Freedom/Pre-Compiler
Fine Tuning CA 2E Generation
User Guide

Version 1.0

HawkBridge Pty Ltd
3 Highett Road
Hampton, VIC 3188
Australia

http://www.HawkBridge..com.au

Table of Contents

Overview

CA 2E is quite powerful as a design tool and building business applications, yet there the limitations imposed by the generator that sometimes require the generated source to be modified prior to the object being compiled. This could stem from a need to interface to non-CA 2E applications with database files that do not fit the mould of "pure" CA 2E generated database files, or implement a feature of IBM i5 not supported by CA 2E such as ALIAS names for SQL tables.

To make it easier to modify generated CA 2E source we have developed the Freedom/Pre-Compiler product to enable fine tuning of the generated object source prior to compilation with script based language for performing common tasks such as scanning, updating, inserting and deleting source lines.

Shareware

This product is distributed as Shareware with a limited set of features enabled without the need to purchase a licence. The limitations of the free version of the product are as follows:

- Only the Source Member Pre-Compile Directive can be used
- Y* CALL HPRCPREDIR (xxxxxxxxxx &C &M &F &L)
- Only the first 9 commands per group accepted by processor
- Freeware comments imbedded into source member
- Implicit Quit command executed after 9th command
- Include and Execute commands will be ignored by processor

Should you wish to use the extended set of features of the product you are required to purchase a license from HawkBridge Pty Ltd for your specific AS/400. The authorization will then enable you to install and use the product on a single AS/400 only. The fully licensed version has no restrictions imposed apart from running on a single nominated AS/400.

Shareware products are provided without warranty either implied or expressed. It is your responsibility to ensure that the software is appropriate for your installation.

Prerequisites

The following prerequisites must be satisfied in order to use the tool:

- OS/400 V5R2M0 or above installed
- CA 2E 8.0, 8.1 or 8.5 Data Model

OS/400 V5R2M0 or above

The AS/400 objects in this release have been saved for V5R2M0. If you are on an earlier release, then contact us for a version that supports your version of OS/400.

Objects have been saved in a format that will enable Freedom/Pre-Compiler to be restored to V6R1 and V7R1 of OS/400.

CA 2E 8.0, 8.1 or 8.5

The tool has been developed for Release 8.0, 8.1 and 8.5 of CA 2E. If you are on a later release, then contact us for the upgraded version of the tool. Prior releases may work, but may not due to internal file changes to the CA 2E data model.

Configuring the Freedom/Pre-Compiler Tool

This section describes how to configure your environment so that the Freedom/Pre-Compiler will be invoked.

Add FREEDOMPCP Library to Compilation Job Descriptions

NOTE: This configuration option is the only one required for the Shareware version of the product.

Make sure that FREEDOMPCP is in your model job description library list for compilation. Ensure you check your model profile for the correct job description to update as it is possible for two job descriptions to be used in your model. One is for generation and the other is for compilation. The Freedom/Pre-Compiler tool only needs you to change the compilation job description.

Setup the Global Pre-Compile Exit Program

The global pre-compile exit program is defined in the CA 2E shipped dataarea YBRTPXA located in library Y1SY or the library that you installed the CA 2E Toolkit Product Library.

To enable the Freedom/Pre-Compiler to be invoked as the global pre-compile exit program you need to execute the following command:

- CHGDTAARA DTAARA(YBRTPXA (1 20)) VALUE(‘HPRCPREPGM')

To disable the Freedom/Pre-Compiler from being invoked as the global pre-compile exit program you need to execute the following command:

- CHGDTAARA DTAARA(YBRTPXA (1 20)) VALUE(‘ ')

Setup the Global Post-Compile Exit Program

The global post-compile exit program is defined in the CA 2E shipped dataarea YBRTPXA located in library Y1SY or the library that you installed the CA 2E Toolkit Product Library.

To enable the Freedom/Pre-Compiler to be invoked as the global post-compile exit program you need to execute the following command:

- CHGDTAARA DTAARA(YBRTPXA (21 20)) VALUE(‘HPRCPSTPGM')

To disable the Freedom/Pre-Compiler from being invoked as the global post-compile exit program you need to execute the following command:

- CHGDTAARA DTAARA(YBRTPXA (21 20)) VALUE(‘ ')

Using the Freedom/Pre-Compiler Tool

To use the Freedom/Pre-Compiler tool to fine tune CA 2E generated source for an EXCEXTFUN function with source member name UUAAXFR of type RP4 so that the P0RTN entry parameter is removed follow these steps:

Create an EXCUSRSRC function to invoke the Source Member Pre-Compile Directive

For this example assume the EXCUSRSRC function has a source member name of UUABUFR of type RP4. Enter the following source lines for the EXCUSRSRC function:

```
 Columns . . . :    1 100                                  Edit                                                MYGEN/QRPGLESRC
 SEU==>                                                                                                            UUABUFR
 FMT *  ..... *. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8 ...+... 9 ...+... 0
        ********** Beginning of data **************************************************************************************
0001.00      ** This function removes the P0RTN parameter from *ENTRY parameters                                  110514
0002.00 >>>>>Y* CALL HPRCPREDIR (UUABUFR &C &M &F &L)                                                           110514
        ***************** End of data **************************************************************************************
 F3=Exit   F4=Prompt   F5=Refresh   F9=Retrieve   F10=Cursor   F11=Toggle
 F16=Repeat find       F17=Repeat change          F24=More keys
```

Insert call to EXCUSRSRC function in the EXCEXTFUN function

Edit the EXCEXTFUN function action diagram and enter the following statement to invoke the Freedom/Pre-Compiler during the compilation of the EXCEXTFUN program:

```
 EDIT ACTION DIAGRAM            Edit     MYMDL      My File
FIND=>                                              My EXCEXTFUN
I(C,I,S)F=Insert construct          I(X,O)F=Insert alternate case
I(A,E,Q,*,+,-,=,=A)F=Insert action  IMF=Insert message
     > My EXCEXTFUN
___  .--                                                                          <<<
___  . zRmv Parameter P0RTN – My Pre-Compiler Overrides  * UUABUFR               <<<
___  '--
F3=Prev block   F5=User points    F6=Cancel pending moves    F23=More options
F7=Find         F8=Bookmark       F9=Parameters              F24=More keys
```

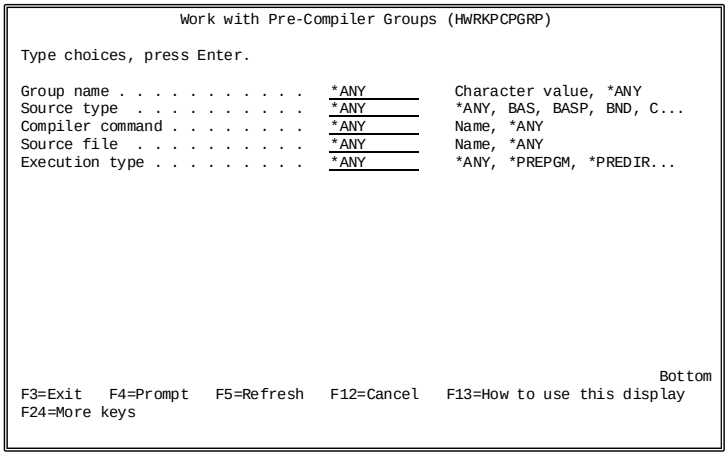
The above action diagram statement can be included anywhere in the EXCEXTFUN function or lower level internal functions that are called by the EXCEXTFUN function.

Create a Freedom/Pre-Compiler Group

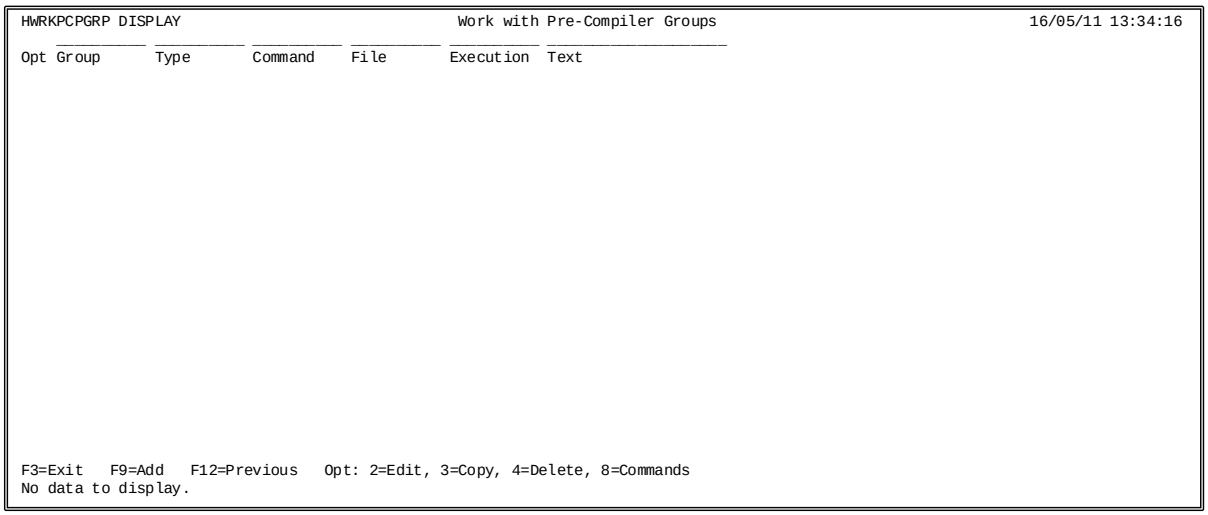
Make sure that FREEDOMPCP is in your library list. The minimum library list to run the tool is:

- QTEMP
- FREEDOMPCP

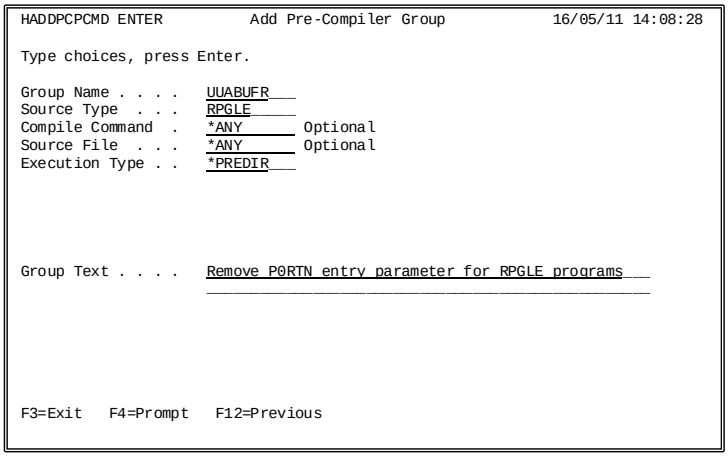
Prompt the command HWRKPCPGRP to start the Freedom/Pre-Compiler editor as follows then press Enter:



From the Work with Pre-Compiler Groups display that next appears press F9 to proceed to the Add Pre-Compiler Group display.



From the Add Pre-Compiler Group display that next appears enter the following details then press Enter. When defining a Pre-Compiler Group the source member name of the EXCUSRSRC is the recommended name to use for the group. This enables you to easily identify the relationship between the pre-compiler group and EXCUSRSRC function. You could choose to use any other name for the pre-compiler group.



Define the Freedom/Pre-Compiler Commands for the new Pre-Compiler Group

From the Work with Pre-Compiler Groups display that next appears type option 8 against the newly created Pre-Compiler Group to proceed to the Work with Pre-Compiler Commands display.

```
HWRKPCPGRP DISPLAY                          Work with Pre-Compiler Groups                          16/05/11 13:34:16
                 _________  _________  _________  _________  _________  ____________________
Opt Group        Type       Command    File       Execution  Text
 8  UUABUFR      RPGLE      *ANY       *ANY       *PREDIR    Remove P0RTN entry parameter for RPGLE programs

F3=Exit   F9=Add   F12=Previous   Opt: 2=Edit, 3=Copy, 4=Delete, 8=Commands
Record added.
```

From the Work with Pre-Compiler Commands display that next appears press F9 to proceed to the Add Pre-Compiler Command display.

```
HWRKPCPCMD DISPLAY                          Work with Pre-Compiler Commands                        16/05/11 17:17:35
Group : UUABUFR    RPGLE      *ANY       *ANY       *PREDIR    Remove P0RTN entry parameter for RPGLE programs

F3=Exit   F9=Add   F12=Previous   Opt: 2=Edit, 3=Copy, 4=Delete, 5=Display
No data to display.
```

From the Add Pre-Compiler Command display that next appears enter the following details then press Enter.

```
HEDTPCPCMD ADD           Add Pre-Compiler Command           16/05/11 17:25:03

Type choices, press Enter.

Source Member  . :   UUABUFR
Source Type  . . :   RPGLE
Compile Command  :   *ANY
Source File  . . :   *ANY
Execution Type . :   *PREDIR

Command Sequence .   010

F3=Exit   F12=Previous
```

From the Edit Pre-Compiler Command display that next appears enter the following details then press Enter and confirm.

```
HEDTPCPCMD ADD          Edit Pre-Compiler Command          16/05/11 17:28:11

Type changes, press Enter.

Group Name . . . :   UUABUFR
Source Type  . . :   RPGLE
Compile Command  :   *ANY
Source File  . . :   *ANY
Execution Type . :   *PREDIR

Command Sequence .   10
Command  . . . . .   Scan_____
Relational Op  . .   EQ
                     :...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5
Compare String . .   >>>>>Y* CALL HPRCPREDIR (UUABUFR_________________
                     ____________________________________________________
Compare Length . .   32
Source Start . . .   ___
Source End . . . .   ___
Replace String . .   <<<<<** CALL HPRCPREDIR (UUABUFR_________________
                     ____________________________________________________

F3=Exit   F4=Prompt   F12=Previous
```

The above Pre-Compiler Command will scan for the compare string which is the CA 2E pre-compiler directive that invoked Freedom/Pre-Compiler to process this pre-compiler group. If the compare string is found in the source member it will be replaced with the replacement string. This will change the CA 2E pre-compiler directive so that it will not execute the next time the source member is used to compile the object without first regenerating the source member.

From the Work with Pre-Compiler Commands display that next appears press F9 to proceed to the Add Pre-Compiler Command display.

```
HWRKPCPCMD DISPLAY                    Work with Pre-Compiler Commands                    16/05/11 17:46:36
Group : UUABUFR    RPGLE      *ANY       *ANY       *PREDIR    Remove P0RTN entry parameter for RPGLE programs
 ___ __________ __ __________ __ __________ __ __________ __ ____________________________________________________
Opt Seq Command                Op Compare String                                Len Str End Replace String
 _   10 Scan                     EQ >>>>>Y* CALL HPRCPREDIR (UUABUFR               32   0   0 <<<<<** CALL HP

                                                                                                      Bottom
F3=Exit   F9=Add   F12=Previous   Opt: 2=Edit, 3=Copy, 4=Delete, 5=Display
Record added.
```

From the Add Pre-Compiler Command display that next appears enter the following details then press Enter.

```
HEDTPCPCMD ADD          Add Pre-Compiler Command           16/05/11 17:25:03

Type choices, press Enter.

Group Name . . . :   UUABUFR
Source Type  . . :   RPGLE
Compile Command  :   *ANY
Source File  . . :   *ANY
Execution Type . :   *PREDIR

Command Sequence .   020

F3=Exit   F12=Previous
```

From the Edit Pre-Compiler Command display that next appears enter the following details then press Enter and confirm.

```
HEDTPCPCMD ADD          Edit Pre-Compiler Command         16/05/11 17:28:11

Type changes, press Enter.

Group Name . . . :   UUABUFR
Source Type  . . :   RPGLE
Compile Command  :   *ANY
Source File  . . :   *ANY
Execution Type . :   *PREDIR

Command Sequence .    20
Command  . . . . .   Scan______
Relational Op  . .   EQ
                     :...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5
Compare String . .        C                   PARM                    P
                     0RTN
Compare Length . .    55
Source Start . . .   ___
Source End . . . .   ___
Replace String . .   <<<<<C*                  PARM                    P
                     0RTN

F3=Exit   F4=Prompt   F12=Previous
```

The above Pre-Compiler Command will scan for the compare string which is the parameter calculation specification for the implicit return code field P0RTN. If the compare string is found in the source member it will be replaced with the replacement string. This will change the source line so that it becomes a comment line and not included in the compilation of the object.

From the Work with Pre-Compiler Commands display that next appears press F3 twice to exit the Freedom/Pre-Compiler editor.

```
HWRKPCPCMD DISPLAY                            Work with Pre-Compiler Commands                            16/05/11 17:46:36
Group : UUABUFR    RPGLE      *ANY       *ANY       *PREDIR    Remove P0RTN entry parameter for RPGLE programs

Opt Seq Command                       Op Compare String                                        Len Str End Replace String
 _   10 Scan                          EQ >>>>>Y* CALL HPRCPREDIR (UUABUFR                       32   0   0 <<<<<** CALL HP
 _   20 Scan                          EQ      C                   PARM                  P0RTN   55   0   0 <<<<<C*

                                                                                                                    Bottom
F3=Exit   F9=Add   F12=Previous   Opt: 2=Edit, 3=Copy, 4=Delete, 5=Display
Record added.
```

Re-Generate and Compile the EXCEXTFUN function

Make sure that FREEDOMPCP is in your model job description library list for generation and compilation. There are no other Freedom/Pre-Compiler or CA 2E configuration options required in order to invoke the Pre-Compiler Directive (*PREDIR) command processor used in this example.

Submit the EXCEXTFUN function for re-generation and compilation so that it now includes the call to the EXCUSRSRC function to invoke the Freedom/Pre-Compiler to remove the P0RTN from the entry parameters.

Once the compilation of the EXCEXTFUN program has completed you can review the generated source and note the changes made by Freedom/Pre-Compiler.

Freedom/Pre-Compiler Command Summary

The Freedom/Pre-Compiler Commands allow you to do many different types of operations on your source file as well as executing OS/400 commands. This section summarizes the pre-compiler commands that are available and also organises them into categories.

For detailed information about specific pre-compiler commands see the next section Freedom/Pre-Compiler Commands.

The following table is a summary of the specifications for each pre-compiler command.

- An empty column indicates that the field must be blank
- All underlined fields are required

Command	Relational Op	Compare String	Compare Length	Source Start	Source End	Replace String
Abort		Return Code				
Backward				Number of Lines		
Delete	Relational Op	Compare String	Compare Length	Source Start	Source End	
ElseIf	Relational Op	Compare String	Compare Length	Source Start	Source End	Variable Name
ElseOnErr	Relational Op	Return Code				
EndIf						
EndOnErr						
Execute		OS400 Command				
Forward				Number of Lines		
If	Relational Op	Compare String	Compare Length	Source Start	Source End	Variable Name
Include		Group Name				
Insert	Pre-Built Ind		Source Length	Source Start		Source String
InsString	Variable Ind	Variable Name	Source Length	Source Start		Source String
OnErr	Relational Op	Return Code				
Quit						
RstSeq	Variable Ind	Variable Name				Sequence Nbr
SavSeq		Variable Name				
Scan	Relational Op	Compare String	Compare Length	Source Start	Source End	Replace String
SubString		Variable Name	Source Length	Source Start		
Update	Relational Op or Pre-Built Ind	Compare String	Compare Length	Source Start	Source End	Source String

Branching Pre-Compiler Commands

The branching pre-compiler commands are shown in the following table.

Command	Description
Abort	Stops processing the pre-compiler commands and exits abnormally with a specific return code or default return code of Y2U9999.
Execute	Transfers control to the OS400 Command specified.
Include	Transfers control to the pre-compiler group specified.
Quit	Stops processing the pre-compiler commands and exist normally.

Compare Pre-Compiler Commands

The compare pre-compiler commands are shown in the following table.

Command	Description
Delete	Deletes the current source line based on the source line cursor.
ElseIf	Ends a preceding If or ElseIf group of pre-compiler commands and allows a series of pre-compiler commands to be processed if a condition is met.
ElseOnErr	Ends a preceding OnErr or ElseOnErr group of pre-compiler commands and allows a series of pre-compiler commands to be processed if an error condition is met.
If	Allows a series of pre-compiler commands to be processed if a condition is met.
OnErr	Allows a series of pre-compiler commands to be processed if an error condition is met.
Scan	Searches for a matching source line starting with the next source line after the current source line cursor.
Update	Updates the current source line based on the source line cursor.

Error Handling Pre-Compiler Commands

The error handling pre-compiler commands are shown in the following table.

Command	Description
Abort	Stops processing the pre-compiler commands and exits abnormally with a specific return code or default return code of Y2U9999.
ElseOnErr	Ends a preceding OnErr or ElseOnErr group of pre-compiler commands and allows a series of pre-compiler commands to be processed if an error condition is met.
EndOnErr	Ends a preceding OnErr or ElseOnErr group of pre-compiler commands.
OnErr	Allows a series of pre-compiler commands to be processed if an error condition is met.

The error handling pre-compiler commands are only performed when there is an error condition resulting from the previous pre-compiler command failing. An error condition is determined when non-blank return codes are returned from pre-compiler commands.

Once the error condition on an OnErr or ElseOnErr pre-compiler command is true the error condition is reset to normal and the value of the system return code is set to blanks.

Source Line Positioning Pre-Compiler Commands

The source line positioning pre-compiler commands are shown in the following table.

Command	Description
Backward	Moves the source line cursor a specified number of lines backwards from the current position.
Forward	Moves the source line cursor a specified number of lines forwards from the current position.
RstSeq	Updates the current source line cursor in memory.
Scan	Searches for a matching source line starting with the next source line after the current source line cursor.

Source Line Update Pre-Compiler Commands

The source line update pre-compiler commands are shown in the following table.

Command	Description
Delete	Deletes the current source line based on the source line cursor.
Insert	Creates a new source file line after the source line identified by the current source line cursor.
InsString	Updates the current source line in memory.
Scan	Searches for a matching source line starting with the next source line after the current source line cursor.
Update	Updates the current source line based on the source line cursor.

Structured Programming Pre-Compiler Commands

The structured programming pre-compiler commands are shown in the following table.

Command	Description
ElseIf	Ends a preceding If or ElseIf group of pre-compiler commands and allows a series of pre-compiler commands to be processed if a condition is met.
EndIf	Ends a preceding If or ElseIf group of pre-compiler commands.
If	Allows a series of pre-compiler commands to be processed if a condition is met.

Variable Pre-Compiler Commands

The variable pre-compiler commands are shown in the following table.

Command	Description
ElseIf	Ends a preceding If or ElseIf group of pre-compiler commands and allows a series of pre-compiler commands to be processed if a condition is met.
If	Allows a series of pre-compiler commands to be processed if a condition is met.
InsString	Updates the current source line in memory.
RstSeq	Updates the current source line cursor in memory.
SavSeq	Updates the variable value with the current source line cursor in memory.
SubString	Updates the current variable value for the specified Variable Name.

Freedom/Pre-Compiler Commands

This section describes, in alphabetical order, each Freedom/Pre-Compiler Command in detail with examples.

Abort

Command	Relational Op	Compare String	Compare Length	Source Start	Source End	Replace String
Abort		Return Code				

The Abort pre-compiler command stops processing the pre-compiler commands in the current pre-compiler group and exits abnormally.

If Return Code is not specified, then a default return code of Y2U9999 is used to signal an abnormal termination of the pre-compiler group.

If the Abort pre-compiler command is executed within a pre-compiler group invoked from the Include pre-compiler command, then return code can be monitored by an OnErr...ElseOnErr...EndOnErr block in the pre-compiler group that initiated the Include pre-compiler command.

The following figure is an example of the Abort pre-compiler command that exits with return code value of 'ERROR'.

```
Command  . . . . .   Abort____
Relational Op  . .   __
                     :...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5
Compare String . .   ERROR_____________________________________________
                     ________________________________________________
Compare Length . .   __5
Source Start . . .   ___
Source End . . . .   ___
Replace String . .   ________________________________________________
                     ________________________________________________
```

Backward

Command	Relational Op	Compare String	Compare Length	Source Start	Source End	Replace String
Backward				Number of Lines		

The Backward pre-compiler command moves the source line cursor a specified number of lines backwards from the current position.

Number of Lines must be specified as an integer in the range of 1 to 999.

The number of lines is based on the before image of the source member and does not include updates that may have already been processed by the pre-compiler commands. For instance using the Delete pre-compiler command to remove a source line and then using the Backward pre-compiler command will not move the source line cursor.

The following figure is an example of the Backward pre-compiler command that moves the source line cursor backward 5 lines.

```
Command  . . . . .   Backward__
Relational Op  . .   __
                     :...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5
Compare String . .   ________________________________________________
                     ________________________________________________
Compare Length . .   ___
Source Start . . .   __5
Source End . . . .   ___
Replace String . .   ________________________________________________
                     ________________________________________________
```

Delete

Command	Relational Op	Compare String	Compare Length	Source Start	Source End	Replace String
Delete	Relational Op	Compare String	Compare Length	Source Start	Source End	

The Delete pre-compiler command deletes the current source line based on the source line cursor.

A source line positioning pre-compiler command must be used before the use of the Delete pre-compiler command in order to initialise the source line cursor.

An optional comparison can be performed to only delete the source line if it matches the comparison criteria specified. If the comparison fails, then return code is set to 'Y2U0022' and must be handled using an error handling pre-compiler command. Failure to monitor for the return code will cause the pre-compiler group to exit abnormally with return code 'Y2U0022'.

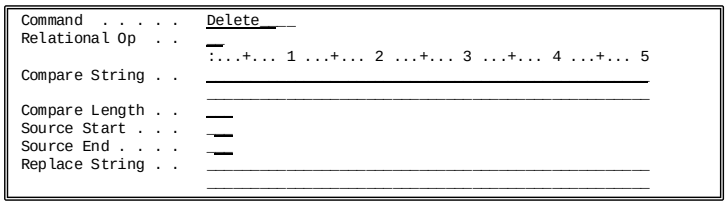
If Relational Op is either EQ or NE, then the current source line based on the source line cursor from the specified Source Start to Source End will be scanned and compared to the Compare String for the specified Compare Length. A Relational Op of EQ will perform an equality comparison and NE will perform an inequality comparison.

If Compare Length is not specified, then the length of the search argument is defaulted to the length of the Compare String excluding trailing blanks.

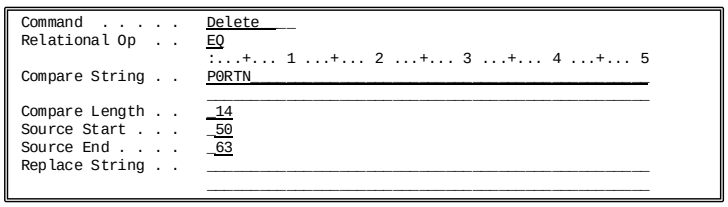
If Source Start is not specified, then the scan starts at the beginning of the current source line.

If Source End is not specified, then the scan completes at the end of the current source line.

The following figure is an example of the Delete pre-compiler command that deletes the current source line.



The following figure is an example of the Delete pre-compiler command that deletes the current source line if 'P0RTNbbbbbbbbb' is found between column 50 and 63.



ElseIf

Command	Relational Op	Compare String	Compare Length	Source Start	Source End	Replace String
ElseIf	Relational Op	Compare String	Compare Length	Source Start	Source End	Variable Name

The ElseIf pre-compiler command ends a preceding If or ElseIf group of pre-compiler commands and allows a series of pre-compiler commands to be processed if a condition is met. The pre-compiler commands controlled by the ElseIf pre-compiler command are performed when the comparison is true and the previous If or ElseIf pre-compiler command comparison was false.

An ElseIf pre-compiler command must be preceded by an If or ElseIf pre-compiler command.

An ElseIf pre-compiler command must be succeeded by an ElseIf or EndIf pre-compiler command.

If a Compare String is not specified, then the pre-compiler commands controlled by the ElseIf pre-compiler command are performed and then control passes to the next EndIf pre-compiler command.

An optional comparison can be performed to only perform pre-compiler commands controlled by the ElseIf pre-compiler command if it matches the comparison criteria specified. If the comparison fails, then control passes to the next ElseIf or EndIf pre-compiler command. If the comparison is true, then the pre-compiler commands controlled by the ElseIf pre-compiler command are performed and then control passes to the next EndIf pre-compiler command.

If Relational Op is either EQ or NE, then the current source line based on the source line cursor from the specified Source Start to Source End will be scanned and compared to the Compare String for the specified Compare Length. A Relational Op of EQ will perform an equality comparison and NE will perform an inequality comparison.

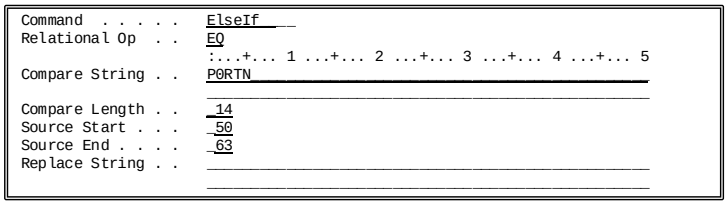
If Variable Name is specified, then the current variable value from the specified Source Start to Source End will be scanned and compared to the Compare String for the specified Compare Length. A Relational Op of EQ will perform an equality comparison and NE will perform an inequality comparison.

If Compare Length is not specified, then the length of the search argument is defaulted to the length of the Compare String excluding trailing blanks.

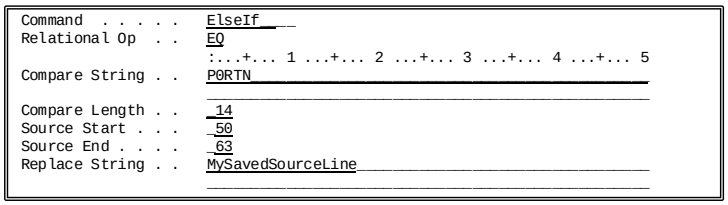
If Source Start is not specified, then the scan starts at the beginning of the current source line or variable value.

If Source End is not specified, then the scan completes at the end of the current source line or variable value.

The following figure is an example of the ElseIf pre-compiler command that compares the current source line between column 50 and 63 with compare string 'P0RTNbbbbbbbbb'.



The following figure is an example of the ElseIf pre-compiler command that compares the current value of variable 'MySavedSourceLine' between column 50 and 63 with compare string 'P0RTNbbbbbbbbb'.



ElseOnErr

Command	Relational Op	Compare String	Compare Length	Source Start	Source End	Replace String
ElseOnErr	Relational Op	Return Code				

The ElseOnErr pre-compiler command ends a preceding OnErr or ElseOnErr group of pre-compiler commands and allows a series of pre-compiler commands to be processed if an error condition is met. The pre-compiler commands controlled by the ElseOnErr pre-compiler command are performed when the error comparison is true and the previous OnErr or ElseOnErr pre-compiler command comparison was false.

An ElseOnErr pre-compiler command must be preceded by an OnErr or ElseOnErr pre-compiler command.

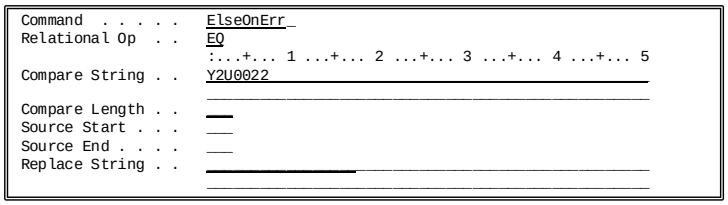
An ElseOnErr pre-compiler command must be succeeded by an ElseOnErr or EndOnErr pre-compiler command.

If a Return Code is not specified, then the pre-compiler commands controlled by the ElseOnErr pre-compiler command are performed and then control passes to the next EndOnErr pre-compiler command.

An optional error comparison can be performed to only perform pre-compiler commands controlled by the ElseOnErr pre-compiler command if it matches the error comparison criteria specified. If the error comparison fails, then control passes to the next ElseOnErr or EndOnErr pre-compiler command. If the error comparison is true, then the pre-compiler commands controlled by the ElseOnErr pre-compiler command are performed and then control passes to the next EndOnErr pre-compiler command.

If Relational Op is either EQ or NE, then the current value of the system return code will be compared to the Return Code specified. A Relational Op of EQ will perform an equality error comparison and NE will perform an inequality error comparison.

The following figure is an example of the ElseOnErr pre-compiler command that compares the current value of the system return code with compare string 'Y2U0022'.



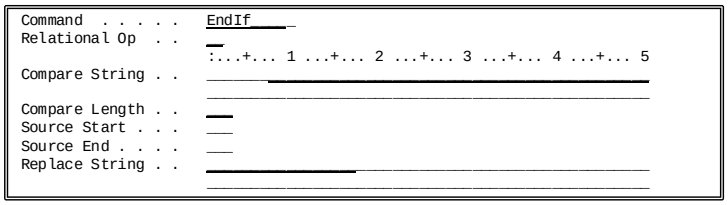
EndIf

Command	Relational Op	Compare String	Compare Length	Source Start	Source End	Replace String
EndIf						

The EndIf pre-compiler command ends a preceding If or ElseIf group of pre-compiler commands.

An EndIf pre-compiler command must be preceded by an If or ElseIf pre-compiler command.

The following figure is an example of the EndIf pre-compiler command.



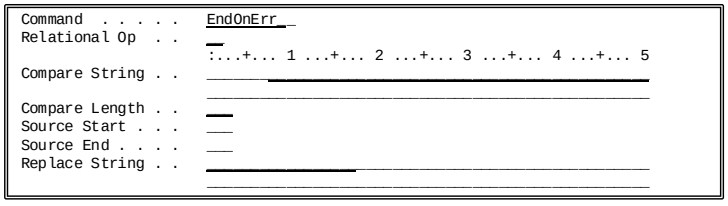
EndOnErr

Command	Relational Op	Compare String	Compare Length	Source Start	Source End	Replace String
EndOnErr						

The EndOnErr pre-compiler command ends a preceding OnErr or ElseOnErr group of pre-compiler commands.

An EndOnErr pre-compiler command must be preceded by an OnErr or ElseOnErr pre-compiler command.

The following figure is an example of the EndOnErr pre-compiler command.



Execute

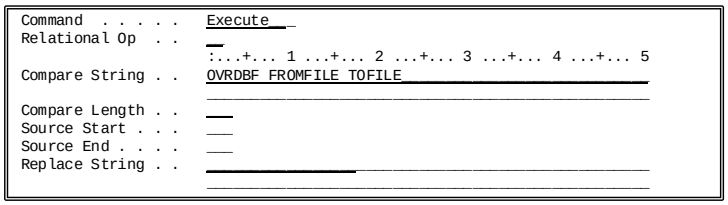
Command	Relational Op	Compare String	Compare Length	Source Start	Source End	Replace String
Execute		OS400 Command				

The Execute pre-compiler command transfers control to the OS400 Command specified.

NOTE: This pre-compiler command is not available in the Shareware version of the product.

The OS400 Command must be specified and must define a valid OS400 command that can be executed. Up to 100 characters can be specified for the OS Command which is executed without increasing the OS400 command invocation level.

The following figure is an example of the Execute pre-compiler command to perform an override to a file.



Forward

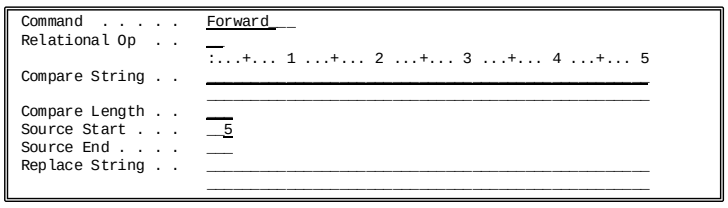
Command	Relational Op	Compare String	Compare Length	Source Start	Source End	Replace String
Forward				Number of Lines		

The Forward pre-compiler command moves the source line cursor a specified number of lines forwards from the current position.

Number of Lines must be specified as an integer in the range of 1 to 999.

The number of lines is based on the before image of the source member and does not include updates that may have already been processed by the pre-compiler commands. For instance using the Insert pre-compiler command to add a source line and then using the Forward pre-compiler command will not include the newly created source line in the count of lines to move.

The following figure is an example of the Forward pre-compiler command that moves the source line cursor forward 5 lines.



If

Command	Relational Op	Compare String	Compare Length	Source Start	Source End	Replace String
If	Relational Op	Compare String	Compare Length	Source Start	Source End	Variable Name

The If pre-compiler command allows a series of pre-compiler commands to be processed if a condition is met. The pre-compiler commands controlled by the If pre-compiler command are performed when the comparison is true.

An If pre-compiler command must be succeeded by an ElseIf or EndIf pre-compiler command.

A comparison is performed to only perform pre-compiler commands controlled by the If pre-compiler command if it matches the comparison criteria specified. If the comparison fails, then control passes to the next ElseIf or EndIf pre-compiler command. If the comparison is true, then the pre-compiler commands controlled by the If pre-compiler command are performed and then control passes to the next EndIf pre-compiler command.

Relational Op must be specified as either EQ or NE, and Compare String must also be specified. The current source line based on the source line cursor from the specified Source Start to Source End will be scanned and compared to the Compare String for the specified Compare Length. A Relational Op of EQ will perform an equality comparison and NE will perform an inequality comparison.

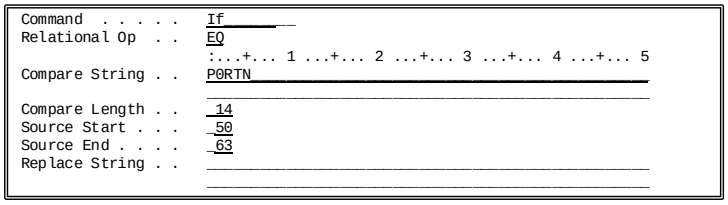
If Variable Name is specified, then the current variable value from the specified Source Start to Source End will be scanned and compared to the Compare String for the specified Compare Length. A Relational Op of EQ will perform an equality comparison and NE will perform an inequality comparison.

If Compare Length is not specified, then the length of the search argument is defaulted to the length of the Compare String excluding trailing blanks.

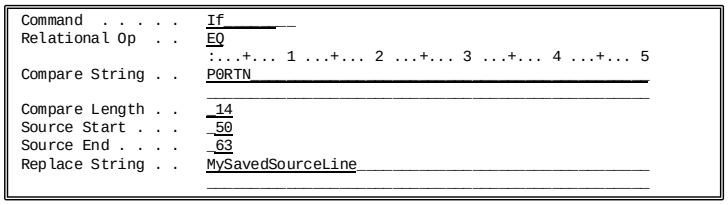
If Source Start is not specified, then the scan starts at the beginning of the current source line or variable value.

If Source End is not specified, then the scan completes at the end of the current source line or variable value.

The following figure is an example of the If pre-compiler command that compares the current source line between column 50 and 63 with compare string 'P0RTNbbbbbbbbb'.



The following figure is an example of the If pre-compiler command that compares the current value of variable 'MySavedSourceLine' between column 50 and 63 with compare string 'P0RTNbbbbbbbbb'.



Include

Command	Relational Op	Compare String	Compare Length	Source Start	Source End	Replace String
Include		Group Name				

The Include pre-compiler command transfers control to the pre-compiler group specified by Group Name.

NOTE: This pre-compiler command is not available in the Shareware version of the product.

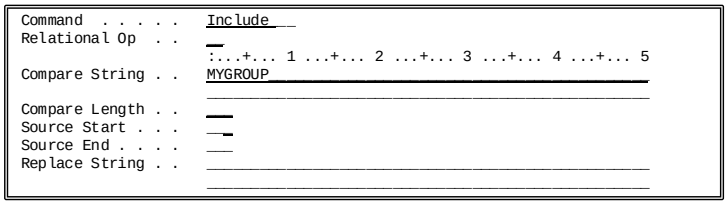
Pre-compiler groups are identified by the following attributes:

- Group Name,
- Source Type,
- Compile Command,
- Source File, and
- Execution Type.

The pre-compiler group specified by Group Name must exist for one of the matching lookup patterns:

- Current Source Type, current Compile Command, current Source File and current Execution Type;
- Current Source Type, Compile Command of '*ANY', current Source File and current Execution Type;
- Current Source Type, Compile Command of '*ANY', Source File of '*ANY' and current Execution Type;

The following figure is an example of the Include pre-compiler command to transfer control to the 'MYGROUP' pre-compiler group.



Insert

Command	Relational Op	Compare String	Compare Length	Source Start	Source End	Replace String
Insert	Pre-Built Ind		Source Length	Source Start		Source String

The Insert pre-compiler command creates a new source file line after the source line identified by the current source line cursor.

Multiple Insert pre-compiler commands performed in succession without changing the current source line cursor will create the source line after the last created source line.

The Pre-Built Ind is specified as '**' and indicates that the current source line value in memory is to be used for the new source line to be created. The current source line value in memory may have been built by the InsString pre-compiler command prior to processing the Insert pre-compiler command.

If the Pre-Built Ind is specified, then Source String, Source Length and Source Start are not allowed.

If the Pre-Built Ind is not specified, then Source String must be specified.

If Source String is specified, then that string will be inserted into the current source line value in memory from the specified Source Start for Source Length prior to the new source line being created.

If Source Length is not specified, then the length is defaulted to the length of the Source String excluding trailing blanks.

If Source Start is not specified, then the Source String is inserted at the beginning of the current source line value.

The following figure is an example of the Insert pre-compiler command that creates a new source line using the current source line value.

```
Command  . . . . .   Insert__
Relational Op  . .   **
                     :...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5
Compare String . .   ____________________________________________________
                     ____________________________________________________
Compare Length . .   ___
Source Start . . .   __
Source End . . . .   ___
Replace String . .   ____________________________________________________
                     ____________________________________________________
```

The following figure is an example of the Insert pre-compiler command that creates a new source line set to '<<<<<** Test Insert'.

```
Command  . . . . .   Insert__
Relational Op  . .   __
                     :...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5
Compare String . .   ____________________________________________________
                     ____________________________________________________
Compare Length . .   100
Source Start . . .   __1
Source End . . . .   ___
Replace String . .   <<<<<** Test Insert____________________________________
                     ____________________________________________________
```

InsString

Command	Relational Op	Compare String	Compare Length	Source Start	Source End	Replace String
InsString	Variable Ind	Variable Name	Source Length	Source Start		Source String

The InsString pre-compiler command updates the current source line in memory.

The Variable Ind is specified as '**' and indicates that the current variable value is to be used to update the current source line in memory.

If the Variable Ind is specified, then Source String is not allowed.

If the Variable Ind is not specified, then Variable Name is not allowed and Source String must be specified.

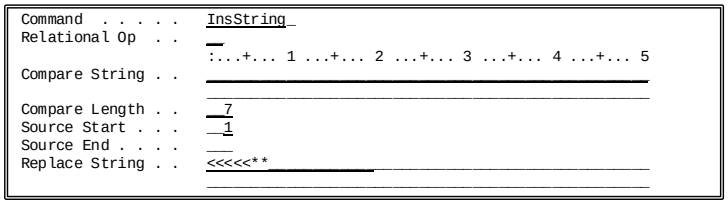
If Variable Name is specified, then the current variable value will be inserted into the current source line value in memory from the specified Source Start for Source Length.

If Source String is specified, then that string will be inserted into the current source line value in memory from the specified Source Start for Source Length.

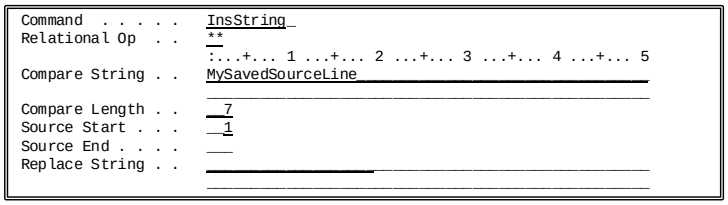
If Source Length is not specified, then the length is defaulted to the length of the Source String excluding trailing blanks.

If Source Start is not specified, then the Source String is inserted at the beginning of the current source line value.

The following figure is an example of the InsString pre-compiler command that updates the current source line value in memory between column 1 and 7 using the string '<<<<<**'.



The following figure is an example of the InsString pre-compiler command that updates the current source line value in memory between column 1 and 7 using the current variable value from MySavedSourceLine.



OnErr

Command	Relational Op	Compare String	Compare Length	Source Start	Source End	Replace String
OnErr	Relational Op	Return Code				

The OnErr pre-compiler command allows a series of pre-compiler commands to be processed if an error condition is met. The pre-compiler commands controlled by the OnErr pre-compiler command are performed when the error comparison is true.

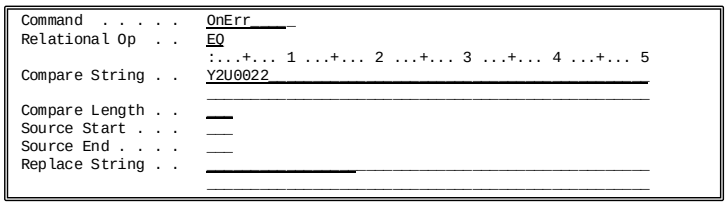
An OnErr pre-compiler command must be succeeded by an ElseOnErr or EndOnErr pre-compiler command.

If a Return Code is not specified, then the pre-compiler commands controlled by the OnErr pre-compiler command are performed and then control passes to the next EndOnErr pre-compiler command.

An optional error comparison can be performed to only perform pre-compiler commands controlled by the OnErr pre-compiler command if it matches the error comparison criteria specified. If the error comparison fails, then control passes to the next ElseOnErr or EndOnErr pre-compiler command. If the error comparison is true, then the pre-compiler commands controlled by the OnErr pre-compiler command are performed and then control passes to the next EndOnErr pre-compiler command.

If Relational Op is either EQ or NE, then the current value of the system return code will be compared to the Return Code specified. A Relational Op of EQ will perform an equality error comparison and NE will perform an inequality error comparison.

The following figure is an example of the OnErr pre-compiler command that compares the current value of the system return code with compare string 'Y2U0022'.



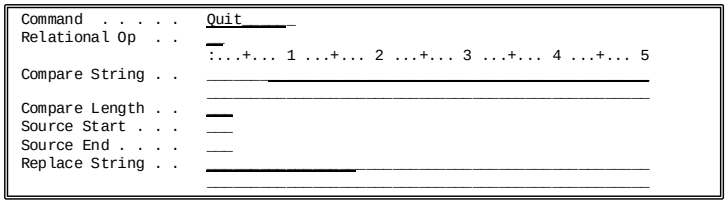
Quit

Command	Relational Op	Compare String	Compare Length	Source Start	Source End	Replace String
Quit						

The Quit pre-compiler command stops processing the pre-compiler commands in the current pre-compiler group and exits normally.

If the Quit pre-compiler command is executed within a pre-compiler group invoked from the Include pre-compiler command, then control returns to the pre-compiler group that initiated the Include pre-compiler command.

The following figure is an example of the Quit pre-compiler command.



RstSeq

Command	Relational Op	Compare String	Compare Length	Source Start	Source End	Replace String
RstSeq	Variable Ind	Variable Name				Sequence Nbr

The RstSeq pre-compiler command updates the current source line cursor in memory.

This pre-compiler command is used to position the current source line cursor in memory so that the next Insert, Delete or Update will be performed on the matching source line; the next Backward or Forward will start from the matching source line; or the next Scan will be performed on the next matching source line.

The Variable Ind is specified as '**' and indicates that the current variable value is to be used to update the current source line cursor in memory.

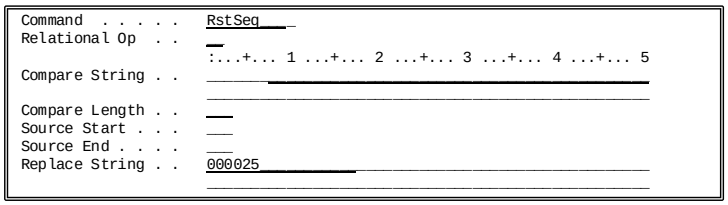
If the Variable Ind is specified, then Sequence Nbr is not allowed.

If the Variable Ind is not specified, then Variable Name is not allowed and Sequence Nbr must be specified and be an integer in the format 999999 including all leading zeros.

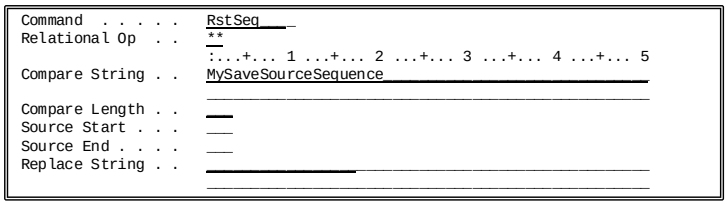
If Variable Name is specified, then the current variable value will be used to update the current source line cursor value in memory.

If Sequence Nbr is specified, then that string will be used to update the current source line cursor value in memory.

The following figure is an example of the RstSeq pre-compiler command that updates the current source line cursor value in memory using the string '000025'.



The following figure is an example of the RstSeq pre-compiler command that updates the current source line cursor value in memory using the current variable value from MySavedSourceSequence.



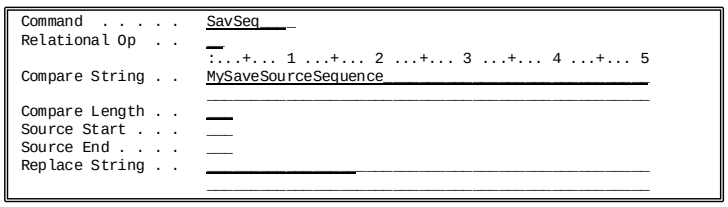
SavSeq

Command	Relational Op	Compare String	Compare Length	Source Start	Source End	Replace String
SavSeq		Variable Name				

The SavSeq pre-compiler command updates the variable value with the current source line cursor in memory.

Variable Name must be specified and contain a valid integer value in the format 999999 including all leading zeros.

The following figure is an example of the SavSeq pre-compiler command that updates current variable value of MySavedSourceSequence with the current source line cursor value in memory.



Scan

Command	Relational Op	Compare String	Compare Length	Source Start	Source End	Replace String
Scan	Relational Op	Compare String	Compare Length	Source Start	Source End	Replace String

The Scan pre-compiler command searches for a matching source line starting with the next source line after the current source line cursor.

A comparison is performed with each source line in turn until it matches the comparison criteria specified. If the comparison fails to find a source line before end of file, then return code is set to 'Y2U0022' and must be handled using an error handling pre-compiler command. Failure to monitor for the return code will cause the pre-compiler group to exit abnormally with return code 'Y2U0022'.

Relational Op must be specified as either EQ or NE, and Compare String must also be specified. Each source line from the next one based on the current source line cursor from the specified Source Start to Source End will be scanned and compared to the Compare String for the specified Compare Length. The Relational Op of EQ will perform an equality comparison and NE will perform an inequality comparison.

If a Replace String is specified and the Relational Op is specified as EQ, then that string will be inserted into the current source line in memory from the start position of the Compare String for the specified Compare Length prior to the source line being updated.

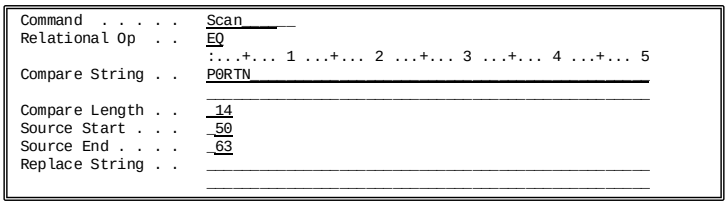
If a Replace String is specified and the Relational Op is specified as NE, then that string will be inserted into the current source line in memory from the specified Source Start for the specified Compare Length prior to the source line being updated.

If Compare Length is not specified, then the length of the search argument is defaulted to the length of the Compare String excluding trailing blanks.

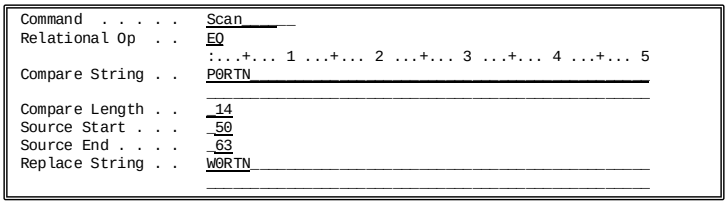
If Source Start is not specified, then the scan starts at the beginning of each source line.

If Source End is not specified, then the scan completes at the end of each source line.

The following figure is an example of the Scan pre-compiler command that searches each source line between column 50 and 63 for an equal comparison with string 'P0RTNbbbbbbbbb'.



The following figure is an example of the Scan pre-compiler command that searches each source line between column 50 and 63for an equal comparison with string 'P0RTNbbbbbbbbb'. If it is found, then the source line is updated with string 'W0RTNbbbbbbbbb'.



SubString

Command	Relational Op	Compare String	Compare Length	Source Start	Source End	Replace String
SubString		Variable Name	Source Length	Source Start		

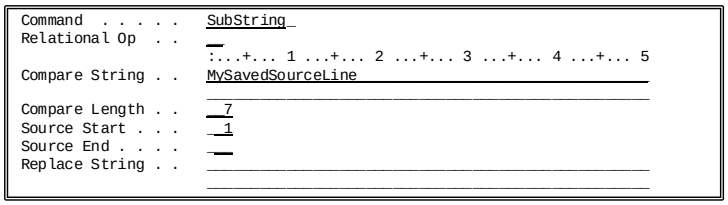
The SubString pre-compiler command updates the current variable value for the specified Variable Name.

Variable Name must be specified and the current variable value will be updated with the current source line value in memory from the specified Source Start for Source Length.

If Source Length is not specified, then the length is defaulted to the length of the current source line in memory from the Source Start excluding trailing blanks.

If Source Start is not specified, then the Source Start is defaulted to the beginning of the current source line value.

The following figure is an example of the SubString pre-compiler command that updates the current variable value for the 'MySavedSourceLine' variable with the current source line value in memory between column 1 and 7.



Update

Command	Relational Op	Compare String	Compare Length	Source Start	Source End	Replace String
Update	Relational Op or Pre-Built Ind	Compare String	Compare Length	Source Start	Source End	Source String

The Update pre-compiler command updates the current source line based on the source line cursor.

A source line positioning pre-compiler command must be used before the use of the Update pre-compiler command in order to initialise the source line cursor.

An optional comparison can be performed to only update the source line if it matches the comparison criteria specified. If the comparison fails, then return code is set to 'Y2U0022' and must be handled using an error handling pre-compiler command. Failure to monitor for the return code will cause the pre-compiler group to exit abnormally with return code 'Y2U0022'.

If Relational Op is either EQ or NE, then the current source line based on the source line cursor from the specified Source Start to Source End will be scanned and compared to the Compare String for the specified Compare Length. A Relational Op of EQ will perform an equality comparison and NE will perform an inequality comparison.

The Pre-Built Ind is specified as '**' and indicates that the current source line value in memory is to be used for the current source line to be updated with. The current source line value in memory may have been built by the InsString pre-compiler command prior to processing the Update pre-compiler command.

If the Pre-Built Ind is specified, then Compare String, Compare Length, Source Start, Source End and Source String are not allowed.

If the Pre-Built Ind is not specified, then Source String must be specified.

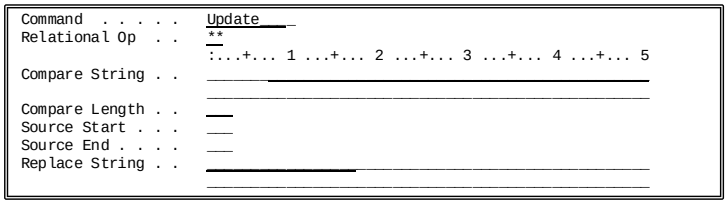
If Compare Length is not specified, then the length of the search argument is defaulted to the length of the Compare String excluding trailing blanks.

If Source Start is not specified, then the scan starts at the beginning of the current source line.

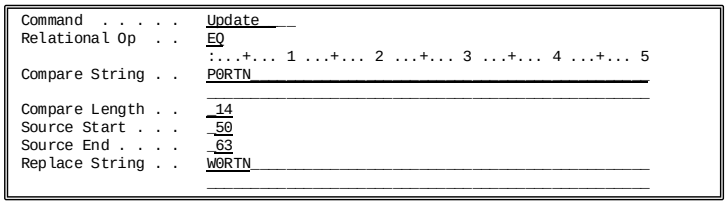
If Source End is not specified, then the scan completes at the end of the current source line.

If Source String is specified, then that string will be inserted into the current source line value in memory from the specified Source Start for Compare Length prior to the source line being updated.

The following figure is an example of the Update pre-compiler command that updates the current source line from the current source line value in memory.

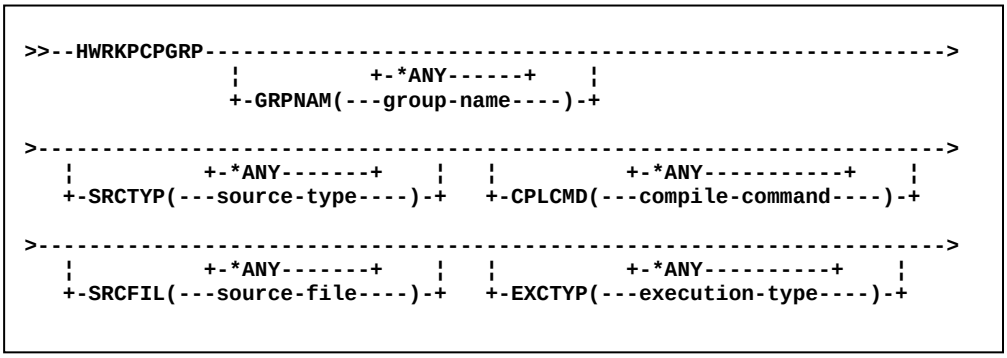


The following figure is an example of the Update pre-compiler command that updates the current source line with 'W0RTNbbbbbbbbb' if 'P0RTNbbbbbbbbb' is found between column 50 and 63.



Command Reference

Work with Pre-Compiler Groups (HWRKPCPGRP)



Purpose

The Work with Pre-Compiler Groups (HWRKPCPGRP) command initiates a editor session for the Freedom/Pre-Compiler groups and associated commands.

Group name (GRPNAM)

Specifies the name or partial name of the pre-compiler group that the initial list is to display.

*ANY

Indicates that the initial list of pre-compiler groups will include all compiler groups.

Source type (SRCTYP)

Specifies the name of the source file types that the initial list is to display.

*ANY

Indicates that the initial list of pre-compiler groups will include all source types.

Compile command (CPLCMD)

Specifies the name of the compile command that the initial list is to display.

*ANY

Indicates that the initial list of pre-compiler groups will include all compiler commands.

Source file (SRCFIL)

Specifies the name of the source file that the initial list is to display.

*ANY

Indicates that the initial list of pre-compiler groups will include all source files.

Execution type (EXCTYP)

Specifies the name of the execution type that the initial list is to display.

*ANY

Indicates that the initial list of pre-compiler groups will include all execution types.

*PREPGM

Indicates that the initial list of pre-compiler groups will include only pre-compile exit program groups.

*PREDIR

Indicates that the initial list of pre-compiler groups will include only pre-compile directive groups.

*POSTDIR

Indicates that the initial list of pre-compiler groups will include only post-compile directive groups.

*POSTPGM

Indicates that the initial list of pre-compiler groups will include only post-compile exit program groups.

**** END OF DOCUMENT ****