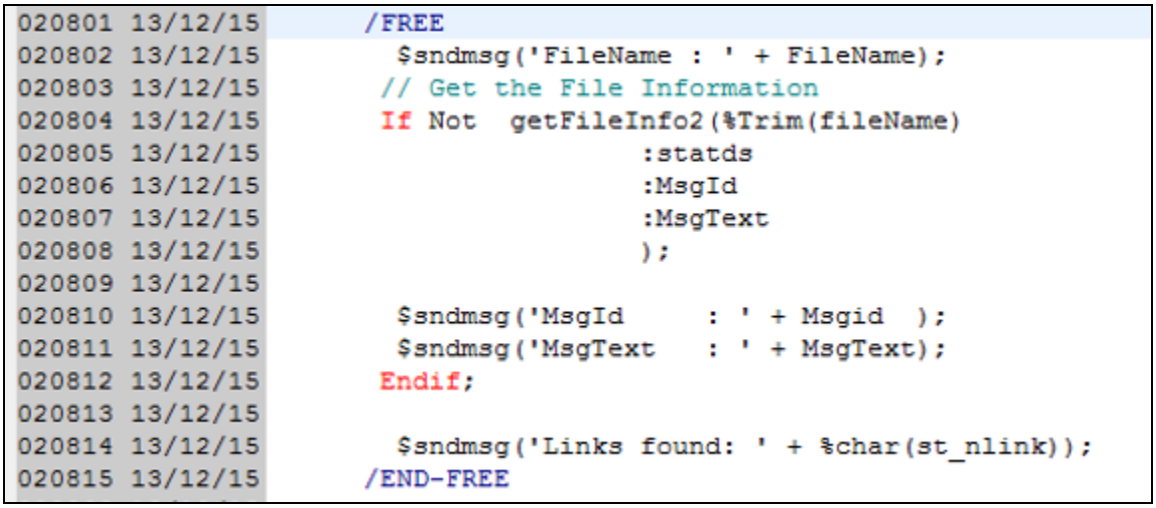
RDi Plugin - Converting a Block of Source to Free Form

From within the LPEX Editor, the plugin allows you to convert a block of source to Free Form syntax. Use the following steps to perform the conversion:

1. Highlight a block of source lines from within the LPEX editor.



2. Right-click on the highlighted lines to bring up the context menu. Then point to **RPG Toolbox** and click **Convert to Free Form**. The selected lines will be converted to free form syntax.

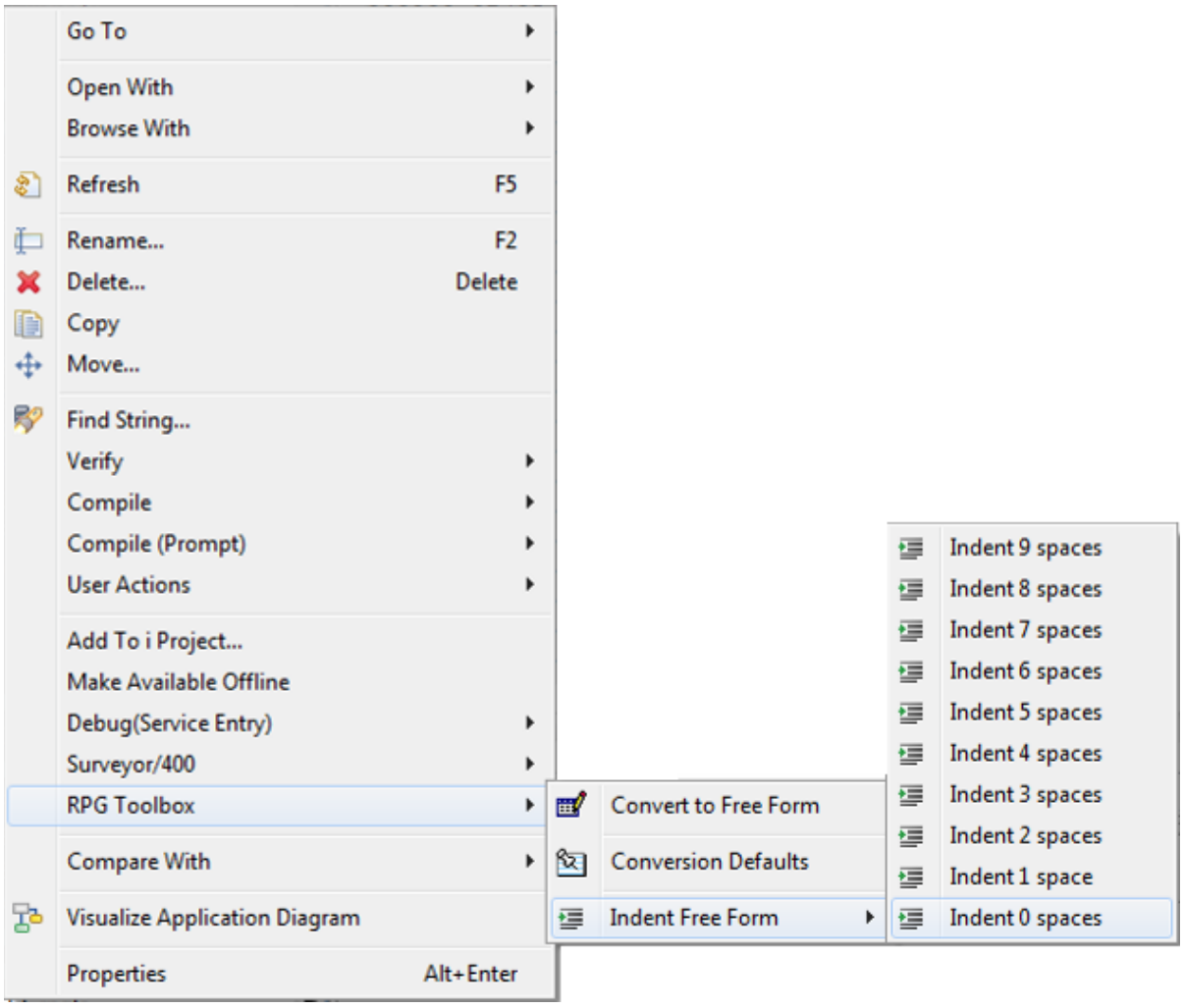


A summary message will appear in the LPEX Editor Message line.

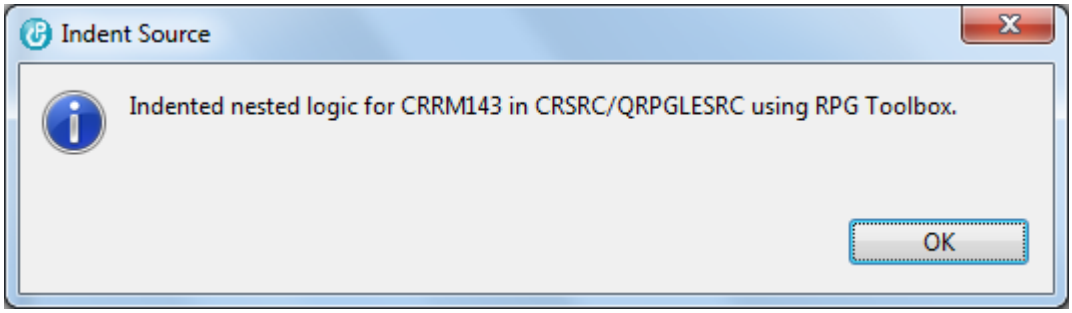
RDi Plugin - Indenting Free Form Nested Logic

The RPG Toolbox RDi plugin can be used to indent free form nested logic in RPGLE and SQLRPGLE source members. This includes indenting logic under IF, DOU, DOW, SELECT, MONITOR, CAS and FOR structures. The amount of indention can be from 0 to 9 spaces. Perform the following steps to indent all the nested logic within a source member:

1. From the **Remote Systems Explorer**, right-click on the RPG source member to convert.
2. Select **RPG Toolbox** and then choose **Indent Free Form**.
3. Specify the number of spaces to indent the nested logic.



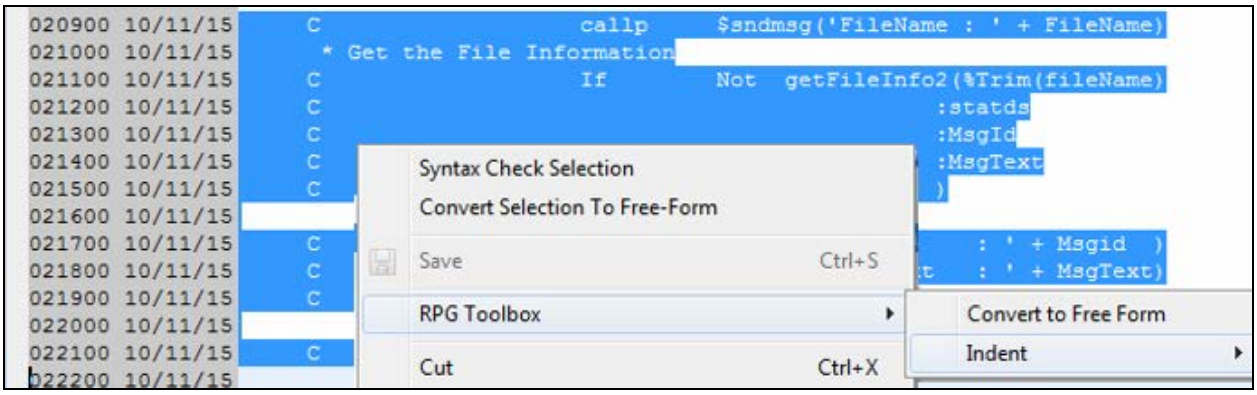
4. A summary window will show the details of the indention. Click **OK** to close the dialog.



RDi Plugin - Indenting a Block of Free Form Nested Logic

The RPG Toolbox RDi Plugin allows you to indent a selected block of nested logic within the LPEX Editor. Use the following steps to perform the indention:

1. Highlight a block of lines from within the LPEX editor.
2. Right-click on the highlighted lines to bring up the context menu. Then point to **RPG Toolbox,** click **Indent** and choose the number of spaces to indent.

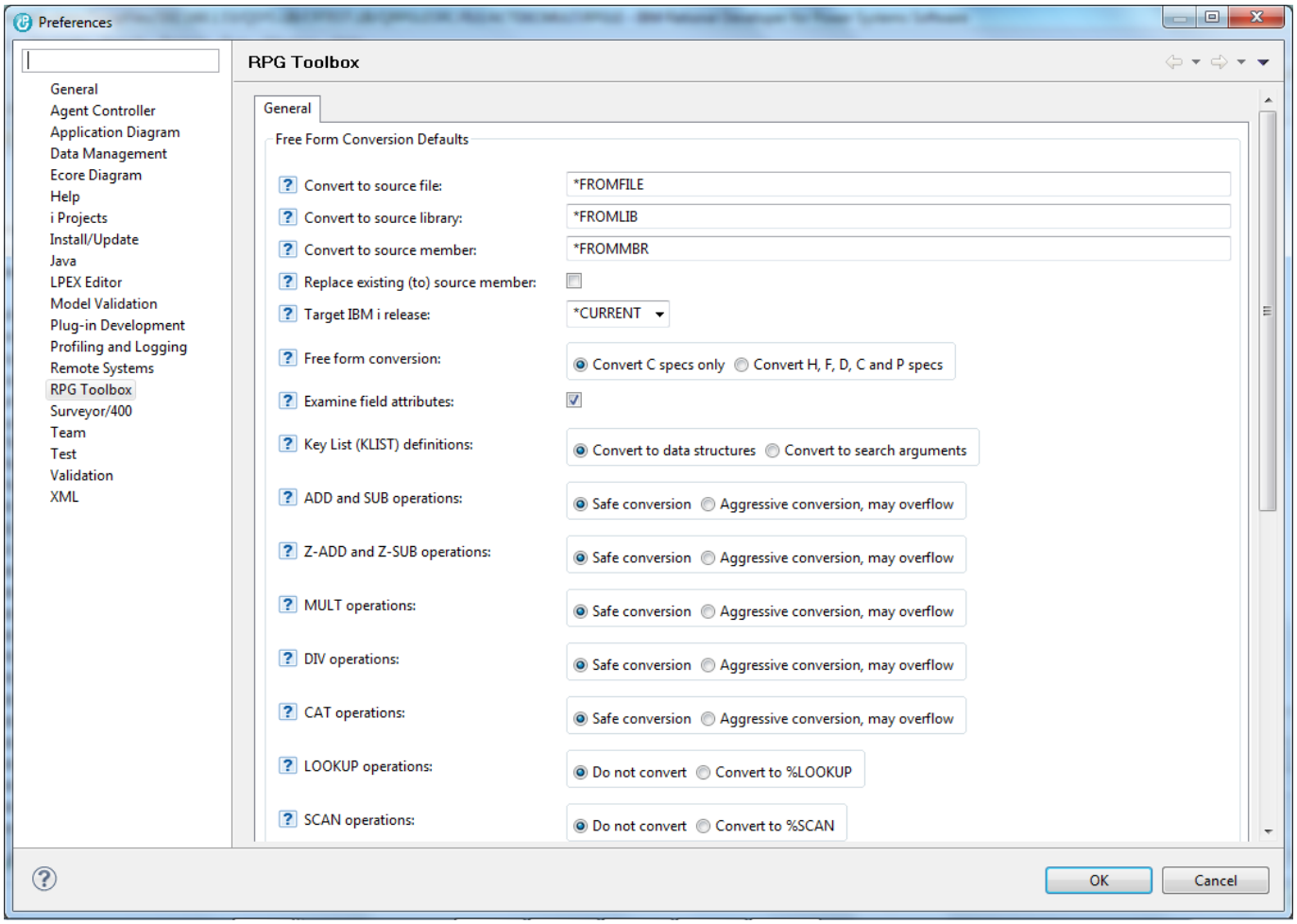


A summary message will appear in the LPEX Editor Message line.

RDi Plugin - Setting Preferences for the Free Form Conversion

Use the following steps to configure the RPG Toolbox (RPGWIZ) preferences for the free form conversion in RDi.

1. Launch the RDi Software.
2. From the **Window** menu, select **Preferences**. Select **RPG Toolbox** from the left hand navigation. Alternatively, you can set the preferences by selecting **Conversion Defaults** from the RPG Toolbox right-click menu on a source member.
3. The current **Free Form Conversion Defaults** will appear.
4. You can view detailed help on any of the settings by clicking on the icon next to the field.
5. Make any changes to the default field settings and then click on the **OK** button to save the preferences**.**



RDi Plugin - Advanced Free Form Conversion Preferences

You can specify advanced preferences for the free form conversion by editing the file named **com.linoma.rpgtoolbox.prefs** located in your workspace. The default values can be overridden by specifying the field name and the new default value.

The field names in the preferences file correspond with the parameter values on the RPGWIZ command. For example, if you wish to not convert GOTO operations, you would specify cvtgoto=*NO in the preferences file.

The following table lists the field settings you can configure in the preferences file:

RPGToolbox (RPGWIZ) Defaults	Field Name	Default Value
Expand copy members	expcpy	*NO
Redefine data structures	redefineds	*YES
Redefine *LIKE DEFN fields	likefld	*YES
Redefine calc. defined fields	calcfld	*YES
Convert left hand indicators	cvtleft	*YES
Convert opcodes to BIFs	opcodebif	*YES
Insert file I/O BIFs	filebif	*YES
Convert MOVE(L)s having *BLANK	cvtmovebl	*EVAL
Convert MOVE(L)s having *ZERO	cvtmovezr	*EVAL
Convert MOVEs having data	cvtmover	*EVAL
Convert MOVELs having data	cvtmovel	*EVAL
Convert MOVEA operations	cvtmovea	*YES
Convert CASxx operations	cvtcas	*YES
Convert DOs to FORs	cvtdo	*YES
Convert GOTO operations	cvtgoto	*YES
Highlight comments	highcomm	*NO
Fixed-form comment designator	cmtdesig	*SLASHES
Case for specification types	cmtspectp	*REMOVE
Comment designator on blanks	cmtblank	*REMOVE
Case for specification types	casespectp	*LEAVE
Document nested logic	docnest	*NO
Save command settings for job	savecmd	*NO

SEU PLUS

Introduction

SEUPLUS dramatically enhances IBM's Source Entry Utility (SEU) with over 70 new SEU line commands and function keys, including features for:

- Developing and maintaining source code faster and easier
- Inserting Snippets of useful source code
- Modernizing RPG source with the latest syntax available
- Coloring, highlighting and underlining source
- Converting source to upper or lower case
- Examining variables, key lists, parameter lists, used indicators and copy books
- Displaying the last compile status and any compile errors
- Saving and retrieving source from Toolbox Memory (like Window's Clipboard)
- Displaying both RPG fixed and free-form source in indented fashion
- Documenting nested logic by placing labels in positions 1 through 4
- Indenting nested RPG free-form logic a specified number of spaces
- Executing popular IBM i commands

An on-line Calculator is even included.

You can also add your own custom SEU line commands for launching IBM i commands and for inserting pre-defined source code (snippets).

Activating

To take advantage of SEUPLUS, you must specify an exit program for SEU. SEUPLUS can either be activated on an individual or global basis. Please follow the instructions below to activate it:

Individual

To make SEU Plus available to just your user id:

- Start IBM's SEU (**STRSEU**)
- Press **F13** to change your session defaults
- **Page down** to the second screen
- Key in **SEUPLUS** as the User exit program, then press Enter
- Key in **RPGTOOLBOX** as the library, then press Enter.
- Each developer that wants SEUPLUS activated must go through these steps.

Global

To make SEUPLUS available to all developers on the IBM i, run the command:

ADDEXITPGM EXITPNT(QIBM_QSU_LCMD) FORMAT(EXTP0100) PGMNBR(*LOW) PGM(RPGTOOLBOX/SEUPLUS)

SEUPLUS Quick Reference

The following is a summary of the custom SEU line commands found in SEUPLUS.

Help

?	Show Valid Line Commands

Editing Source

BR	Break a line into two lines (or press **F8**)
CMT	Change Line to Comment or Reactivate
CMTB	Change Block to Comments or Reactivate
COM	Compact Free-Form or Extended Factor 2
E	Edit Free-Form or Extended Factor 2 – Line
EE	Edit Free-Form or Extended Factor 2 – Block
IE	Insert Free-Form Operation
IEX	Insert Extended Factor 2 Expression
MRK	Modification Marker on Line
MRKB	Modification Marker on Block
SORT	Sort lines in ascending order

Inserting Snippets of Source Code

IX	Insert Snippet Source Lines
IXC	Insert a Snippet into the Current Line
I*	Insert Comments – Asterisk (*)
I/	Insert Comments – Slashes (//)
I/*	Insert Comments – Slash Asterisk (/* */)

Coloring Source

CB	Blue
CBI	Blue and Reverse Image
CBU	Blue and Underline
CD	Red
CDI	Red and Reverse Image
CDU	Red and Underline
CI	Reverse Image
CN	Normal
CP	Pink
CPI	Pink and Reverse Image
CPU	Pink and Underline
CT	Turquoise
CTI	Turquoise and Reverse Image
CU	Underline
CW	White
CWI	White and Reverse Image
CWU	White and Underline
CY	Yellow
CYI	Yellow and Reverse Image
CYU	Yellow and Underline

* use prefix CC to color a block (i.e. CCB to color block blue)

Highlighting Comment Lines

HC	Highlight Comment Lines
RH	Remove Highlighting from Comment Lines

Case Conversion

LO	Lower Case Line
LOB	Lower Case Block
MX	Mixed Case Line
MXB	Mixed Case Block
UP	Upper Case Line
UPB	Upper Case Block

Modernizing RPG

Z	RPG Wizard on Line (no prompt)
ZZ	RPG Wizard on Block (no prompt)
ZP	RPG Wizard on Line (prompt)
ZZP	RPG Wizard on Block (prompt)

Examining

RC	Retrieve Compile Information
WC	Work with Compile Information
RV	Retrieve Variable Attributes
RVK	Retrieve Variables, KLISTs and PLISTs
V V	View Variable Attributes (or press **F7**)
VCB	View Copy Book Source
V I	View Indicators

Toolbox Memory

CM	Copy Line into Memory
CCM	Copy Block into Memory
DM	Delete Line into Memory
DDM	Delete Block into Memory
AM	Insert Memory After Current Line
BM	Insert Memory Before Current Line
RM	Replace Line with Memory
RRM	Replace Block with Memory
WM	Work with Toolbox Memory
TM	Toggles Memory between Add & Replace
CLRM	Clear Memory

Nesting

DI	Display Indented Source
LI	List Indented Source
DN	Document Nested Logic
RN	Remove Nested Logic Documentation
IN0-IN9	Indent Free-Form Nested Logic 0 to 9 spaces

Executing IBM i commands

ACT	Work with Active Jobs
CPYSRC	Copy Source Member
DFD	Display File Description
LOG	Display JOB LOG
MSG	Display Messages
OPR	Display QSYSOPR message queue
SBM	Work with Submitted Jobs
SFE	Surveyor/400 File Editor
SOP	Surveyor/400 Object Properties
SPL	Work with Spool Files

Miscellaneous

CALC	On-line calculator
RESET	Resets Linoma's Toolbox
USRDFT	Work with User Defaults

Function keys

F7	View Variable Attributes (same as **V V**)
F8	Break a line into two lines (same as **BR**)

Entering Line Commands

A SEUPLUS line command can be entered anywhere on a statement number within SEU.

In the example below, the **CB** line command is entered to color the 2nd line blue and the **CW** line command is entered to color the 3rd line white:

```
Columns . . . :    1 100                                        Edit
 SEU==> ____________________________________________________________
0091.00      C                   if        extfnd = *off
CB92.00      C                   eval      extfnd = *on
009CW00      C                   else
```

As illustrated above, multiple line commands can often be entered at the same time (depending on the operation to perform).

* Please note that SEUPLUS line commands cannot be entered on the SEU==> command line on the top of the screen.

While in the trial period
When you first run a SEUPLUS line command, you will be presented with a screen indicating the expiration date of the trial. You can then press Enter to proceed.

On-line Help

Enter the line command of **?** or **HELP** within SEU to view all the available line commands provided by SEUPLUS.

A line command can be selected from the HELP screen. However, once you memorize the line command, you can simply enter it on a SEU line without having to select it from the HELP screen.

Screen shown after entering the **?** line command:

```
                          SEU LINE COMMANDS

Type option, press Enter.
  1=Select   2=Change   4=Delete   5=Help   8=Display details

Opt Command Description                                 Comments
 _  ?       Show Valid Line Commands                    Same as HELP command
 _  ACT     Work with Active Jobs                       WRKACTJOB
 _  AM      Insert Memory After                         After current line
 _  BM      Insert Memory Before                        Before current line
 _  BR      Break a line into two lines                 Or use F8
 _  CALC    On-Line Calculator
 _  CB      Color Line - Blue                           hex 3A
 _  CBI     Color Line - Blue and Reverse Image         hex 3B
 _  CBU     Color Line - Blue and Underline             hex 3E
 _  CCB     Color Block - Blue                          hex 3A
 _  CCBI    Color Block - Blue and Reverse Image        hex 3B
 _  CCBU    Color Block - Blue and Underline            hex 3E
                                                                  More...
F3=Exit   F5=Refresh   F6=Create   F12=Cancel
```

Option 1 = Select
To select a command to execute, enter option **1** next to the line command.

Option 5 = Help
To get additional help text for any command, enter option **5** next to the line command and press Enter.

Screen shown after entering option 5 next to the BR line command:

```
                          LINE COMMAND HELP

Help for: BR       Break a line into two lines

You can break a line into two lines by either 1) Position the cursor
to the breaking point and press F8 or 2) Key in the line command BR,
then position the cursor to the breaking point and press Enter.

If the break is performed within a RPG IV extended factor 2
expression, the second line will start in extended factor 2.
```

Editing Source

Listed below are the SEUPLUS line commands for editing source:

BR - Break a Line into Two Lines (or press F8)

Break a line into two lines, by either:
1. Position the cursor to the breaking point and press F8 or
2. Key in the line command BR, then position the cursor to the breaking point and press Enter.

If the break is performed within a RPG IV extended factor 2 expression, the second line will start in extended factor 2.

CMT - Change Line to Comment or Reactivate

Changes a source line to either a comment or an active line, depending on its current status.

Works with RPG fixed-format, RPG free-format, CL, CMD and DDS source.

When used on an active line (non-comment), this command will change the line to a comment and colors it red. If specified, a user defined modification marker will also be placed in positions 1-5 of the source code. If you want to change the default color (from red) or change the modification marker value for the CMT command, change the user defaults by running the USRDFT line command.

When used on an existing comment line, the CMT command will remove the comment designator and reactivate it. If specified, a user defined modification marker will also be placed in positions 1-5 of the source code.

RPG example of commenting out lines using the CMT command with a modification marker:

Before

```
     C            TEST1     IFEQ    'NEB'
     C                      MOVE    REG      STS
     C                      END
```

After

```
RAL   *C            TEST1     IFEQ    'NEB'
RAL   *C                      MOVE    REG      STS
RAL   *C                      END
```

CL example of commenting out lines using the CMT command:

Before

```
       DCL        &VERSION *CHAR 10
       RTVDTAARA  DTAARA(QTEMP/VERSION) RTNVAR(&VERSION)
```

After

```
/*     DCL        &VERSION *CHAR 10 */
/*     RTVDTAARA  DTAARA(QTEMP/VERSION) RTNVAR(&VERSION) */
```

CMTB - Change Block to Comments or Reactivate

Changes a block of source lines to either a comments or active lines, depending on their current status.

Works with RPG fixed-format, RPG free-format, CL, CMD and DDS source.

When used on an active line (non-comment), this command will change the line to a comment and colors it red. If specified, a user defined modification marker will also be placed in positions 1-5 of the source code. If you want to change the default color (from red) or change the modification marker value for the CMTB command, change the user defaults by running the USRDFT line command.

When used on an existing comment line, the CMTB command will remove the comment designator and reactivate it. If specified, a user defined modification marker will also be placed in positions 1-5 of the source code.

COM - Compact Free-Form or Extended Factor 2

Compacts either a free form operation or a fixed format extended factor 2 expression into the available space.

Example:

```
Before              After
-----------------   -----------------
if  a = b or        if a = b or c = d
    c = d
```

Key the compact line command at the start of the block and key it again at the end of the block to compact, then press Enter.

Optionally position the cursor to the starting point of compaction before pressing Enter.

If the display mode is set to 80 columns, the expression will be formatted so it doesn't extend beyond position 76. Otherwise if the display mode is 132 columns, the expression may fill up to position 80.

E - Edit Free-Form or Extended Factor 2 Line

Use this line command to either:

1. Edit a Free Form operation which only occupies one line OR
2. Edit a Fixed Format operation which uses the extended factor 2 expression area (i.e. EVAL) and only occupies one line.

An expression up to 370 characters in length may be edited and this command will automatically format the expression to fit into one or more lines. This line command additionally splits and hyphenates constants which carry over.

If the display mode is set to 80 columns, the expression will be formatted so it doesn't extend beyond position 76. Otherwise if the display mode is 132 columns, the expression may fill up to position 80.

Refer to the **IE** and **IEX** line commands for an example of the edit screen.

EE - Edit Free-Form or Extended Factor 2 Block

Use this command to either

1. Edit a Free Form operation which occupies multiple lines or
2. Edit a Fixed Format operation which uses the extended factor 2 expression area (i.e. EVAL) and occupies multiple lines.

An expression up to 370 characters in length may be edited and this command will automatically format the expression to fit into one or more lines. It also splits and hyphenates constants which carry over.

Key the EE line command at the start of the block and key it again at the end of the block to edit, then press Enter.

Refer to the **IE** and **IEX** line commands for an example of the edit screen.

IE - Insert Free-Form Operation

A free form operation up to 370 characters in length may be entered and the command will automatically format the operation to fit into one or more lines. It also splits and hyphenates constants which carry over.

Example:

```
OrderTaxes = (OrderNetAmount * StateRate) + (OrderNetAmount * CityRate)_____
_____
_____
_____
Cmt:  Calculate the taxes_____

F7=Insert Snippet   F12=Cancel
```

To insert a source code snippet within the edit screen, position the cursor to the point of insertion and press **F7**.

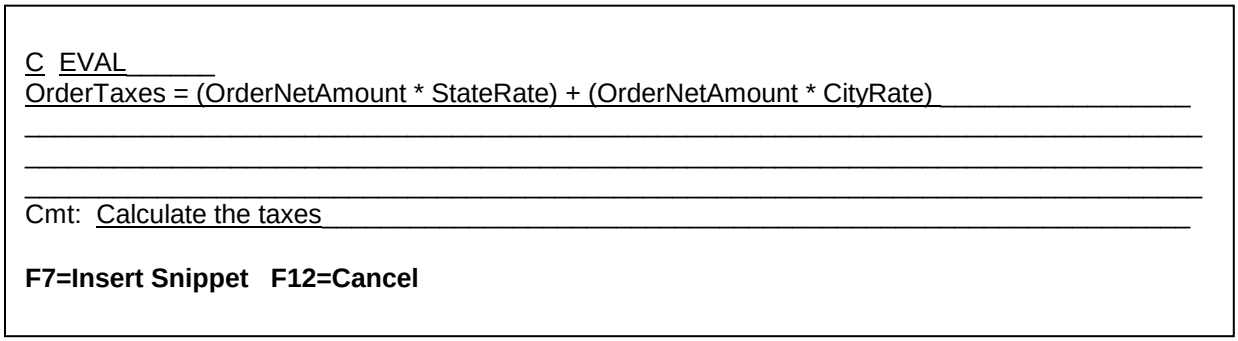
The current indentation will be maintained, unless you position the cursor to a valid starting position before running the command.

IEX - Insert Extended Factor 2 Expression

Allows you to inserts an operation with its corresponding expression in extended factor 2. Supported operations include CALLP, DOU, DOW, ELSEIF, EVAL, FOR, IF, RETURN and WHEN.

An expression up to 370 characters in length may be entered and the command will automatically format the expression to fit into one or more lines of extended factor 2. It also splits and hyphenates constants which carry over.

Example:



To insert a source code snippet within the edit screen, position the cursor to the point of insertion and press **F7**.

MRK or MRKB – Modification Marker

Adds the user's modification marker to positions 1-5 of a single source line (MRK) or a block of source lines (MRKB). These commands work with both RPG and DDS source.

If you want to change the modification marker value for the MRK and MRKB commands, change the user defaults by running the USRDFT line command.

SORT - Sort lines in ascending order,

Sorts a block of lines into ascending order. A potential use for this command is for sorting stand-alone fields defined in RPG IV's Definition specifications.

Key the SORT line command at the start of the block and key it again at the end of the block to sort, then press Enter.

By default, the sort criteria starts in position 6 of your source code. To change the starting position for the sort criteria, position your cursor to the desired starting point before pressing Enter.

Example of sorting stand-alone work fields in the Definition specifications:

```
Before
-----------------------------------
 D CUSTOMER        s              1
 D WORKORDER       s              1
 D APPLICANT       s              1

After
-----------------------------------
 D APPLICANT       s              1
 D CUSTOMER        s              1
 D WORKORDER       s              1
```

Inserting Snippets of Source Code

Snippets of predefined source code can be entered into the current source member. Refer to the Snippets section later in this document for complete details on how to create, change and use Snippets. To insert a snippet, enter one of the following line commands:

IX - Insert Snippet Source Lines

Inserts a Snippet of source after the current line. You will be prompted for the snippet name when you execute this command. If you do not know the name of the snippet, press F4 to perform a search/select.

IXC - Insert a Snippet into the Current Line

Inserts a Snippet into the current source line. You will be prompted for the snippet name when you execute this command. If you do not know the name of the snippet, press F4 to perform a search/select.

Insert the snippet into the current statement by keying in the line command IXC, then position the cursor to the point of insertion and press Enter.

* Shortcut: If you frequently use this line command, you can save keystrokes by assigning this command to function key F7 or F8. Read about SEUPLUS Function Keys later in this document.

I* - Insert Comments using Asterisks (*)

Inserts 3 highlighted comment lines into RPG or DDS source code using asterisks (*). Uses the COMMENT1 snippet.

I/ - Insert Comments using Slashes (//)

Inserts 3 highlighted comment lines into RPG free-form source code using slashes (//). Uses the COMMENT2 snippet.

I/* - Insert Comments using Slash/Asterisk (/*)

Inserts 3 highlighted comment lines into CL or CMD source code using slash/asterisk (/* */). Uses the COMMENT3 snippet.

Coloring a Line

Source lines can be colored, underlined and/or reverse-imaged by entering one of the following SEU line commands:

CB	Blue
CBI	Blue and Reverse Image
CBU	Blue and Underline
CD	Red
CDI	Red and Reverse Image
CDU	Red and Underline
CI	Reverse Image
CN	Normal
CP	Pink
CPI	Pink and Reverse Image
CPU	Pink and Underline
CT	Turquoise
CTI	Turquoise and Reverse Image
CU	Underline
CW	White
CWI	White and Reverse Image
CWU	White and Underline
CY	Yellow
CYI	Yellow and Reverse Image
CYU	Yellow and Underline

Any RPG or DDS source lines can be colored. Comment lines within CL and CMD source can also be colored.

The source line is colored by placing a special hex character into the source.

When coloring lines in RPG and DDS, the hex character will be placed in position 6 for asterisk comment lines without a specification type and in position 5 for all other lines. When coloring CL or CMD comment lines, the hex character will be placed after the /* comment designator.

You can optionally color just a portion of a comment by positioning to the desired starting point before pressing Enter.

Multiple color line commands can be entered at one time.

The hex character must occupy a position within the source line and therefore will replace the existing character at the designated position.

After coloring lines within RPGLE or SQLRPGLE source member types, you may have to enter W5 (to window 5) to see the color since the hex character must be within the display area.

* Shortcut: If you frequently use a particular color command, you can save keystrokes by assigning the line command to function key F7 or F8. Read about SEUPLUS Function Keys later in this document.

You can use the line command of CN to remove the color from a source line.

Coloring a Block of Lines

A block of source lines can be colored, underlined or reverse-imaged by entering one of the following SEU line commands at the start of the block and again at the end of the block:

CCB	Blue
CCBI	Blue and Reverse Image
CCBU	Blue and Underline
CCD	Red
CCDI	Red and Reverse Image
CCDU	Red and Underline
CCI	Reverse Image
CCN	Normal
CCP	Pink
CCPI	Pink and Reverse Image
CCPU	Pink and Underline
CCT	Turquoise
CCTI	Turquoise and Reverse Image
CCU	Underline
CCW	White
CCWI	White and Reverse Image
CCWU	White and Underline
CCY	Yellow
CCYI	Yellow and Reverse Image
CCYU	Yellow and Underline

Any RPG or DDS source lines can be colored. Comment lines within CL and CMD source can also be colored.

The source lines in the block are colored by placing a special hex character into the source.

When coloring lines in RPG and DDS, the hex character will be placed in position 6 for asterisk comment lines without a specification type and in position 5 for all other lines. When coloring CL or CMD comment lines, the hex character will be placed after the /* comment designator.

You can optionally color just a portion of comment lines in a block by positioning to the desired starting point before pressing Enter.

The hex character must occupy a position within the source lines and therefore will replace the existing character at the designated position.

After coloring lines within RPGLE or SQLRPGLE source member types, you may have to enter W5 (to window 5) to see the color since the hex character must be within the display area.

You can use the block line command of CCN to remove the color from a block of source lines.

Highlighting Comment Lines

In-line comment lines may be highlighted or the highlighting can be removed within RPG, DDS, CL and CMD source member types.

HC – Highlight Comment Lines

Highlight in-line comment lines within a block by entering the line command of HC at the start of the block and again at the end of the block.

Highlighting is achieved by placing a hexadecimal 22 in the source line. When highlighting comments in RPG and DDS source code, this hex character will be placed in position 6 if the asterisk comment line does not have a specification type. Otherwise it will be placed in position 6. When highlighting CL and CMD comment lines, the hex character will be placed after the /* comment designator.

For RPGLE and SQLRPGLE, you may have to enter W5 (to window 5) to see the highlighting since the hex character must be within the display area.

* Tip: You can optionally use the HLTCMT command outside of SEU to highlight all in-line comments within a source member. Read about the HLTCMT command later in this document for more information.

RH – Remove Highlighting from Comment Lines

Remove highlighting from in-line comment lines within a block by entering the line command of RH at the start of the block and again at the end of the block.

The highlighting hexadecimal 22 character is removed from the in-line comments.

Case Conversion

Source lines can be converted to lower, mixed and upper case by entering one of the following SEU line commands:

LO -Lower Case Line

Converts a line to lower case.

To convert just a portion of a line to lower case, position to the desired starting point before pressing Enter.

Constants surrounded by quotes(') will be ignored.

Multiple LO line commands can be entered at a time.

* Shortcut: If you frequently use this line command, you can save keystrokes by assigning this command to function key F7 or F8. Read about SEUPLUS Function Keys later in this document.

LOB - Lower Case Block

Converts a block of lines to lower case. Key the LOB line command at the start of the block and key it again at the end of the block to convert, then press Enter.

To convert just a portion of each line in the block to lower case, position to the desired starting point before pressing Enter.

Constants surrounded by quotes(') will be ignored.

MX - Mixed Case Line

Converts a line to mixed case.

To convert just a portion of a line to mixed case, position to the desired starting point before pressing Enter.

Constants surrounded by quotes(') will be ignored.

Multiple MX line commands can be entered at a time.

* Shortcut: If you frequently use this line command, you can save keystrokes by assigning this command to function key F7 or F8. Read about SEUPLUS Function Keys later in this document.

MXB - Mixed Case Block

Converts a block of lines to mixed case. Key the MXB line command at the start of the block and key it again at the end of the block to convert, then press Enter.

To convert just a portion of each line in the block to mixed case, position to the desired starting point before pressing Enter.

Constants surrounded by quotes(') will be ignored.

UP - Upper Case Line

Converts a line to upper case.

To convert just a portion of a line to upper case, position to the desired starting point before pressing Enter.

Constants surrounded by quotes(') will be ignored.

Multiple UP line commands can be entered at a time.

* Shortcut: If you frequently use this line command, you can save keystrokes by assigning this command to function key F7 or F8. Read about SEUPLUS Function Keys later in this document.

UPB - Upper Case Block

Converts a block of lines to upper case. Key the UPB line command at the start of the block and key it again at the end of the block to convert, then press Enter.

To convert just a portion of each line in the block to upper case, position to the desired starting point before pressing Enter.

Constants surrounded by quotes(') will be ignored.

Modernizing RPG

RPG IV source can be modernized in existing fixed-format or converted to free-form RPG using the RPG Wizard (RPGWIZ). Read about the RPG Wizard later in this document for complete help on the conversion features.

Examples:

Before

```
 TEST1     IFEQ    'NEB'
           MOVE    REG      STS
 FACT1     ADD     FACT2    TOTL
           END
```

After (in fixed-format)

```
           IF      TEST1 = 'NEB'
           EVAL    STS = REG
           EVAL    TOTL = FACT1 + FACT2
           ENDIF
```

After (in free-format)

```
IF TEST1 = 'NEB';
   STS = REG;
   TOTL = FACT1 + FACT2;
ENDIF;
```

Enter one of the following SEU line commands to modernize your RPG IV source code with RPGWIZ:

Z - RPG Wizard on Line (no prompt)

Converts a line using the default RPGWIZ settings.

ZZ - RPG Wizard on Block (no prompt)

Converts a block of lines using the default RPGWIZ settings. Key the ZZ line command at the start of the block and key it again at the end of the block to convert, then press Enter.

ZP - RPG Wizard on Line (prompt)

Converts a line. You will first be prompted for the RPGWIZ settings.

ZZP - RPG Wizard on Block (prompt)

Converts a block of lines. You will first be prompted for the RPGWIZ settings. Key the ZZP line command at the start of the block and key it again at the end of the block to convert, then press Enter.

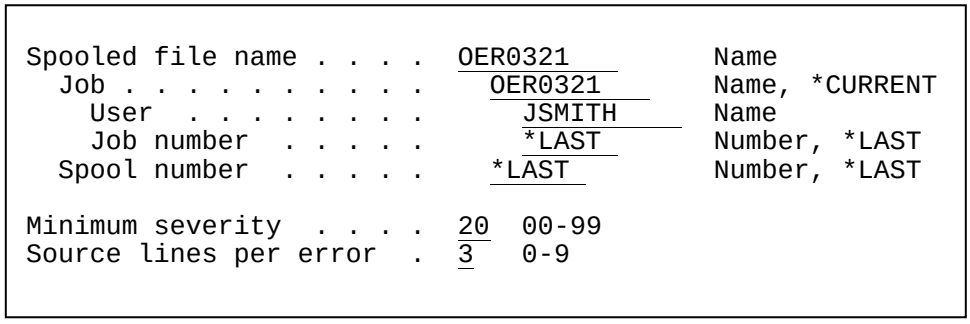
Examining Compile Information

Having to browse through compile listings for errors can often be a tedious process using traditional methods. Generally a programmer will just want to quickly see the errors that caused the compile to fail, while skipping over irrelevant compiler information and low severity errors.

SEUPLUS provides new line commands to allow you to quickly retrieve a compile listing and then display the status of the compile with any important compiler errors. As you correct the errors in your source code, you can then mark them off as "fixed" so you can just focus on the remaining items to correct. This SEUPLUS feature will work with RPG, DDS, CL and CMD compile listings.

RC - Retrieve Compile Information

The RC line command will load errors from a compiler listing into working storage. It will then automatically advance you to the "Work with Compile Information" screen (see the WC command on the next page) to show any errors found. When you run the RC command, you will be prompted with the following screen.



Following is the screen field descriptions:

Spooled file name
Specify the location of the compile spool listing by entering the spooled file name, job, user, job number and spool number. By default, a search will be conducted for the last batch job (in an OUTQ status) for the source member name and your user name. If you compiled the object interactively, then you will want to specify *CURRENT for the job name.

Minimum severity
Specify the minimum severity of error(s) to retrieve from the compile listing. For instance, specify the value of 20 if you are only interested in retrieving compile errors which are a severity of 20 or higher.

Source lines per error
For each error found, indicate the number of prior report lines to retrieve from the compile listing. You will probably want to display at least 2 to 3 source lines prior to the error message. When retrieving lines from the compiler report, the RC command should ignore irrelevant data (i.e. page headings).

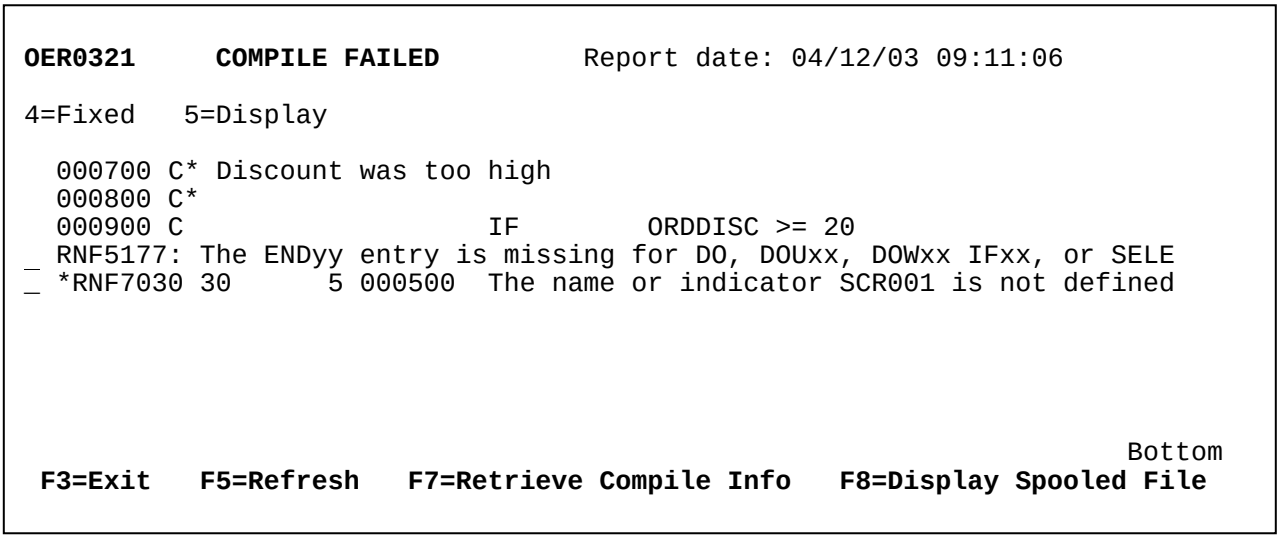
* Note: You only need to run the RC line command once for each compile. You can then use the WC line command (described on next page) to continue to work with the last retrieved compile information.

(continued on next page)

WC – Work with Compile Information

The WC line command allows you to work with Compile Information in a popup window. The top line of this display will show the spooled file name of the last retrieved compile, the object creation status and the date of the spooled file. If the Report date shows "N/A", then you need to retrieve the compile listing into working storage with the F7 function key (same as running the RC line command).

In the following screen example, two errors were found (5177 and 7030).



Following is the valid list options:

4=Fixed
When you fix the error in your source code, you can use this option 4 to mark the error as fixed. This option will change the text of the error message to * FIXED * and will automatically position to the next error in the list. If you no longer want to see the fixed error messages on the screen, you can press F5 to refresh the screen.

5=Display
Displays the compiler error message id, the message severity and the entire message text.

Following is the valid function keys:

F3=Exit
Exits the screen

F5=Refresh
Refreshes the screen from working storage. Only the unfixed errors will be displayed (those errors that you did not mark with option 4).

F7=Retrieve Compile Info
Retrieves a compile listing into working storage. Same as the RC (Retrieve Compile) line command.

F8=Display Spooled File
Displays the entire spooled file for the last retrieved compile listing.

This screen will display the actual source statement number on the left side of any source lines listed.

* Note: Once a compile listing has been retrieved into working storage, you can continue to work with that compile listing by running the WC line command as many times as necessary. If you recompile your source, then you will need to re-retrieve the compile information into working storage by pressing F7 from the WC screen or running the RC line command.

Examining Variables, Indicators, Key Lists and Copy Books

You can use SEUPLUS to view variable types and lengths, key lists, parameter lists and used indicators in RPGLE and SQLRPGLE source members. You can also view /COPY books quickly within most member types.

RV - Retrieve Variable Attributes

Retrieves the attributes of the variables used within a RPGLE or SQLRPGLE source member. Performs an interactive *NOGEN compile using the CRTRPGMOD command for the source member, which does not actually create an object. It then reads the compile listing to retrieve the variable attributes.

Before executing this command:

1. Make sure the source member has been saved recently since the compiler uses the saved source member.
2. Make sure your library list is set correctly so all externally described variable attributes can be located.

If any RNF7030 errors are encountered within the compile listing, the corresponding variable's attributes will not be available and you will be notified with a message.

After the retrieval, you can view individual variable attributes by using the line command **V V** or function key F7, which is describe below.

RVK - Retrieve Variables, KLISTs and PLISTs

Retrieves the Variable Attributes (same as RV line command) and additionally retrieves key lists (KLIST) and parameter lists (PLIST).

Before executing this command:

1. Make sure the source member has been saved recently, since the compiler uses the saved source member.
2. Make sure your library list is set correctly so all externally described variables, key lists and parameter lists can be located.

Please note that this command will take slightly more time to run than the RV command since more information must be retrieved.

To view a variable, key list or parameter list after retrieval, you can then use either the line command **V V** or function key F7, which is describe below.

V V - View Variable Attributes (or press F7)

When positioned on a variable, the variable's type, length and decimal positions will be displayed on the message line. When positioned on a key list or parameter list name, the fields that make up that list will be displayed in a popup window.

To view a variable, key list or parameter list, you can either:

1. Position the cursor to the variable (or list name) and press F7 or
2. Key in the line command VV, then position the cursor to the variable (or list name) and press Enter.

Make sure you first run the line command **RV** or **RVK** to retrieve the attributes for all the variables in the source member. Read the help text on the RV and RVK commands for more information.

VCB - View Copy Book Source

Displays the copy book source by executing the DSPPFM command for the member listed in the RPG IV /COPY or /INCLUDE statement.

If the copy book's source file name is not listed (i.e. /COPY MEMBER), the command will search for the member in your current source file/library.

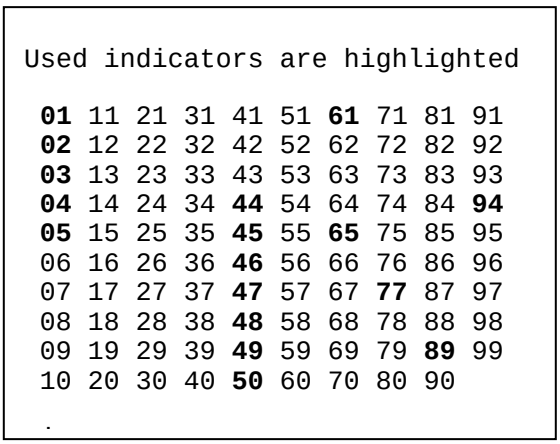
If the copy book's source file library is not listed (i.e. /COPY FILE, MEMBER), the command will search the library list for the source file.

V I - View Indicators

Displays all the numbered indicators within a popup window. Used indicators will be highlighted. This command can be run within either a RPGLE or SQLRPGLE member type.

Make sure you first run the line command **RV** or **RVK** to retrieve the attributes for all the variables in the source member. Read the help text on the RV and RVK commands for more information.

Example of viewing used indicators in a program with the VI line command:



Toolbox Memory

Toolbox memory allows you to save source code lines into memory, then view and/or insert them into a source member at a later time.

Toolbox memory is persistent within your job, similar to the Clipboard in Microsoft Windows, so it can be used within other SEU sessions in your job. For instance, you could copy some lines to memory in one source member, then open another source member and insert the memory into that source.

CM - Copy Line into Memory

Copies a line into the Toolbox memory. Multiple lines can optionally be copied into memory at one time by keying the CM command on all the lines to copy.

CCM - Copy Block into Memory

Copies a block of source lines into the Toolbox memory. Key the line command at the start of the block and key it again at the end of the block to copy, then press Enter.

DM - Delete Line into Memory

Deletes a line and moves it into the Toolbox memory. Only one DM line command can be entered at a time.

DDM - Delete Block into Memory

Deletes a block of source lines and moves them into the Toolbox memory. Key the line command at the start of the block and key it again at the end of the block to delete into memory, then press Enter.

AM - Insert Memory After current line

Inserts all the source lines stored in the Toolbox Memory into your current source member (after the current line).

BM - Insert Memory Before current line

Inserts all the source lines stored in the Toolbox Memory into your current source member (before the current line).

RM – Replace Line with Memory

Replace the current line with the source lines stored in the Toolbox Memory.

RRM – Replace Block with Memory

Replace a block of lines with the source lines stored in the Toolbox Memory.

TM - Toggles Memory

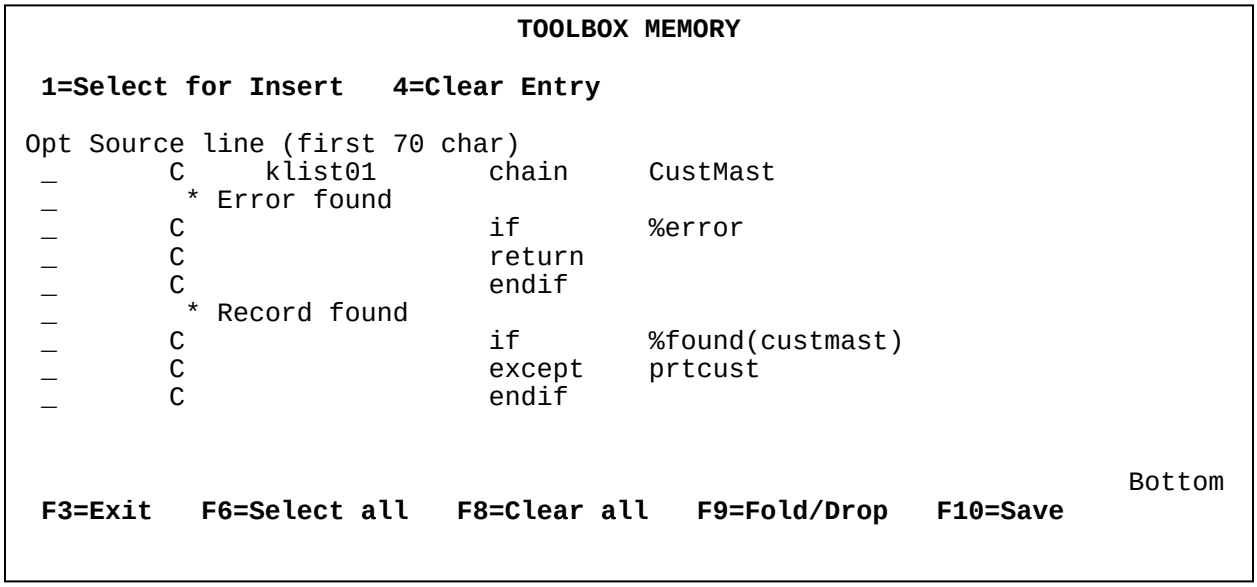
Toggles the memory mode between Append and Replace mode. By default, lines are appended to memory, which means new lines are added to the end of memory. In replace mode, memory is cleared before adding new lines.

CLRM - Clear Memory

Clears all entries in the Toolbox Memory.

WM - Work with Toolbox Memory

Allows you to work with the source lines stored in the Toolbox memory. A screen will appear showing the memory contents (demonstrated below). From this screen, you can select entries from memory to insert into your current source member.



```
                              TOOLBOX MEMORY

 1=Select for Insert   4=Clear Entry

Opt Source line (first 70 char)
 _       C     klist01       chain     CustMast
 _        * Error found
 _       C                   if        %error
 _       C                   return
 _       C                   endif
 _        * Record found
 _       C                   if        %found(custmast)
 _       C                   except    prtcust
 _       C                   endif


                                                                      Bottom
 F3=Exit   F6=Select all   F8=Clear all   F9=Fold/Drop   F10=Save
```

Options:

1=Select for Insert
Inserts one or more lines into the source member after the current line.

4=Clear Entry
Clears one or more entries from Toolbox memory.

Function keys:

F3=Exit
Exits the screen

F6=Select all
Inserts all Toolbox memory entries into the source member, after the current line.

F8=Clear all
Clears all Toolbox memory entries. Same as the CLRM line command.

F9=Fold/Drop
Shows the original sequence number and the last 30 positions of each memory entry.

F10=Save
Saves the contents of Toolbox memory to another source member you specify.

Nested Logic Tools

Working with RPG IV nested logic can be simplified by using the following line commands:

DI – Display Indented Source

Displays a partial source member listing for RPGLE and SQLRPGLE member types. Any nested logic will be shown in indented fashion.

Key the DI line command at the start of the block and key it again at the end of the block to display, then press Enter.

Both fixed-format and free-form RPG logic is supported. The beginning and ending of control structures (IFs, DOs, FORs, etc.) will be visually connected and the logic within them will be indented. The level of nesting will also be shown on the far right side of the report.

Example Output

```
SEQNBR*...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ..LEVL

  1.00     C                   dou       stoplp = 'DONE'                       B001
  2.00     C                   |         or xx >= 10                              001
  4.00     C                   |  if        ppcode = 'C'                       B002
  5.00     C                   |  |  eval      payment = 'CREDIT CARD'           002
  6.00     C                   |  |  eval      xx = xx + 1                       002
  7.00     C                   |  else                                          X002
  8.00     C                   |  |  eval      payment = 'CASH'                  002
  9.00     C                   |  endif                                         E002
 11.00     C                   enddo                                            E001
```

* Tip: You can optionally use the DSPIND command outside of SEU to display/print a nested listing for an entire source member. Read about DSPIND later in this document for more information.

By default, the nested logic will be indented 2 spaces, the pipe character will be used as a connector, in-line comments will be intersected with the connector, and right-hand comments will be suppressed from the report. If you wish to change these defaults, then you need to run IBM's CHGCMDDFT over the Toolbox's DSPIND command.

LI – List Indented Source

Performs the same indenting function as the DI line command, but prints the source member listing to your default output queue instead of displaying it onto your screen.

Key the LI line command at the start of the block and key it again at the end of the block to print, then press Enter.

(continued)

DN - Document Nested RPGIV Logic

Documents RPG IV nested logic by placing tags in positions 1-4 of your source code. Both fixed-format and free-form nested logic is documented.

Key the DN line command at the start of the block and key it again at the end of the block to document, then press Enter.

The beginning of a structure is denoted by a "B". An "else" operation is denoted by an "X". The end of a structure is denoted by an "E". The numeric portion of the tag indicates the level of the structure. For instance, a value of "B002" indicates the beginning of a structure at the second nested level.

Example:

```
B001    IF     B = 1
 001    EVAL   C = 2
B002    DOW    X < 3
 002    EVAL   X = X + 1
E002    ENDDO
X001    ELSE
 001    EVAL   D = 3
E001    ENDIF
```

You may have to window-left to view the nested tags.

* Tip: You can optionally use the DOCNST command outside of SEU to document the all the logic within a source member. Read about DOCNST later in this document for more information.

RN - Remove Nested Logic Documentation

Clears positions 1-4 of your source code. This is the area in which nested logic is documented with the DN line command.

Key the RN line command at the start of the block and key it again at the end of the block to process, then press Enter.

IN1 through IN9 - Indent Free-Form Nested Logic from 1 to 9 Spaces

Indents your RPG IV nested free-form logic the requested number of spaces.

Example of a request to indent 3 spaces using the IN3 line command:

```
Before            After
---------------   ---------------
if b = 1;         if b = 1;
c = 2;               c = 2;
dow x < 3;           dow x < 3;
x = x + 1;              x = x + 1;
enddo;               enddo;
endif;            endif;
```

Key the IN1-9 line command at the start of the block and key it again at the end of the block to indent, then press Enter.

Right-hand comments (in positions 81-100) will be preserved in their current location.

* Tip: You can optionally use the INDNST command outside of SEU to indent all the nested free-form logic within a source member. Read about INDNST later in this document for more information.

IN0 - Indent Free-Form Nested Logic 0 Spaces

Removes any leading blanks from your RPG IV free-form logic.

Example:

```
Before              After
---------------     ---------------
if b = 1;           if b = 1;
   c = 2;           c = 2;
   dow x < 3;       dow x < 3;
      x = x + 1;    x = x + 1;
   enddo;           enddo;
endif;              endif;
```

Key the IN0 line command at the start of the block and key it again at the end of the block to process, then press Enter.

Right-hand comments(in positions 81-100) will be preserved in their current location.

Executing IBM i commands

IBM i commands can be launched quickly from within SEU by entering simple line commands. Use one of the supplied line commands or create your own (see "Creating SEU Line Commands" later in this document).

If the IBM i command acts upon an object name, the SEU line command will attempt to locate the object name within the source line before running the IBM i command.

* Shortcut: If you frequently use one of the line commands listed below, you can save keystrokes by assigning the command to function key F7 or F8. Read about SEUPLUS Function Keys later in this document.

Listed below are the SEU line commands that execute IBM i commands:

ACT - Work with Active Jobs

Executes the WRKACTJOB command, which will prompt you for the subsystem name.

CPYSRC – Copy Source Member

Executes the CPYSRCF command, which allows you to copy the current source member to a new or existing member. This command will be prompt you for the destination source file, library and member.

DFD - Display File Description

Executes IBM's DSPFD command. If you are positioned on a RPG IV "F" specification or a calculation specification which contains a file I/O operation such as CHAIN or READ, the file listed will be used.

Optionally, you can position the cursor to a file name anywhere in the source before pressing Enter and that file will be used.

If a file name cannot be determined automatically, you will be prompted to enter a file name for the command.

Your library list(*LIBL) will be used as the library.,

LOG - Display Job Log

Displays the job log using the IBM i command DSPJOBLOG.

MSG - Display Messages

Displays your workstation's message queue using the IBM i command DSPMSG.

OPR - Display QSYSOPR Message Queue

Displays messages in QSYSOPR using the IBM i command DSPMSG QSYSOPR.

SBM - Work with Submitted Jobs

Allows you to work with your submitted jobs using the IBM command WRKSBMJOB.

SFE - Surveyor/400 File Editor

Launches the graphical File Editor within Linoma's Surveyor/400 product by executing the command SURVEYOR/SFE. This command only works within a Surveyor/400 emulator session.

If you are positioned on a RPG IV "F" specification or a calculation specification which contains a file I/O operation such as CHAIN or READ, the file listed will be used. Optionally, you can position the cursor to a file name anywhere in the source before pressing Enter and that file will be used. If a file name cannot be determined automatically, you will be prompted to enter a file name for the command.

Your library list (*LIBL) in Surveyor/400 will be used as the library.

SOP - Surveyor/400 Object Properties

Launches the graphical Object Properties within Linoma's Surveyor/400 product by executing the command SURVEYOR/SP. This command only works within a Surveyor/400 emulator session.

If you are positioned on a RPG IV "F" specification or a calculation specification which contains a file I/O operation such as CHAIN or READ, the file listed will be used. Optionally, you can position the cursor to an object name anywhere in the source before pressing Enter and that object will be used. If an object name cannot be determined automatically, you will be prompted to enter a object name for the command.

Your library list (*LIBL) in Surveyor/400 will be used as the library.

SPL - Work with Spool Files

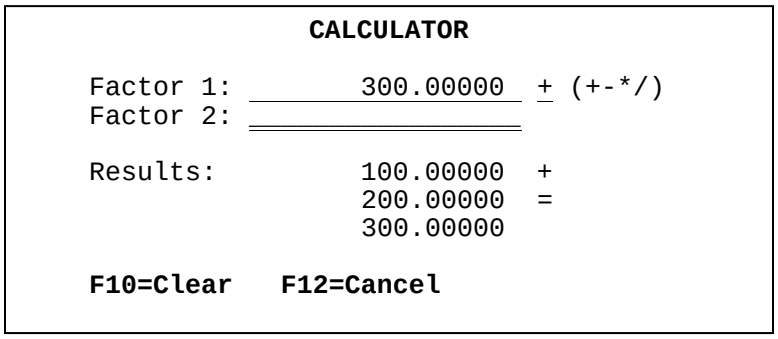
Allows you to work with your spool files using the IBM i command WRKSPLF.

Miscellaneous

Listed below are miscellaneous line commands within SEUPLUS:

CALC - On-Line Calculator

Displays a simple calculator that allows you to add, subtract, multiply and divide numbers.



Key in Factor 1 and the operator. Then key in Factor 2 and press the Field exit, Field+ or Field- key to perform the calculation. The result will then be displayed.

The calculator can also be used as an adding machine for adding and subtracting multiple numbers. The result will always be copied back into Factor 1 and the cursor will be positioned to Factor 2, so you can then add/subtract more numbers to/from the Result.

* Shortcut: If you frequently use this line command, you can save keystrokes by assigning this command to function key F7 or F8. Read about SEUPLUS Function Keys later in this document.

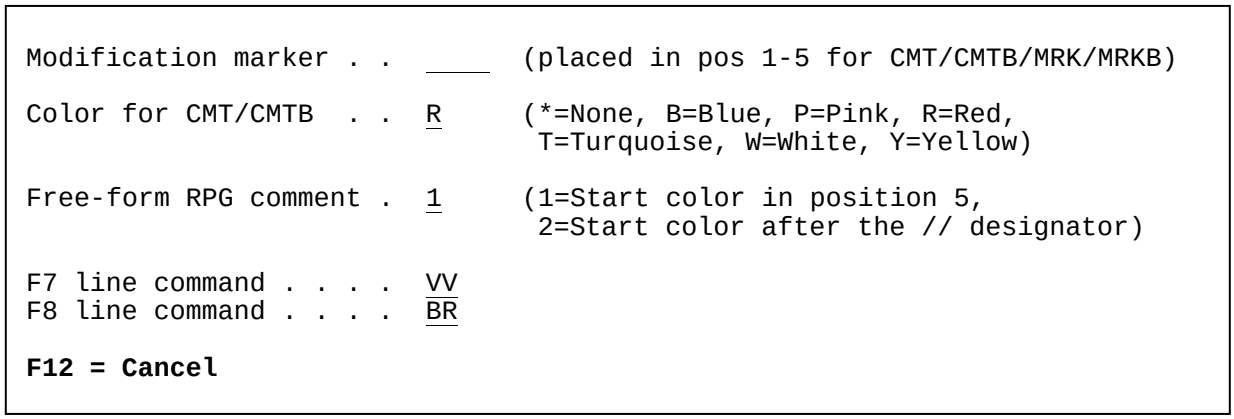
RESET - Resets Linoma's Toolbox

Resets Linoma's Toolbox by removing the SEU exit program from your job's memory. Toolbox memory will also be cleared.

This is necessary to release file locks when upgrading the Toolbox or to potentially correct any problems in the Toolbox.

USRDFT – Work with User Defaults

If you want to change the SEUPLUS defaults for your particular user id, execute the USRDFT line command. The USRDFT screen is shown below.



Following is the screen field descriptions:

Modification marker
The modification marker to be placed in positions 1-5 of the source line when executing the CMT, CMTB, MRK and MRKB line commands. The default is blanks, which indicates that no modification marker will be placed.

Color for CMT/CMTB
The color to use when commenting out source lines with the CMT and CMTB line commands. The default is R for Red.

Free-form RPG comment
When commenting out free-form RPG source lines (with CMT and CMTB) or coloring free-form RPG comment lines with the color line commands, this option indicates where to place the color hex attribute. By default, the hex attribute will be placed in position 5 of the line.

F7 line command
Specifies the SEUPLUS line command that will execute when the F7 function key is pressed. The default is VV (View Variable Attributes).

F8 line command
Specifies the SEUPLUS line command that will execute when the F8 function key is pressed. The default is BR (Break a Line into Two Lines).

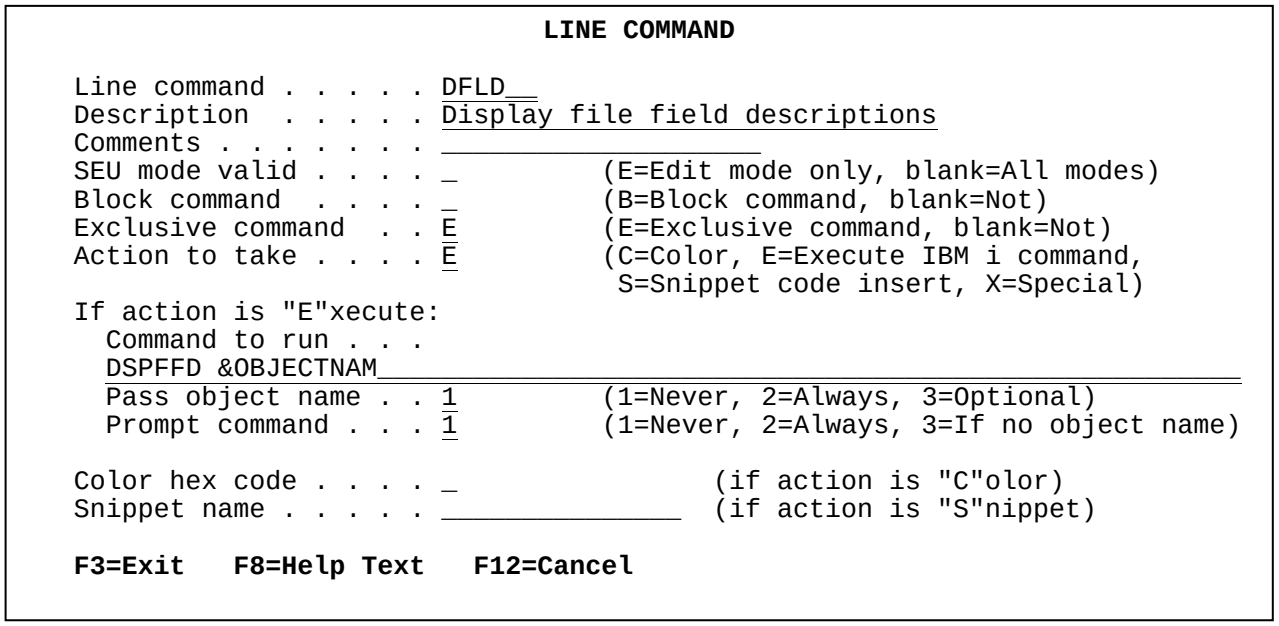
* Note: You can change the SEUPLUS global defaults by modifying the values stored under the user id of *DEFAULT in the database file of RPGTOOLBOX/BXP080.

Creating SEUPLUS Line Commands

You can create your own custom SEU line commands for executing IBM i commands or inserting source code snippets.

To create a line command:

1. Enter the line command of **?** or **HELP** from within SEU.
2. A list of the Toolbox's line commands will be displayed.
3. Press **F6** to create a command.
4. Enter the necessary values on the screen. Example of a line command to execute DSPFFD:



Following is the screen field descriptions:

Line command
The actual line command which can be entered on a SEU line. This command cannot be a duplicate of an existing IBM or SEUPLUS line command. This command also cannot begin with the letter "F" or "P", since those letters are reserved by IBM for formatting and prompting.

Description
A short description of the line command.

Comments
Additional comments about the line command.

SEU mode valid
Enter an **E** if the line command adds, changes or deletes source lines. Otherwise leave blank to indicate it's valid in all modes. For example, you would enter an **E** if the line command will insert a Snippet and normally enter a blank for executing IBM i commands.

Block command
Enter a **B** if the line command must be entered at the beginning of a block of source lines and again at the end of the block. This is only valid for Special (**X**) actions.

Exclusive command
Enter an **E** if the line command cannot be entered in conjunction with other line commands at the same time.

Action to take
C to Color a line (reserved for SEUPLUS)
E to Execute an IBM i command
S to Insert a Snippet
X is Special (reserved for SEUPLUS)

Command to run
If the action is an **E**, then enter the IBM i command to execute. The following special keywords may be entered within the command:

&OBJECTNAM
The special keyword of &OBJECTNAM (all uppercase) may be entered within the IBM i command to indicate that the keyword will be replaced with an object name found in the source. For instance, the IBM i command DSPFFD &OBJECTNAM will display the file's fields for the object name found within the source line.

If &OBJECTNAM is specified and the user enters the line command on an F spec or a calculation spec with a file I/O operation, &OBJECTNAM will be automatically filled in from the file name found in the source line. Otherwise the user can position the cursor to the object name before pressing enter to have that name fill the &OBJECTNAM keyword.

&SRCMBR, &SRCFIL and &SRCLIB
The special keywords of &SRCMBR, &SRCFIL and &SRCLIB (all uppercase) may be entered within the IBM i command to indicate that the keywords will be replaced with the current source member name, source file name and source library name. See the SEU PLUS line command of CPYSRC for an example of how these keywords are implemented.

Pass object name
If the action is an **E**, enter a
1 if the &OBJECTNAM is never passed to the IBM i command.
2 if &OBJECTNAM is required to be filled in.
3 if &OBJECTNAM is optional.

Prompt command
If the action is an **E**, enter a
1 to indicate the IBM i command should never be prompted.
2 to indicate the IBM i command should always be prompted first.
3 to indicate the IBM i command should be prompted only if the &OBJECTNAM keyword value could not be determined.

Color hex code
If the action is **C**, enter the color hex code to insert into the source line. All the possible color line commands should already be available from SEUPLUS, so you should not need to specify this.

Snippet name
If the action is **S**, enter the name of the source code snippet to insert.

5. Press **F8** to enter detailed help text for the line command

6. After entering all the needed information, press **Enter** to create the line command.

7. The new line command can now be used from within SEU.

Changing a SEUPLUS Line Command

You can change user-created SEU line commands within SEUPLUS.

To change a line command:

1. Enter the line command of **?** or **HELP** from within SEU to bring up the list of the line commands.
2. Find the line command you wish to change, then enter option **2** next to it.
3. Make the necessary changes or press **F8** to change it's corresponding help text.
4. After making the changes, press **Enter** to update the command.
5. The changes in the line command will be recognized immediately.

* Line commands supplied by Linoma Software cannot be modified.

Securing SEUPLUS Line Commands

By default all SEU users can create, change and delete custom line commands. However you can restrict certain users from having maintenance capability by changing the object authority on a data area called @SEUPLUS in the RPGTOOLBOX library.

If a user has at least *CHANGE authority to the @SEUPLUS data area, then that user will be able to maintain custom line commands. Otherwise the user will only be able to use those commands.

Use the following IBM command to change authority on this data area:
EDTOBJAUT OBJ(RPGTOOLBOX/@SEUPLUS) OBJTYPE(*DTAARA)

Active SEUPLUS users will not immediately realize changes to the authority. They must first enter the line command RESET or sign off to reset their session.

* Line commands supplied by Linoma Software cannot be modified.

Function Keys

The SEU function keys of F7 and F8 can be used as shortcuts for executing SEUPLUS line commands. This is especially helpful when a line command requires the cursor be positioned somewhere within the source line for execution.

For instance, when breaking a source line into two lines, you must first position the cursor to the place where you want the break to occur. Using a function key, it is simpler to position the cursor to the point of the break and just press F8 (versus having to key in the line command of BR, position the cursor to the point of break and then press Enter).

Reassigning Function Keys

By default, F7 is assigned to line command V V (View Variable) and F8 is assigned to line command BR (Break a line into two lines). You can easily reassign these function keys to other SEUPLUS line commands for your user id by running the SEUPLUS line command of USRDFT (Work with User Defaults).

Snippets of Source Code

Introduction

Snippets of predefined source code can quickly be found and used from within SEU. By using pre-tested and proven source code, the development cycle can be greatly shortened with less coding and testing time. Using Snippets can also help ensure programming standards are being followed.

Over 200 pre-defined snippets are included with the Toolbox, including source snippets for:

- All RPG IV built-in functions (BIF) available up to V5R4.
- All RPG IV free-format operations.
- The program data structure (SDS) and the file information data structure (INFDS).
- Definition (D) specifications for creating standalone fields, constants, data structures, arrays, procedure interfaces and prototypes.
- File I/O operations using either result indicators or BIFs (%found, %error, %eof, etc.).
- Various embedded SQL statements for RPG IV.
- A template for creating a RPG IV sub-procedure.
- Templates for creating new CL programs, Physical files and Logical files.
- Standard header comments for starting out new RPG, DDS, CL and CMD source members.
- Miscellaneous snippets (insert /Free block, comment block, etc.)

Intelligent Prompting

Most of the included snippets will intelligently prompt for values to fill into the snippet. For instance, the RPG CHAIN operation Snippet will prompt for the key list name, file name and other related values. The Toolbox will then merge those values with the snippet before inserting into your current source member.

Create your own Snippets

You can also easily create your own custom snippets of source code using the Toolbox-supplied Snippets as template examples.

Using Snippets

A snippet of source code can be inserted into a SEU source member using one of the five techniques:

IX line command - Inserts line(s) of snippet source below the current source statement.

Enter the SEU line command of **IX** on the source line to insert the snippet after. You will then be prompted for the snippet name. If you do not know the name of the snippet, press F4 to perform a search/select.

IXC line command - Inserts a snippet into the current source statement.

Key the SEU line command of **IXC** on the source line, then optionally position the cursor to the point of insert and press Enter. You will then be prompted for the snippet. If you do not know the name of the snippet, press F4 to perform a search/select.

Only the first line of the snippet will be inserted and it must not exceed the available space.

F7 from within Expression Editor - Inserts a snippet into the current expression.

While editing an expression or operation using one of the line commands of **IE**, **IEX**, **E** or **EE**, position the cursor to the point of insertion and press function key **F7**. You will then be prompted for the snippet name. If you do not know the name of the snippet, press F4 to perform a search/select.

Only the first line of the snippet will be inserted and it must not exceed the available space.

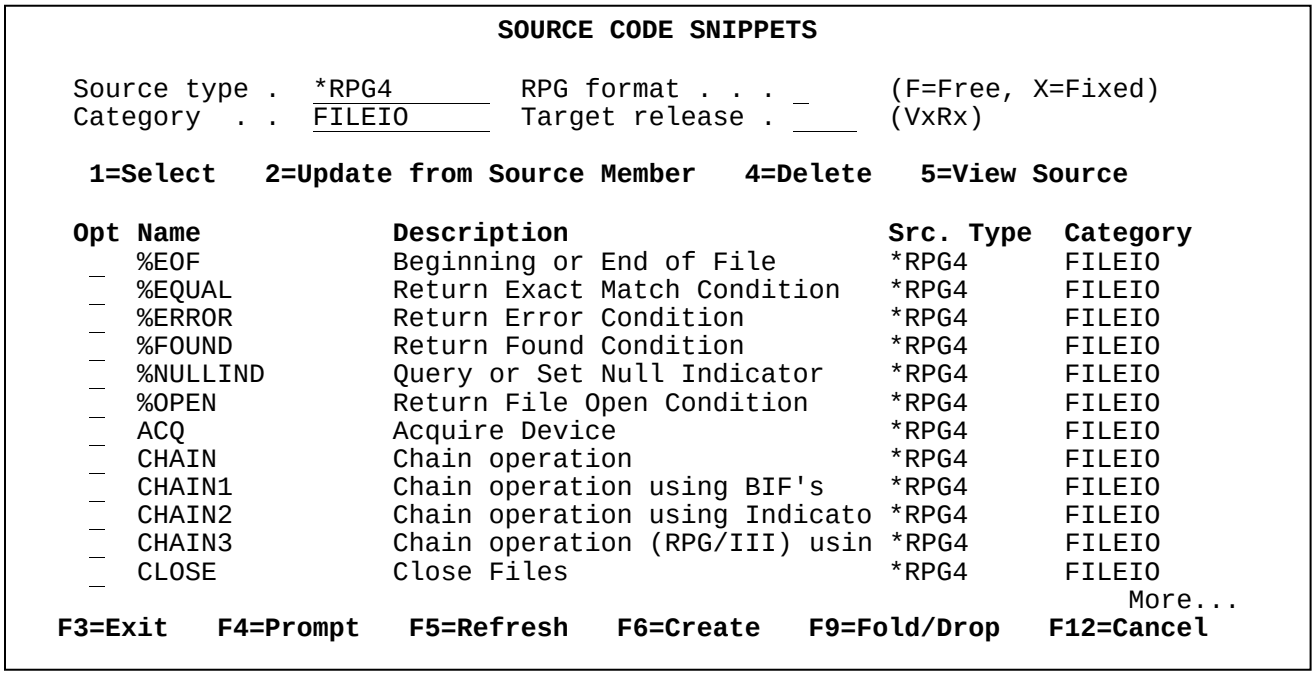
Shortcut SEU line command - Inserts line(s) of snippet source below the current source statement

Toolbox SEU line commands can be created for inserting frequently used snippets (without having to specify the snippet name each time). For instance, the Toolbox supplied **I*** line command will insert the snippet called COMMENT1, which inserts three comment lines into your source code.

Read the section labeled "Creating SEU line commands" earlier in this document on how you can create your own shortcut commands for inserting frequently used snippets.

Finding Snippets

To list all of the snippets available in the Toolbox, enter the SEU line command of **IX** or **IXC** and then press F4. The following screen will be displayed:



```
                          SOURCE CODE SNIPPETS

  Source type .  *RPG4        RPG format . . .  _      (F=Free, X=Fixed)
  Category  . .  FILEIO       Target release .  ____   (VxRx)

   1=Select   2=Update from Source Member   4=Delete   5=View Source

  Opt Name            Description                     Src. Type  Category
    _  %EOF           Beginning or End of File        *RPG4      FILEIO
    _  %EQUAL         Return Exact Match Condition    *RPG4      FILEIO
    _  %ERROR         Return Error Condition          *RPG4      FILEIO
    _  %FOUND         Return Found Condition          *RPG4      FILEIO
    _  %NULLIND       Query or Set Null Indicator     *RPG4      FILEIO
    _  %OPEN          Return File Open Condition      *RPG4      FILEIO
    _  ACQ            Acquire Device                  *RPG4      FILEIO
    _  CHAIN          Chain operation                 *RPG4      FILEIO
    _  CHAIN1         Chain operation using BIF's     *RPG4      FILEIO
    _  CHAIN2         Chain operation using Indicato  *RPG4      FILEIO
    _  CHAIN3         Chain operation (RPG/III) usin  *RPG4      FILEIO
    _  CLOSE          Close Files                     *RPG4      FILEIO
                                                                    More...
  F3=Exit   F4=Prompt   F5=Refresh   F6=Create   F9=Fold/Drop   F12=Cancel
```

By default, just the source snippets for your current source member type will initially be listed. In the example above, the user wanted to filter for only the RPG IV file I/O snippets.

Filtering Snippets

To help you find a snippet quickly, use the top four fields on the screen to filter what snippets to list. These filtering fields can be used separately or in conjunction with each other.

Source type filter

The source type in which the snippets are associated. Leave this field blank to ignore the source type OR enter the source type to filter OR press F4 to on the source type to select one. The valid values are:

Value	Description
*CL	CL Programs (CLP, CLLE)
*RPG3	RPG/400 and RPG III programs (RPG, RPT, RPG38, RPT38, SQLRPG)
*RPG4	RPG IV programs (RPGLE, SQLRPGLE)
CMD	Command
DSPF	Display File
LF	Logical File
PF	Physical File

Notice that *CL, *RPG3 and *RPG4 include multiple source types since the source syntax is similar. For instance, enter *RPG4 to just find the snippets that can be inserted into RPGLE or SQLRPGLE source types.

Category filter
The category in which the snippets are associated. Leave this field blank to ignore the category OR enter the category to filter OR press F4 on the category to select one. The valid values are:

Value	Description
ARITHMETIC	Arithmetic Operations
BRANCH	Branch to SR or Program
COMMENTS	General Comments
CONTROL	Controlling Operations
DATE/TIME	Date/Time Operations
DECLARE	Declare Statements (free form)
DS	Data structures
FILEIO	RPG File IO operations
MEMORY	Memory operations
SQL	Embedded SQL statements
STRING	String Manipulation

For instance, enter FILEIO to just find the snippets that are related to File I/O operations (chains, reads, etc.).

RPG format filter
The format in which RPG-related snippets are developed in. Leave this field blank to ignore the format OR enter the format to filter. The valid values are:

Value	Description
F	Free form RPG operations and BIFs
X	Fixed format RPG operations

For instance, enter X to just find the snippets that are fixed format.

Target release filter
The minimum IBM i Release the snippet was created for. Enter a value in the format of VxRy OR leave blank to ignore the release. For instance, enter V5R1 to just find the snippets that work on that IBM i release and greater.

Function keys

You can press **F9** to show additional information about each snippet listed.

Options

Once you locate the snippet you're interested in, you can enter one of the following options next to it:

1=Select
Selects the snippet to insert into your current source member.

2=Update from Source Member
Described later in this document.

4=Delete
Described later in this document.

5=View Source
View the actual source associated with the snippet.

Snippet Example

The following is an example of the Toolbox supplied CHAIN1 snippet, which contains the RPG Chain operation and the optional %error, %found and not(%found) BIFs.

When the user requests to insert the CHAIN1 snippet, the following screen will prompt for the snippet values:

```
                          Enter Snippet Values

 Snippet: CHAIN1 - Chain operation using BIF's

                                 Value                Comments
 File name  . . . . . . . . . . custmast_____
 Field, key list or rrn . . . . klist01______
 Data structure name  . . . . . ______________
 Trap Errors or No Lock . . . . (e)               (e), (n), or (en)
 Include %error snippet . . . . Y                 Y=Yes, blank=No
 Include %found snippet . . . . Y                 Y=Yes, blank=No
 Include not(%found) snippet  . _                 Y=Yes, blank=No
```

After entering the snippet values, the following source code is inserted into the current SEU source member:

```
 * Retrieve a record
C     klist01       chain(e)  custmast
 * Error found
C                   if        %error
C                   endif
 * Record found
C                   if        %found(custmast)
C                   endif
```

Notice the klist name and file name from the prompt screen was filled into the source. At the user's request, the %error and %found BIFs were also inserted.

Snippet Source Described

The following snippet source code demonstrates the CHAIN1 snippet used in the example on the previous page. If you wish to view this source on-line, display the member CHAIN1 in the source file of RPGTOOLBOX/SNIPPETS.

```
 */SNIP_COMMENTS_BEGIN
 ********************************************************************************************
 * Snippet name......... CHAIN1
 * Description.......... Chain operation using BIF's
 * Source type.......... *RPG4
 * FREE or FIXED format. FIXED
 * Category............. FILEIO
 * Min. IBM i release... V4R4
 * Authored by.......... Linoma Software
 * Date created......... June, 2001
 ********************************************************************************************
 * KEYWORD    LABEL                          LEN REQ COMMENTS             DEFAULT    CAS
 * ---------- ------------------------------ --- --- -------------------- ---------- ---
 */SNIP_COMMENTS_END
 */SNIP_PROMPTS_BEGIN
 * @file      File name  . . . . . . . . . . 014  Y                                   L
 * @key       Field, key list or rrn . . . . 014  Y                                   L
 * @ds        Data structure name  . . . . . 014  N                                   L
 * @en        Trap Errors or No Lock . . . . 004  N  (e), (n), or (en)    (e)        L
 * @ierror    Include %error snippet . . . . 001  N  Y=Yes, blank=No      Y          U
 * @ifound    Include %found snippet . . . . 001  N  Y=Yes, blank=No      Y          U
 * @inotfound Include not(%found) snippet  . 001  N  Y=Yes, blank=No                 U
 */SNIP_PROMPTS_END
 */SNIP_FIXED_FORMAT
 * Retrieve a record
C     @key          chain@en  @file         @ds
 */SNIP_IF @ierror
 * Error found
C                   if        %error
C                   endif
 */SNIP_ENDIF
 */SNIP_FREE_FORMAT
 */SNIP_IF @ifound
 * Record found
C                   if        %found(@file)
C                   endif
 */SNIP_ENDIF
 */SNIP_IF @inotfound
 * Record not found
C                   if        not%found(@file)
C                   endif
 */SNIP_ENDIF
```

Snippet Directives

The highlighted source lines in the CHAIN1 snippet on the previous page are called **Snippet Directives**, which are used to control the Snippet's behavior.

Snippet Directives must:

- Start in position 8 of the source line.
- Begin with a forward slash.
- Be all capitalized

Listed below are the valid snippet directives.

/SNIP_COMMENTS_BEGIN
Marks the beginning of snippet comments, which can be inserted anywhere within the snippet for placing technical documentation. Snippet comments are bypassed during a Snippet Insert request.

/SNIP_COMMENTS_END
Marks the end of comments.

/SNIP_PROMPTS_BEGIN
Marks the beginning of prompts. Read about snippet prompts in the following pages.

/SNIP_PROMPTS_END
Marks the end of prompts.

/SNIP_FIXED_FORMAT
The source lines following this directive are in fixed format. The Toolbox will replace each keyword in a fixed format line with the keyword value for the maximum length of the value without shifting the remaining source to make room.

/SNIP_FREE_FORMAT
The source lines following this directive are in free format. The Toolbox will make room in a free format source line to replace the keyword with the keyword value.

/SNIP_IF keyword
If the keyword contains a value, then insert the source code following this directive until another controlling directive is encountered. There must be one space separating the directive and the keyword.

/SNIP_IFNOT keyword
If the keyword does not contain a value, then insert the source code following this directive until another controlling directive if encountered. There must be one space separating the directive and the keyword.

/SNIP_ELSE
If a prior /SNIP_IF or /SNIP_IFNOT condition was not met, then insert the source code following this directive.

/SNIP_ENDIF
Marks the end of a /SNIP_IF or /SNIP_IFNOT.

/SNIP_DONE
Stop inserting any additional snippet source code.

Snippet Header Section

The first section of a snippet source member contains header information, which describes the snippet. The header section is used by the Toolbox to automatically fill the snippet index database, so this information must always be in the same format and positions.

CHAIN1 header information:

```
*/SNIP_COMMENTS_BEGIN
*********************************************************************************************
* Snippet name......... CHAIN1
* Description.......... Chain operation using BIF's
* Source type.......... *RPG4
* FREE or FIXED format. FIXED
* Category............. FILEIO
* Min. IBM i release... V4R4
* Authored by.......... Linoma Software
* Date created......... June, 2001
*********************************************************************************************
```

Header information must be contained within a snippet comment section (notice the /SNIP_COMMENTS_BEGIN directive above).

The label headings for the header information must start in column 9 and the corresponding values must start in column 31 of the source.

The header information consists of the following fields:

Snippet name
The name of the snippet, up to 15 characters in length. The snippet name must be in all uppercase and cannot duplicate another snippet name in the snippet index database. The snippet name does not have to match the source member name.

Description
The description of the snippet, up to 30 characters in length.

Source type
The source type in which the snippet can be inserted into. Valid values are:

Value	Description
*CL	CL Programs (CLP, CLLE)
*RPG3	RPG/400 and RPG III programs (RPG, RPT, RPG38, RPT38, SQLRPG)
*RPG4	RPG IV programs (RPGLE, SQLRPGLE)
CMD	Command
DSPF	Display File
LF	Logical File
PF	Physical File

You can also create your own snippet source types, which is described later in this document.

FREE or FIXED format
The general format of the snippet source. Valid values are:

Value	Description
FREE	Fixed format (logic is within bound by columns)
FIXED	Free format (logic is not bound by columns)

Category
The category in which the snippet is grouped into. Valid values are:

Value	Description
ARITHMETIC	Arithmetic Operations
BRANCH	Branch to SR or Program
COMMENTS	General Comments
CONTROL	Controlling Operations
DATE/TIME	Date/Time Operations
DS	Data structures
FILEIO	RPG File IO operations
MEMORY	Memory operations
STRING	String Manipulation

You can also create your own categories, which is described later in this document.

Min. IBM i release
The minimum IBM i release the snippet was created for. Enter a value in the format of VxRy. For instance, enter V5R1 to indicate the snippet can be used on release V5R1 and greater.

Authored by
The snippet author's name. All Toolbox supplied snippets have the author of "Linoma Software". If you create your own snippet, you should enter your name here. The author name is not loaded in the snippet index database, so the length and format of this field is at your discretion.

Date created
The date the snippet source was created. The date is not loaded in the snippet index database, so the length and format of this field is at your discretion.

Snippet Prompts

Snippet prompts are optional. If specified, the Toolbox will ask the user to enter values for filling out the snippet source or to determine what source lines to insert. The Prompts section within a snippet source member controls what is displayed on the screen.

CHAIN1 prompts:

```
* KEYWORD    LABEL                          LEN REQ COMMENTS             DEFAULT    CAS
* ---------- ------------------------------ --- --- -------------------- ---------- ---
*/SNIP_COMMENTS_END
*/SNIP_PROMPTS_BEGIN
* @file      File name  . . . . . . . . . . 014  Y                                   L
* @key       Field, key list or rrn . . . . 014  Y                                   L
* @ds        Data structure name  . . . . . 014  N                                   L
* @en        Trap Errors or No Lock . . . . 004  N  (e), (n), or (en)    (e)        L
* @ierror    Include %error snippet . . . . 001  N  Y=Yes, blank=No      Y          U
* @ifound    Include %found snippet . . . . 001  N  Y=Yes, blank=No      Y          U
* @inotfound Include not(%found) snippet  . 001  N  Y=Yes, blank=No                 U
*/SNIP_PROMPTS_END
```

Snippet prompts must begin with the /SNIP_PROMPTS_BEGIN directive and end with the /SNIP_PROMPTS_END directive.

The prompt elements are arranged into fixed columns, as described below:

KEYWORD (columns 9 through 18)
Each prompt field must have an associated keyword assigned to it. When a value is entered for a prompt, it is assigned to the keyword. Keywords must be unique and should be distinguishable from the actual source that's inserted. For instance, in the CHAIN1 snippet, all the keywords begin with the @ symbol so they don't conflict with the normal source. Keywords are case sensitive.

LABEL (columns 20 through 49)
The label which will appear on the prompt screen. For better screen appearance, it is recommended to use dots to fill in the remaining space after the label.

LEN (columns 51 through 53)
The length of the prompt value. The value's length cannot exceed 40 characters, so the valid range for the length is 001 to 040. The length must contain leading zeros.

REQ (column 56)
Whether or not the prompt value is required to be filled on the prompt screen. The value of "Y" indicates the prompt value is required, otherwise a "N" indicates the prompt value is not required.

COMMENTS (columns 59 through 78)
Any comments to show the user for the prompt value on the screen. This is a good place to list valid values. The comments will not be shown on the screen if the length (LEN) for the prompt value is greater than 20 (due to limited screen space).

DEFAULT (columns 80 through 89)
The default value for the prompt, which will be pre-filled on the prompt screen.

You can optionally enter special values in this field to automatically fill in the default value. Specify the special value *DATE for the current date, *TIME for the current time, *USER for the current user, *MBRNAME for the current member name or *MBRTEXT for the current member text description. Look at the STARTRPG4 snippet for an example of how these special values are used.

CAS (column 92)
The case conversion to perform for the value entered on the prompt screen. The value of "U" will convert the prompt value entered into upper case. The value of "L" will convert the prompt value entered into lower case. A blank value in the CAS field will leave the prompt value as-is.

Snippet Source to Insert

The actual source to insert is listed at the bottom of the snippet. This snippet source can optionally contain conditional blocks of source lines and replacement values.

CHAIN1 source to insert:

```
 */SNIP_FIXED_FORMAT
 * Retrieve a record
C     @key          chain@en  @file         @ds
 */SNIP_IF @ierror
 * Error found
C                   if        %error
C                   endif
 */SNIP_ENDIF
 */SNIP_FREE_FORMAT
 */SNIP_IF @ifound
 * Record found
C                   if        %found(@file)
C                   endif
 */SNIP_ENDIF
 */SNIP_IF @inotfound
 * Record not found
C                   if        not%found(@file)
C                   endif
 */SNIP_ENDIF
```

The /SNIP_FIXED_FORMAT directive proceeds the chain operation since the source line positions are in fixed format. Then the /SNIP_FREE_FORMAT directive is listed above the %found BIF to direct the Toolbox to make room for the keyword value.

Notice the controlling directives. For example, if the @ierror keyword contains a value then the source below the /SNIP_IF @ierror directive will be inserted up to the following /SNIP_ENDIF directive.

Snippet Advanced Logic

IF/ELSE Controlling Logic

/SNIP_IF and /SNIP_IFNOT directives can be nested within each other.

```
 */SNIP_IF @test1
C                   eval      @test1 = *blank
 */SNIP_ELSE
 */SNIP_IF @test2
C                   if        @test2 = *blank
 */SNIP_ELSE
 */SNIP_DONE
 */SNIP_ENDIF
 */SNIP_ENDIF
```

In the above example:

- If @test1 contains a value, then insert the line below /SNIP_IF @test1.
- Otherwise if @test2 contains a value, then insert the line below /SNIP_IF @test2.
- If neither @test1 or @test2 contain values, then the /SNIP_DONE directive is encountered and no more source is inserted.

Conditioned Prompts

Prompts can be conditioned based on prior prompt values.

```
 */SNIP_PROMPTS_BEGIN
 * @test1     Enter test1 value  . . . . . 014  Y                                L
 */SNIP_PROMPTS_END

 */SNIP_IF @test1
 */SNIP_PROMPTS_BEGIN
 * @test2     Enter test2 value  . . . . . 014  Y                                L
 */SNIP_PROMPTS_END
 */SNIP_ELSE

 */SNIP_PROMPTS_BEGIN
 * @test3     Enter test3 value  . . . . . 014  Y                                L
 */SNIP_PROMPTS_END
 */SNIP_ENDIF
```

In the above example:

- If @test1 contains a value, then @test2 will be prompted
- Otherwise @test3 will be prompted.

Creating a Snippet

Snippet Components

A snippet consists of:

- The snippet source code, which can be stored in a regular source member AND
- An entry in the snippet index database, which is needed for efficient searches and retrieval.

Creating the Snippet Source Member

To create your own snippet source:

1. First read the prior section "Snippet Source Described" to understand all the components of a Snippet source member.
2. You should place the snippet source member in a source file with a record length of 112.
3. Based on the type of snippet you wish to create, it is recommended to copy one of the existing Toolbox snippets(found in RPGTOOLBOX/SNIPPETS source file) as a starting template.

All the Toolbox-supplied snippets are stored in the source file of RPGTOOLBOX/SNIPPETS. It is strongly recommended you do not create new source members in this source file since it will be replaced during a Toolbox upgrade.

Creating the Snippet Database Index Entry

After creating the snippet source member in your source file, you need to create an entry in the snippet index database:

1. Enter the SEU line command of **IX**.
2. Press **F4** on the prompt screen.
3. When the Snippet list is displayed, press **F6** to create a new snippet index entry.
4. You will then be prompted to enter the member name, source file name and library name of your snippet source.
5. Key in your source member information and then press Enter.
6. The Toolbox will read the header information in the specified source member to create a snippet index entry.
7. The snippet is now available to use.

* User-created snippet index entries will be preserved during a Toolbox upgrade.

Changing a Toolbox-supplied Snippet

You may wish to change certain Toolbox-supplied snippets to meet your programming standards or to add additional functionality. For instance, the RPG IV snippet source is currently in all lower-case. However if your standards are for all upper case, then you need to make the appropriate changes.

All the toolbox-supplied snippets are stored in the source file of RPGTOOLBOX/SNIPPETS. It is strongly recommended you do not change any snippet source directly within this source file since this file will be replaced during a Toolbox upgrade.

To change a Toolbox-supplied Snippet, it is recommended that you copy the snippet source member from the RPGTOOLBOX/SNIPPETS source file into your own designated source file. Then make the change to the copied source member within your own source file.

After copying and modifying the snippet source member in your source file, you need to redirect the snippet index database to point to your source member:

1. Enter the SEU line command of **IX**.
2. Press **F4** on the next screen.
3. Find the snippet you wish to redirect.
4. Key option **2** next to the snippet and press Enter.
5. You will then be prompted to enter the new member name, source file name and library name.
6. After keying in your source member information, press Enter.
7. The Toolbox will update the source member location, and will also read the header information in the source member to update the snippet index database with the header values.

Changes to the Snippet index database will be preserved during a Toolbox upgrade.

Changing a User-Created Snippet

Snippet source is retrieved at the time of insert, so changes you make to your snippet source member will be recognized immediately (excluding the header information). So in most cases, you can just make the changes to the snippet source member and you're done.

The only time you need to update the snippet index database is when either:

- The snippet source member was renamed or moved to a different source file OR
- The header information at the top of the snippet source member was changed. For instance, the snippet category was changed to a different value.

To update the snippet index database:

1. Enter the SEU line command of **IX**.
2. Press **F4** on the next screen.
3. Find the snippet you wish to change.
4. Key option **2** next to the snippet and press Enter.
5. You will then be prompted to enter the member name, source file name and library name.
6. After keying in your source member information, press Enter to update the Snippet index database.
7. The Toolbox will read the header information in the source member and update the snippet index database.

Changes to the Snippet index database will be preserved during a Toolbox upgrade.

Deleting a Snippet

To delete a snippet, you need to remove the entry from the snippet index database and optionally delete the related source member.

* Please note that Toolbox-supplied snippets cannot be deleted.

To delete the snippet from the index database:

1. Enter the SEU line command of **IX**.
2. Press **F4** on the next screen.
3. Find the snippet you wish to delete.
4. Key option **4** next to the snippet and press Enter.
5. Just the entry from the snippet index database will be removed. You can then optionally delete the related source member through PDM.

Securing Snippets

By default all SEU users can create, change and delete entries in the snippet index database. However you can restrict certain users from having maintenance capability by changing the object authority on a data area called @SNIPPETS in the RPGTOOLBOX library.

If a user has at least *CHANGE authority to the @SNIPPETS data area, then that user will be able to maintain the snippet index database. Otherwise the user will only be able to use snippets.

Use the following IBM command to change authority on this data area:
EDTOBJAUT OBJ(RPGTOOLBOX/@SNIPPETS) OBJTYPE(*DTAARA)

Active SEUPLUS users will not immediately realize changes to the authority. They must first enter the line command RESET or sign off to reset their session.

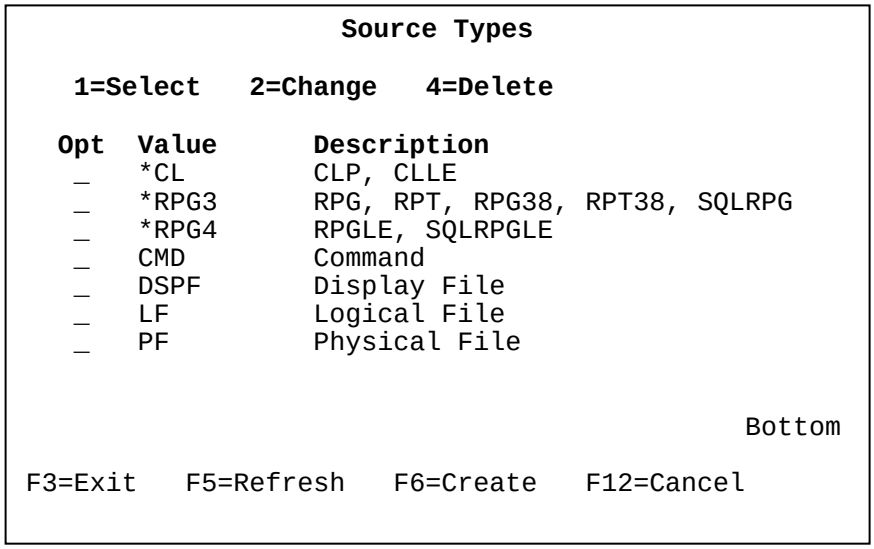
Snippet Source Types (Maintaining)

Snippet Source Types are used to associate a snippet with a type of source member, which allows the user to quickly find the correct snippet to insert. Most of the common Source Types have already been supplied.

* Please note that vendor-supplied Source Types cannot be changed or deleted.

To create, change or delete Snippet Source Types:

1. Enter the SEU line command of **IX**.
2. Press **F4** to display the list of snippets.
3. When the list of snippets is display, position to the "Source type" field on the top of the screen and press **F4**.
4. You will be presented with the following screen:



```
                    Source Types

   1=Select   2=Change   4=Delete

  Opt  Value      Description
   _   *CL        CLP, CLLE
   _   *RPG3      RPG, RPT, RPG38, RPT38, SQLRPG
   _   *RPG4      RPGLE, SQLRPGLE
   _   CMD        Command
   _   DSPF       Display File
   _   LF         Logical File
   _   PF         Physical File


                                              Bottom

F3=Exit   F5=Refresh   F6=Create   F12=Cancel
```

5. Press **F6** to create your own Source Type (you can then use that Source Type when defining new snippets) OR
6. Enter option **2** to change a Source Type description OR
7. Enter option **4** to delete a Source Type.

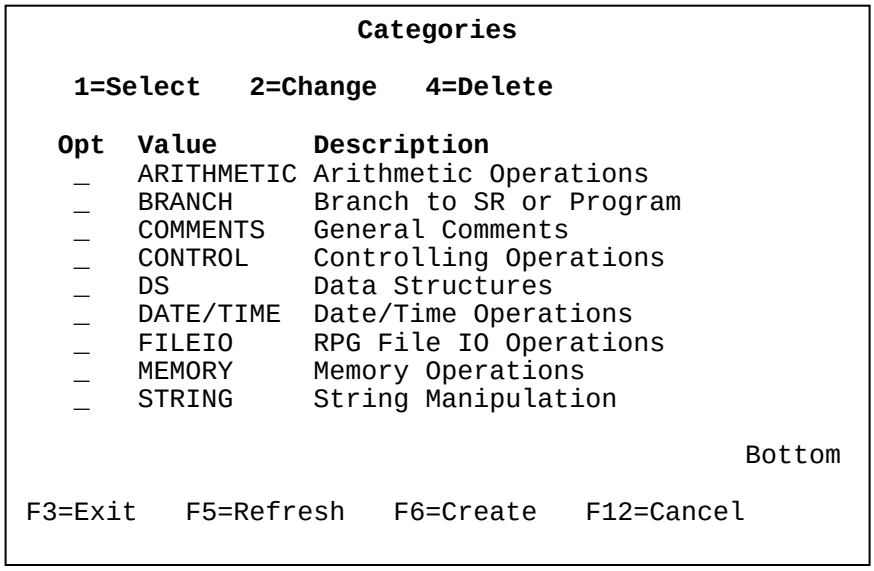
Snippet Categories (Maintaining)

Snippet categories allow you to group similar snippets together, which allows the user to quickly find the correct snippet to insert.

* Please note that vendor-supplied Categories cannot be changed or deleted.

To create, change or delete Snippet Categories:

1. Enter the SEU line command of **IX**.
2. Press **F4** to display the list of snippets.
3. When the list of snippets is display, position to the "Category" field on the top of the screen and press **F4**.
4. You will be presented with the following screen:



5. Press **F6** to create your own Category (you can then use that Category when defining new snippets) OR
6. Enter option **2** to change a Category description OR
7. Enter option **4** to delete a Category.

Sharing Snippets

If you created a snippet which you want to share with another organization, you can share the snippet easily by just sending it's source member.

A simple way to share a snippet with another organization:

1. Download the snippet source member into a PC text file using a file transfer program or FTP.
2. Send the PC text file via e-mail to the recipient.
3. The recipient will need to upload the PC text file into a source member and then update their snippet index database.

Linoma Software would appreciate receiving any snippets you feel would benefit other customers of the Toolbox. After you receive authorization from your organization, please send these snippets to support@linoma.com with the subject of "Snippet". Please include any details on the snippet in the e-mail message. We will need written permission from your organization before we can publish the snippet.

Support

Getting Support

Answers to common RPG Toolbox questions can be found within the support section of Linoma Software's web site at www.linomasoftware.com. Otherwise, please e-mail any RPG Toolbox questions to Linoma Software support at support@linomasoftware.com or call 402-944-4242.

Non-English Customers

The SEUPLUS line commands of RC (Retrieve Compile) and WC (Work with Compile) will retrieve information from compiler listings. If your compiler listings are not in the English language and if these line commands do not perform as expected, then please contact Linoma Software support. A product PTF may be available for your language.

History of the RPG Toolbox

The predecessor to the RPG Toolbox is a product called CVTILERPG (Convert to ILE RPG). It's primary purpose was for converting RPG III and RPG/400 source code to RPG IV fixed-form syntax.

Linoma originally created CVTILERPG because of their own frustrations with converting a customer's source code to RPG IV using IBM's limited tool. CVTILERPG was commercially released to the AS/400 community in 1996 and became immediately successful. CVTILERPG was priced affordably, could be downloaded from the internet (which was a fascination at that time) and did a thorough job in modernizing RPG.

Since that time, CVTILERPG has been used by customers all over the world to effectively convert and enhance literally hundreds of thousands of RPG programs.

While CVTILERPG was very effective at RPG conversion, it also proved itself as a valuable learning aid for programmers moving to RPG IV. CVTILERPG's solid functionality and educational aspects earned praise from well-known RPG experts and instructors. IBM was so impressed with CVTILERPG, they also incorporated this product into a course focused on modernizing RPG.

In mid-2001, CVTILERPG was renamed to Linoma's RPG Toolbox and was dramatically enhanced. This new version gave RPG developers even more productivity tools and features, including the capability to convert to the new free-form syntax and the addition of over 70 new SEU line commands.

The RPG Toolbox will continue to grow and improve as IBM creates new capabilities in RPG and with valuable feedback from our customers.

* As an interesting side-note, most of the Toolbox is itself written in RPG IV.

UnInstalling

RPGTOOLBOX can be removed from your IBM i by running the command:
DLTLICPGM LICPGM(4RPGBOX)

The library "RPGTOOLBOX" will be deleted when you run this command.

License agreement and limited warranty

READ CAREFULLY BEFORE USING

Linoma's RPG Toolbox ('RPGTOOLBOX') is a proprietary product of Linoma Software ('LINOMA SOFTWARE') and is protected by copyright laws and international treaties.

LINOMA SOFTWARE LICENSE AGREEMENT

This is a legal agreement between you, the user, and LINOMA SOFTWARE(the 'Agreement'). Loading RPGTOOLBOX onto your IBM i indicates your acceptance to be bound by the terms of this Agreement. If you do not wish to agree to the terms of this Agreement, you should promptly return the full software package and all accompanying items to LINOMA SOFTWARE.

RPGTOOLBOX LICENSE

A. LINOMA SOFTWARE grants you the right to use RPGTOOLBOX on your IBM i computer system.

B. You may not alter, modify, decompile, disassemble or reverse engineer RPGTOOLBOX, or otherwise attempt to reproduce the source code thereof.

C. You acknowledge that RPGTOOLBOX is provided pursuant to a license and all title and ownership to RPGTOOLBOX shall remain with LINOMA SOFTWARE or its licensors.

D. You may not use RPGTOOLBOX to modernize or convert any RPG source code owned by another organization unless that organization has also purchased a license to RPGTOOLBOX.

TERM

This license is effective until terminated. You may terminate it at any time by destroying RPGTOOLBOX together with all copies and merged portions in any form. It will also terminate upon conditions set forth elsewhere in this Agreement, or if you fail to comply with any term or condition of this Agreement.

LIMITED WARRANTY

LINOMA SOFTWARE will replace, at no charge, defective media on which RPGTOOLBOX is furnished that is returned within 60 days of the date of the original date of purchase. THE FOREGOING WARRANTY IS IN LIEU OF ALL OTHER WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE WITH RESPECT TO RPGTOOLBOX AND THE ACCOMPANYING WRITTEN MATERIALS.

LINOMA SOFTWARE does not warrant that the functions contained in RPGTOOLBOX meet your requirements or that the operation of RPGTOOLBOX will be uninterrupted or error free. The entire risk arising out of use or performance of RPGTOOLBOX remains with you.

IN NO EVENT WILL LINOMA SOFTWARE BE LIABLE TO YOU FOR ANY SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES, INCLUDING BUT NOT LIMITED TO ANY LOST PROFITS, LOST SAVINGS OR LOST BUSINESS OPPORTUNITIES ARISING OUT OF THE USE OR INABILITY TO USE RPGTOOLBOX EVEN IF LINOMA SOFTWARE HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

LINOMA SOFTWARE's entire liability and your exclusive remedy shall be (a) the replacement of any media not meeting LINOMA SOFTWARE's "Limited Warranty" and which is returned to LINOMA SOFTWARE during the Warranty period, or (b) if LINOMA SOFTWARE is unable to deliver replacement media which is free of defects in materials or workmanship, you may receive a refund of your purchase price by returning RPGTOOLBOX.

This limited warranty gives you specific legal rights. Some states provide other rights, and some states do not allow excluding or limiting implied warranties or limiting liability for incidental or consequential damages. As a result, the above limitations and or exclusions may not apply to you. Furthermore, some jurisdictions have statutory consumer product provisions which may supersede these provisions of the Agreement.

GENERAL

If any provision of this Agreement shall be unlawful, void or for any reason unenforceable, then that provision shall be deemed severable from this Agreement and shall not affect the validity and enforceability of the remaining provisions of this Agreement. This Agreement will be governed by the laws of the State of Nebraska.