isCOBOL Evolve: Transitioning Guides

Key Topics:

- Transitioning from ACUCOBOL-GT
- Transitioning from MicroFocus
- Transitioning from RM/COBOL

Table of Contents

1. Transitioning from ACUCOBOL-GT 1

Compiler options 1
Copybooks 5
Configuration variables 6
Data acces. . . . . .s 7
Syntax and Behavior 8
User interface 11
Library routines. 14
Web 19

2. Transitioning from MicroFocus 20

Compiler options 20
Data access. 22
Syntax and Behavior. 22
Configuration 24
Library routines 24
Intrinsic Functions 26

3. Transitioning from RM/COBOL 27

Compiler options 27
Run time and configuration 27
Data access. 28
Syntax and Behavior 28
Library routines 29

4. Transitioning from ICOBOL 33

Compiler options 33
Run time and configuration 33
Data access 33
Library routines 33

5. Transitioning from Microsoft COBOL 34

Compiler options 34
Run time and configuration 34
Data access. 34

6. Transitioning from MBP COBOL 35

Compiler options 35
Data access. 35

7. Transitioning from NCR COBOL 36

Compiler options 36
Data access 36

Transitioning from ACUCOBOL-GT

isCOBOL Evolve is highly compatible with ACUCOBOL-GT syntax and features.

The following sections describe the key areas which need to be addressed during the transition.

Compiler options

- When compiling source files of an ACUCOBOL-GT application, the following options should always be used:

```
-ca -cnlz -smat -wlu
```

- If the ACUCOBOL-GT version is 7.0 or later, the following option should also be used:

```
-dvexta=32
```

- If you didn't use -Cf option with the ACUCOBOL-GT compiler, then the following option should be used with the isCOBOL compiler:

```
-crv
```

- If you didn't use -Ck option with the ACUCOBOL-GT compiler, then the following option should be used with the isCOBOL compiler:

```
-cko
```

- If you didn't use -Za option with the ACUCOBOL-GT compiler, then the following option should be used with the isCOBOL compiler:

```
-m1
```

- If you didn't use -Zz option with the ACUCOBOL-GT compiler, then the following option should be used with the isCOBOL compiler:

```
-cudc
```

- If your application uses characters that take more than one byte in memory (e.g. Chinese characters) and you want string operations such as INSPECT, STRING and UNSTRING to work byte-wise rather than character-wise, then the following option should be used as well:

```
-b
```

- If you have level 78 items defined just below group items to measure their size, then the following option should be used:

```
-s78c
```

- If you have $ signs in COPY statements in order to reference environment variables in the copybook path, then the following option should be used:

```
-sevc
```

This is a typical case where *-s78c* is necessary:

```
01 group1.
   03 data-item1 pic x(10).
   03 data-item2 pic 9(3).
78 group1-sz      value length of group1.
```

- Some ACUCOBOL-GT options are available in isCOBOL with the same name and produce the same effect. Some others are not available in isCOBOL, but their effect can be obtained anyway. The following table lists the known ACUCOBOL-GT options along with their equivalent in the isCOBOL environment, if available:

ACUCOBOL-GT option	**isCOBOL solution**
--noTrunc	Use -dznt
--truncANSI	Use -dzta
-a	Default behavior using isCOBOL
-Ca	Use -vansi
-Cb	Default behavior using isCOBOL
-Ce	The same option is available in isCOBOL. Use -ce=*extensions* compiler option
-Ci	Use -ci -dcd. Add iscobol.gui.screen_col_plus_base=0 to the configuration
-Ck	Default behavior using isCOBOL
-Cm	Not available in isCOBOL. isCOBOL has an option named -cm, but it's for other purposes
-Cp	Default behavior using isCOBOL
-Cr	Use -crlk and -va compiler options. Set iscobol.gui.screen_col_zero=true in the configuration.
-Cv	The same option is available in isCOBOL. Use -cv compiler option
-D1	The same option is available in isCOBOL. Use -d1 compiler option
-D5	The same option is available in isCOBOL. Use -d5 compiler option
-D6	Add the following entry to the configuration file: *iscobol.compiler.regexp="(?i)(pic\\s+9\\([0-9]+\\)\\s+)(packed-decimal)" "$1 comp-6"*
-Da#	Use -align=#

ACUCOBOL-GT option	isCOBOL solution
-Dca	The same option is available in isCOBOL. Use -dca compiler option
-Dcb	The same option is available in isCOBOL. Use -dcb compiler option
-Dci	The same option is available in isCOBOL. Use -dci compiler option
-Dcm	The same option is available in isCOBOL. Use -dcm compiler option
-Dcn	The same option is available in isCOBOL. Use -dcn compiler option
-Dcr	The same option is available in isCOBOL. Use -dcr compiler option
-Dd31	Default behavior using isCOBOL. In order to have the same precision on the last decimal digits, use -cfp36 compiler option
-Df	Use -cv compiler option
-Di	Use -dia compiler option
-Dm	Use -dcdm compiler option
-Ds	The same option is available in isCOBOL. Use -ds compiler option
-Dv=#	Use -dv=# and -dvexta=# compiler options
-Dw32	Default behavior using isCOBOL.
-Dw64	Use -d64 compiler option. Note that only USAGE POINTER data items are affected.
-Dz	The same option is available in isCOBOL. Use -dz compiler option
-e	Use -eo=*folder* compiler option. isCOBOL allows to specify the destination folder for error files, but not the file name. Files are always named *ProgramName*.err
-Fa	Use -efa compiler option. isCOBOL generates dictionaries in XML format. Acucobol-GT's XFD and isCOBOL XML are not compatible
-Fo	Use -efo=*folder* compiler option
-Fx	Use -efd compiler option. isCOBOL generates dictionaries in XML format. Acucobol-GT's XFD and isCOBOL XML are not compatible

ACUCOBOL-GT option	isCOBOL solution
-Ga	Use either -d or -dx compiler options
-Lf	The same option is available in isCOBOL, but isCOBOL list files are not in ANSI format by default. Use -lf and -la compiler options combined
-Lo	Use -lo=*folder* compiler option. isCOBOL allows to specify the destination folder for list files, but not the file name. Files are always named *ProgramName*.list
-Ls	Use -ld compiler option
-o	Use -od=*folder* compiler option. isCOBOL allows to specify the destination folder for object files, but not the file name. Files are always named *PROGRAMNAME*.class
-Rc	The same option is available in isCOBOL. Use -rc=*word1,word2* compiler option
-Rw	The same option is available in isCOBOL, but the usage is slightly different. In isCOBOL the option can't be repeated, so all the reserved words must be specified at once separated by comma. E.g. the following Acu options: -rw *word1* -rw *word2* are translated to isCOBOL as follows: -rw=*word1,word2*
-S# (where # is the number of columns for each tab)	Use -stl=# compiler option
-Sa	The same option is available in isCOBOL. Use -sa compiler option and avoid using -smat
-Si	Add IF Directive to the source
-Sl	The same option is available in isCOBOL. Use -sl compiler option.
-Sp	The same option is available in isCOBOL. Use -sp=*folders* compiler option
-Sr	Use -stl=8,4
-St	The same option is available in isCOBOL. Use -st compiler option and avoid using -smat
-Sx	Add IF Directive to the source
-v	The same option is available in isCOBOL. Use -v compiler option
-Va	Not available in isCOBOL. isCOBOL has an option named -va, but it's for other purposes
-Vc	Default behavior using isCOBOL

ACUCOBOL-GT option	isCOBOL solution
-Vh	The same option is available in isCOBOL. Use -vh compiler option
-Vl	Default behavior using isCOBOL, unless -vh is used
-Vu	The same option is available in isCOBOL. Use -vu compiler option
-Vx	The same option is available in isCOBOL. Use -vx compiler option
-Za	Set iscobol.array_check=1 and iscobol.substring_check=1 in the configuration
-Zd	Use either -d or -dx compiler options
-Zi	The same option is available in isCOBOL. Use -zi compiler option
-Zo	Use -efo=*folder* compiler option
-Zr#	Use either -pt0, pt1 or -pt2 compiler options according to your needs
-Zx	Use -efd compiler option. isCOBOL generates dictionaries in XML format. Acucobol-GT's XFD and isCOBOL XML are not compatible
-Zy	The same option is available in isCOBOL. Use -zy compiler option
-Zz	Default behavior in isCOBOL

The latest ACUCOBOL-GT compilers support conditional compilation through the use of special constructs in the COBOL source file and by accepting command-line arguments that turn on compiler directives and set constants to values. This feature is implemented in compatibility with Micro Focus. Refer to Compiler options in the *Transitioning from Micro Focus* guide for details about how to obtain the same result with isCOBOL.

Copybooks

In order to experience correct behaviors at run time it's suggested to replace ACUCOBOL-GT copybooks by the equivalent isCOBOL copybooks, if available. Although their content is very similar there are some small differences that might compromise the behavior of IIO, WINPRINTER, ESQL and the ACCEPT *system-information* FROM SYSTEM INFO statement. Veryant encourages the use of isCOBOL copybooks in order to take advantage of those features that are available only in isCOBOL.

The table below lists the Acucobol-GT copybooks that can be replaced by an equivalent isCOBOL copybook:

ACUCOBOL-GT copybook(s)	isCOBOL copybook(s)
acucobol.def	iscobol.def

ACUCOBOL-GT copybook(s)	isCOBOL copybook(s)
acugui.def	isgui.def isreg.def
controls.def	iscontrols.def
crtvars.def	iscrt.def
filesys.def	isfilesys.def
fonts.def	fonts.def
lmresize.def	isresize.def
opensave.def	isopensave.def
palette.def	ispalette.def
socket.def	issocket.def
stdfonts.def	stdfonts.def
winhelp.def	iswinhelp.def
winprint.def	isprint.def
winvers.def	iswinvers.def
SQLCA	SQLCA (same name but quite different content)

Configuration variables

- Most of the ACUCOBOL-GT configuration variables have an equivalent configuration property in isCOBOL. The easiest way to convert the ACUCOBOL-GT configuration file to the equivalent isCOBOL property file is by using the ISCONFIG utility.
- In order to have Acucobol-GT behavior that are not default in isCOBOL, the following boolean properties should be set in the isCOBOL configuration:

```
iscobol.ccopy.client_temp_as_base_dir=true
iscobol.file.env_naming=true
iscobol.file.index.check_all_keys=true
iscobol.key.default_shortcuts_enabled=false
iscobol.memory.alpha_edited=true
iscobol.properties.acu_compat=true
iscobol.terminal.autowrap=true
```

In order to have an ACUCOBOL-GT compatible behavior on sequential files, set:

```
iscobol.file.linesequential=lseqacu
iscobol.file.sequential=seqacu
```

If the programs have GUI, the following settings should be used as well:

```
iscobol.gui.grid.lm_on_columns=false
iscobol.gui.ignore_invalid_handle=true
iscobol.gui.push_override_focus_change=false
```

If you display 3-D buttons over tool-bars and you're not interested in having the ability to detach the tool-bar from the window, then you should also set:

```
iscobol.gui.tool_bar.native=false
```

In order to have the same cell size as ACUCOBOL-GT when using internal fonts, set:

```
iscobol.font.default=Microsoft Sans Serif-Bold-08
iscobol.font.medium=Microsoft Sans Serif-Bold-08
iscobol.font.large=Microsoft Sans Serif-Bold-10
iscobol.font.small=Microsoft Sans Serif-08
iscobol.font.traditional=Fixedsys Excelsior 2.00-11
iscobol.font.fixed=Fixedsys Excelsior 2.00-11
```

Note that the Fixedsys Excelsior font is usually not installed by default in the operating system. The TTF file is provided along with isCOBOL so you can install the font.
The ISCONFIG utility takes care of the above settings.

- If you ACCEPT FROM ENVIRONMENT a variable that was not set, ACUCOBOL-GT returns the variable default value, isCOBOL returns spaces or zero (depending on the destination item picture) instead.
- DLL_CONVENTION is available in isCOBOL but does not support alphabetic values like "WINAPI". Use numeric values.
- The KEYSTROKE feature lacks of the ability to make special and function keys (like F1 and Enter) send characters to the program. They can only produce exceptions. For example, this setting is valid for isCOBOL:

```
KEYSTROKE Exception=100 ^M
```

While this one is not:

```
KEYSTROKE Data=65 ^M
```

- MOUSE_FLAGS configuration variable is not supported by isCOBOL. isCOBOL allows to set mouse flags in the ACCEPT Statement. So, for example, the following code:

```
SET ENVIRONMENT "MOUSE-FLAGS" TO "2". |enables left button pressed action
ACCEPT Screen1.
```

Must be changed as follows:

```
ACCEPT Screen1 MOUSE FLAGS 2.
```

- The same feature as DECIMAL-POINT configuration variable is provided through the compiler option -sddp.
- COLOR_MAP HIGH, LOW and DEFAULT can be remapped using iscobol.colormap.high *, iscobol.colormap.low * and iscobol.colormap.default * configuration properties. Other COLOR_MAP settings are not supported.
- When a file is created by OPEN I-O having IO-CREATES set to 1, ACUCOBOL-GT sets the file status to "00", isCOBOL sets the file status to "05", instead.
- When a file is created by OPEN EXTEND having EXTEND-CREATES set to 1, ACUCOBOL-GT sets the file status to "00", isCOBOL sets the file status to "05", instead.
- The CODE_CASE feature can be obtained at compile time by using the -ssnl and -ssnu options.
- The hotkey configuration is slightly different in isCOBOL. An entry like the following one:

```
 Hot-Key  MYPROG  = 201
```

need to be translated to:

```
iscobol.hot_key.MYPROG=201
```

- The WAIT_FOR_LOCKS configuration variable could be set to 1 meaning "wait for the locked record if no Declarative is available for the file, otherwise return error". The same behavior can be obtained in isCOBOL by compiling programs with the -crlk option.
- The isCOBOL equivalent of A_CHECKDIV is the property iscobol.checkdiv.
- The isCOBOL equivalent of WINDOW_TITLE is the property iscobol.gui.window_title.
- The isCOBOL equivalent of ICON is the property iscobol.gui.icon_file.
- The WIN3_GRID setting supports less values in isCOBOL. You can specify only high intensity colors using values from 1 to 7.
- In isCOBOL, setting a keystroke to a different function doesn't reset the current setting, but appends information to it. For example, having the following setting in the environment:

```
KEYSTROKE Exception=100 k1
```

If you perform:

```
SET ENVIRONMENT "KEYSTROKE" TO "Edit=next"
```

In ACUCOBOL-GT you obtain:

```
KEYSTROKE Edit=Next k1
```

In isCOBOL you obtain:

```
KEYSTROKE Edit=Next Exception=100 k1
```

This rule applies also to the *iscobol.key* properties.

- The ACUCONNECT_RUNTIME_FLAGS configuration variable doesn't have an exact equivalent in isCOBOL, as isCOBOL doesn't allow to set runtime options for remote programs. However, isCOBOL allows to specify a configuration file for remote programs, so, if you used ACUCONNECT_RUNTIME_FLAGS to specify just the -c option, you can replace:

```
ACUCONNECT_RUNTIME_FLAGS=-c /path/to/remote_configuration_file
```

with

```
iscobol.remote_conf=/path/to/remote_configuration_file
```

- Command line switches are passed through the iscobol.switches * configuration property in isCOBOL. For example, the following two commands:

```
wrun32 -1 prog1
wrun32 -12 prog1
```

can be translated as follows:

```
iscrun -J-Discobol.switches=1 PROG1
iscrun -J-Discobol.switches=1,2 PROG1
```

Data access

- If the ACUCOBOL-GT application worked on databases using the Acu4GL interface, consider that the same feature can be obtained with isCOBOL by taking advantage of isCOBOL Database Bridge. The -ca option should be used when launching the EDBI routine generator (edbiis).
- Veryant does not provide direct support on the Vision file system, but you're allowed to convert your indexed files to a supported file system (JIsam or c-tree) easily with the ISMIGRATE (Index File Migration) utility. Choose "com.iscobol.io.ScanVision" as input file handler to let ISMIGRATE read Vision files without the need of external components.

  If you want to maintain Vision as file handler, Veryant can provide a special native library that is obtained by recompiling your ACUCOBOL-GT runtime. For the Linux/Unix platform a File Connector solution is also available.
  See The Vision File Connector for details about the File Connector solution. This solution is suggested unless you're working on Windows, in that case contact Veryant for more information on the special native library.

Be aware that every solution has dependences to the ACUCOBOL-GT runtime libraries and needs a valid license in order to work.

- Fixed length sequential and relative files as well as variable length sequential files are compatible between ACUCOBOL-GT and isCOBOL assuming that you set the following variables in the configuration:

```
iscobol.file.linesequential=lseqacu
iscobol.file.sequential=seqacu
```

- Relative files with variable record length are not supported.
  They must be converted to fixed length relative files by an ACUCOBOL-GT program that reads from the variable length file and writes to a fixed length file.

- By default, isCOBOL returns ANSI 2002 file status codes, while ACUCOBOL-GT returns ANSI 85 file status codes. You should set the iscobol.file.status * property according with the following table:

value of FILE_STATUS_CODES in ACUCOBOL-GT configuration	value of iscobol.file.status in isCOBOL configuration
not set	com.iscobol.io.FileStatus85
74	com.iscobol.io.FileStatus74
85	com.iscobol.io.FileStatus85
DG	com.iscobol.io.FileStatusDG
VAX	com.iscobol.io.FileStatusVax
IBM	com.iscobol.io.FileStatusIBM

- Errors on DELETE FILE cannot be intercepted and handled by the COBOL program through Declaratives. Unless iscobol.file.errors_ok is set to true, they cause the Framework to raise a Java exception.

- The file status "30" is returned when you you open a file whose path doesn't exist and when you try to create a file where there are no write permissions. ACUCOBOL-GT returns "35" and "37" respectively.

- The syntax “@server:filepath” in file names and file-prefix is not supported by isCOBOL. See isCOBOL File Server for information about how to replace Acuserver with the isCOBOL File Server.

Syntax and Behavior

- The PROGRAM-ID paragraph must be explicitly declared, and the name of the program must match the name of the source file.
- If a paragraph is declared, then its parent section must be declared as well. For example, INPUT-OUTPUT SECTION must be declared above FILE-CONTROL.

- Only numeric data items can be used to receive the special register CRT STATUS.
- USAGE POINTER data items are translated into numeric items and cannot be passed to external C language functions, unless the “-cp” compiler option is used. If “-cp” is used, pointers are natively handled and can be passed to external C language functions.

- The LOCK THREAD and UNLOCK THREAD statements are not supported. Synchronization can be achieved using the SYNCHRONIZED statement.
- The CANCEL SORT instruction is not supported.
- A Picture description cannot be distributed on multiple lines. The following code would not compile:

```
       01 COM-02                  PIC XBXBXBXBXBXBXBXBXBXBXBXBXBX
      -    BXBXBXBXBXBXBXBXBXBXBXBXBXBXBXBX.
```

- OCCURS indexes are checked only at runtime, not at compile-time.
- If the JUSTIFIED clause is declared for a control, it must appear immediately after the PICTURE clause.
- It's not possible to ACCEPT or DISPLAY a control that has subordinate controls directly below it. Consider the following screen, for example:

```
01  SCREEN.
    03 FR FRAME.
       05 EF ENTRY-FIELD.
```

  You can ACCEPT and DISPLAY EF, but you can't ACCEPT or DISPLAY FR. In order to ACCEPT or DISPLAY FR, you should define it at the same level of EF.

- The "RECEIVE *MyMessage* FROM ANY THREAD" statement must include the NO WAIT statement, or its equivalent, BEFORE TIME 0. Any other timeout is not supported. For example, the following code below will not compile:

```
RECEIVE MyMessage FROM ANY THREAD
RECEIVE MyMessage FROM ANY THREAD BEFORE TIME 1
```

  It must be changed to:

```
RECEIVE MyMessage FROM ANY THREAD NO WAIT
RECEIVE MyMessage FROM ANY THREAD BEFORE TIME 0
```

- ActiveX support is not provided. Third-party tools must be used. Contact your Veryant representative for more information.
- Data items defined as USAGE UNSIGNED-LONG or USAGE SIGNED-LONG are 8 bytes long. To declare a 4-byte native data item, it must be PIC 9(9) COMP-5 or PIC S9(9) COMP-5.
- If used, the ENTRY Statement must be declared at the beginning of the paragraph.
- Items defined in the LINKAGE section must be referenced also in the USING phrase. In ACUCOBOL-GT it is not mandatory. You can rely on the -wlu compiler option to find out this kind of issue in the source files.
- Subordinate items of a group item in the LINKAGE section cannot be referenced in the USING phrase. The entire group must be referenced. The following syntax is not supported by isCOBOL:

```
LINKAGE SECTION.
01 GROUP-ITEM.
   03 ITEM-1 PIC XX.
   03 ITEM-2 PIC X(10).

PROCEDURE DIVISION USING ITEM-2.
```

The following is supported instead:

```
LINKAGE SECTION.
01 GROUP-ITEM.
   03 ITEM-1 PIC XX.
   03 ITEM-2 PIC X(10).

PROCEDURE DIVISION USING GROUP-ITEM.
```

- The display format of the DATE-ENTRY control does not depend on settings specified by the user in the Windows Control Panel. The following styles use European conventions for time and date formatting:

```
TIME
CENTURY-DATE
LONG-DATE
```

For time and date formatting conforming to American conventions, use the following properties:

```
DISPLAY-FORMAT "h:mm a"
DISPLAY-FORMAT "MM/dd/yyyy"
DISPLAY-FORMAT "EEEE, MMMM dd, yyyy"
```

If your code does not currently specify any of the above styles and does not specify the DISPLAY-FORMAT property, use DISPLAY-FORMAT "M/d/yyyy". If no display format styles or properties are specified, then DISPLAY-FORMAT "dd/MM/yyyy"is the default.

- A window handle defined as PIC X(10) cannot be redefined by a USAGE HANDLE OF WINDOW item. However isCOBOL allows MODIFY on PIC X(10) as well as on USAGE HANDLE items.
- isCOBOL Reserved Words cannot be used as variable or constant names by default. In order to use them as variable or constant names, they must be treated as common words by using -rc compiler option.
- ACCEPT SYSTEM-INFORMATION FROM SYSTEM-INFO returns the complete name of the Operating System.
- File-Status codes are ANSI-2002 compliant by default. To use ANSI-85 codes instead, add the following line to the isCOBOL properties file:

```
iscobol.file.status=com.iscobol.io.FileStatus85
```

- Environment variables that belong to the system environment can be retrieved only when using a Java Virtual Machine version 1.5 or above.
- The CALL RUN statement creates a new thread. Environment variables and external variables are initialized to their default value.

- The STOP THREAD statement requires a YIELD statement to work correctly, otherwise, the thread will not be released. For example:

```
           perform thread show-time handle t1.
           ...
           STOP THREAD t1.
           ...
           perform thread show-time handle t1.
           ...
       show-time.
           perform until 1 = 2
             accept w-time from time
             display w-time line 2 pos 70
             YIELD
           end-perform.
```

- If your application has recursive programs, and you don't want the data shared between different copies of the same program, add the following line to your configuration file:

```
iscobol.recursion_data_global=False
```

- The FOR ANY THREAD clause in WAIT statement is not supported.
- The syntax "-D" at the beginning of a physical file name is not supported.
- The syntax "-Q" at the beginning of a physical file name doesn't support CHARSET and COLS options.
- In ACUCOBOL-GT USAGE HANDLE items are automatically synchronized to offsets that are a multiple of 4. It doesn't happen in isCOBOL. This difference might cause problems if USAGE HANDLE items are part of group items like for example record definitions, as it changes the size of the group item. In order to have USAGE HANDLE synchornized, you should change

```
01 my-group.
   ...
   05 my-item USAGE HANDLE.
   ...
```

to

```
01 my-group.
   ...
   05 my-item PIC 9(9) COMP-5 SYNC.
   ...
```

- Copybooks whose content is not terminated by a dot are not allowed. There are two solutions:
  - add a dot after the last line of the copybook content, or
  - use two dots to close the COPY statement (e.g. `COPY "mycopy.prd"..`)
- The DIVIDE statement cannot have REMAINDER without GIVING. The following statement is compiled by ACUCOBOL-GT:

```
DIVIDE ITEM1 INTO ITEM2 REMAINDER REM.
```

In order to be compiled by isCOBOL, it must be changed to:

```
DIVIDE ITEM1 INTO ITEM2 GIVING FOO REMAINDER REM.
```

Or you can use the REM intrinsic function:

```
MOVE FUNCTION REM(ITEM1, ITEM2) TO REM.
```

- When a numeric data-item is used as configuration variable value in a SET ENVIRONMENT statement, Acucobol stores leading zeroes as well, isCOBOL instead stores only the numeric value, unless you compile with the -cdlz option.
- MOVE CORRESPONDING can have only one receiver data item in isCOBOL.
- The help cursor mode in help automation is not supported.
- The statement WRITE print-record AFTER ADVANCING 0 LINES doesn't reset the caret position in isCOBOL, hence the new text is appended to the previous text instead of overwriting it.

User interface

- All the windows have a visible System-Menu, even if the SYSTEM-MENU style is not explicitly set.
- Windows are always link-to-thread. Therefore, they cannot be programmatically activated in response to a CMD-ACTIVATE Event.
- The scroll-bar control raises only the MSG-SB-THUMB event.
- Check-Boxes with the LEFT-TEXT style show the title on the left of the box, however the text is not left justified. In order to have the text left justified like it happens with Acucobol, the LEFT-TEXT-ALIGNMENT property must be set to 1. This property can be easily added at compile time by adding the following entry to the Compiler configuration:

```
iscobol.compiler.gui.check_box.defaults=left-text-alignment 1
```

- Performing DISPLAY WINDOW ERASE on a graphical window may not clean the graphical window completely. In some operating systems, the pixels of the graphical window border are not affected by the statement.
- The coordinates of pop-up menus are relative to the upper left corner of the current window, not to the upper left corner of the physical screen.

- The BUTTONS, LINES-AT-ROOT and SHOW-LINES styles of TREE-VIEW behaves slightly different in isCOBOL. Consult the User Interface Manual for the description of their behavior.
- Modifying the EXPAND property of TREE-VIEW control does not generate events until ACCEPT is performed.
- When you collapse the parent of the current TREE-VIEW item, causing the selection to change, events are generated in a different order. In ACUCOBOL-GT you have MSG-TV-SELCHANGE first and then MSG-TV-EXPANDED. In isCOBOL you have MSG-TV-EXPANDED first and then MSG-TV-SELCHANGE.
- The INQUIRE and MODIFY statements work only with valid handles. Use the FUNCTION HANDLE-TYPE to ensure that the handle is valid before using it. Alternatively, to avoid errors due to invalid handles, add the following line to your configuration file:

```
iscobol.gui.ignore_invalid_handle=True
```

- The "INQUIRE *WindowHandle*, SYSTEM-HANDLE IN *SystemHandle*" syntax is not supported. This is a Java restriction. The system handle of the active window can be retrieved only with the native GetActiveWindow and GetForegroundWindow APIs.
- The following WEB-BROWSER properties are not supported by isCOBOL: CLEAR-SELECTION, COPY-SELECTION, CUSTOM-PRINT-TEMPLATE, PAGE-SETUP, PRINT-PREVIEW, PROPERTIES, SELECT-ALL and TYPE.
- Modal floating windows don't allow to switch to any other active window while they have the focus. ACUCOBOL-GT allowed to switch to a child independent window, instead.
- Resizable floating windows cannot be maximized.
- The following TAB-CONTROL styles don't have any effect:
  - FIXED-WIDTH
  - HOT-TRACK
- GRID doesn't support columns distributed on multiple rows, therefore a similar setting:

```
DISPLAY-COLUMNS (1, 5, 10, 1, 8, 12)
```

  is treated to:

```
DISPLAY-COLUMNS (1, 5, 10, 13, 20, 24)
```

- In isCOBOL pressing the DEL key when the focus is on a Grid fires a MSG-BEGIN-ENTRY event even if a keystroke is associated to the DEL key. In Acucobol instead, if a keystroke is associated to the DEL key, then the Accept is interrupted and the CRT STATUS is set to the exception value of the DEL key. In isCOBOL you can intercept the DEL key pressure by inquiring the ENTRY-REASON property during the MSG-BEGIN-ENTRY event: if DEL was pressed, then the ENTRY-REASON value is X"01".
- Due to Java implementation, the styles CENTER(ED) and RIGHT don't have effect on MULTILINE ENTRY-FIELD.
- If the -ci flag is used, the ACUCOBOL-GT compiler treats "COL + 0" as "COL + 1" unless a specific configuration variable is set to change this behavior. In order to obtain the same behavior with isCOBOL, set iscobol.gui.screen_col_plus_base to 0 in the configuration.

- In the After Procedure of an input control is possible to force the focus to stay in the control by setting Screen-Control items properly. However, if the user leaves the input field by clicking on a button control (Push-Button, Radio-Button or Check-Box) isCOBOL executes the button event procedure. This incompatible behavior can be avoided only for Push-Buttons by setting iscobol.gui.push_override_focus_change (boolean) to false.
- When the focus returns on a ENTRY-FIELD after a message-box or another modal window is closed, the text inside the ENTRY-FIELD is never selected. This behavior can be corrected by setting the Cursor property to the value -1.
- Modifying CELL-COLOR of a GRID cell that hasn't been created yet has no effect.
- Destroying a Screen Section control removes it permanently from the Accept of the Screen, even if you soon re-display it. The following snippet reproduces the case:

```
SCREEN SECTION.
01  SCREEN1.
    03 ENTRY-FIELD
       LINE 2 COL 2, AFTER EF-AFTER
       .
    03 ENTRY-FIELD
       LINE 4 COL 2
       .
    03 COMBO1.
       05 COMBO-BOX DROP-LIST   item-to-add ("1", "2", "3")
          LINE 6, COL 2,  LINES 10, SIZE 20
          .
    03 ENTRY-FIELD
       LINE 8 COL 2
       .

PROCEDURE DIVISION.
MAIN.
    DISPLAY STANDARD GRAPHICAL WINDOW.
    DISPLAY SCREEN1.
    ACCEPT  SCREEN1 UNTIL CRT-STATUS = 27 ON EXCEPTION CONTINUE.
    STOP    RUN.

EF-AFTER.
    DESTROY COMBO1.
    DISPLAY COMBO1.
```

  With ACUCOBOL-GT you're able to move the focus on the combo-box, with isCOBOL you're not.
- The background intensity can't be changed dynamically at run time by setting the BACKGROUND-INTENSITY configuration property with the SET ENVIRONMENT statement.
- In ACUCOBOL-GT the FRAME FILL-COLOR was affected by the control's background intensity. In isCOBOL it's not affected.
- In ACUCOBOL-GT, when you select a RADIO-BUTTON using the keyboard, it's automatically checked. In isCOBOL you have also to press the SPACE-BAR in order to check it.
- The NTF-RESIZED event is always returned in the same way when the user drags the window border with the mouse. In ACUCOBOL-GT it's returned in two different ways depending on where it's intercepted: if intercepted in Event Procedure, many events are returned while the user drags the mouse; if intercepted as exception on the ACCEPT of the Screen, only one event is returned when the user releases the mouse. isCOBOL always returns many events while the user drags the mouse. You can reduce the number of

events by setting iscobol.gui.ntf_resized_delay in the configuration. Alternatively you can configure the operating system to have only one event at mouse release (for example, in Windows, uncheck the option "Show window content while dragging" under "Visual Effects" settings).

- The statement `SET generic-handle TO HANDLE OF screen-name` is always successful even if the control identified by *screen-name* doesn't exist. In order to check if a control exists, you can rely on a Format 2 INQUIRE statement and query the STATUS property.
- In isCOBOL it's not possible to modify the ACTION property of a GRID within MSG events handling. The error "Action in event procedure" is raised in these cases. ACUCOBOL-GT allowed this kind of operation instead.
- The TERMINAL-ABILITIES structure set by ACCEPT FROM TERMINAL-INFO has the following differences between Acu and isCOBOL:
  - The TERMINAL-NAME field is set to "Windows" by Acu while isCOBOL sets it to "xterm";
  - The CLIENT-MACHINE-NAME field includes the process ID in Acu while isCOBOL returns only the machine name.
- The SYSTEM-INFORMATION structure set by ACCEPT FROM SYSTEM-INFO has the following differences between Acu and isCOBOL:
  - The OPERATING-SYSTEM returns the full name of the operating system in isCOBOL. Such name is usually truncated due to the field picture.
  - The STATION-ID field is not set by the isCOBOL runtime, but its value can be set via configuration.
- An unexpected CMD-ACTIVATE event may be generated if the program executes MODIFY statements on the parent window and then performs an ACCEPT on the child window. The unexpected event may be avoided by setting iscobol.gui.cstimeout * to zero in the configuration, but this is not suggested for performance reasons.
- Adding or removing a menu bar or a toolbar on a window always causes the window to be resized to preserve the window's client area (the area where graphical controls can be displayed). It happens with both resizable windows and non-resizable windows. In Acucobol, instead, resizable windows are not resized when a menu bar or a toolbar are added or removed, hence the window's client area changes.
- The window maximization triggered by ACTION-MAXIMIZE is asynchronous, so there's no guarantee that inquiring window and control's dimensions after it will provide updated values.

Library routines

- The following ACUCOBOL-GT library routines are supported by isCOBOL

Library Routine	Known Limitations and differences
$WINHELP	
ASCII2HEX	
ASCII2OCTAL	
C$ASYNCPOLL	
C$ASYNCRUN	
C$CALLEDBY	In isCOBOL, the returned program name is always upper case.

Library Routine	Known Limitations and differences
C$CALLERR	
C$CARG	
C$CHDIR	C$CHDIR as implemented by isCOBOL doesn't actually change the working directory, it just specifies a path that the framework must put before relative file names during i/o. External functions invoked by the programs (like C functions, for example) don't use the directory set by C$CHDIR as working directory. Also commands run either via the SYSTEM library routine or the C$RUN library routine don't use the directory set by C$CHDIR as working directory.
C$CODESET	
C$COPY	Special directory specifiers are not supported.
C$DARG	
C$DELETE	Special directory specifiers are not supported.
C$FILEINFO	
C$FULLNAME	isCOBOL always returns a full path, even if the FILE-PREFIX path where the file is found is a relative path. In some cases C$FULLNAME might not be suitable, for example if the file handler appends a custom extension to the physical name of your files. In these cases, use C$FSFULLNAME.
C$GETCGI	
C$GETLASTFILEOP	The isCOBOL implementation is RM/COBOL compatible. It uses only one parameter that receives the operation name.
C$GETPID	

C$JUSTIFY	The isCOBOL implementation allows to justify only alphanumeric items. If you need to justify numeric edited data items, append "(1:)" to their name, for example:
	`CALL "C$JUSTIFY" USING PICZITEM, "L"`
	where PICZITEM is something like
	`77 PICZITEM PIC Z(3)9.,`
	has to be changed to:
	`CALL "C$JUSTIFY" USING PICZITEM(1:), "L"`

C$KEYMAP	
C$LIST-DIRECTORY	Due to Java implementation listdir-file-creation-time and listdir-file-last-access-time are not returned.
C$LOCKPID	Supported only by c-tree having *iscobol.file.index=ctreej* or *iscobol.file.index=fscsc*. The c-tree client thread ID is returned.
C$MAKEDIR	
C$MYFILE	
C$NARG	

Library Routine	Known Limitations and differences
C$OPENSAVEBOX	Not all opensave flags are supported. The isCOBOL implementation of C$OPENSAVEBOX supports: -opensave-overwriteprompt -opensave-pathmustexist -opensave-filemustexist -opensave-createprompt -opensave-noreadonlyreturn -opensave-browse-dontgobelowdomain -opensave-browse-browseincludefiles
C$PARAMSIZE	
C$PARSEXFD	In isCOBOL the routine is named C$PARSEEFD and it processes EFD dictionaries generated by the isCOBOL compiler. PARSEEFD-TEST-CONDITIONS requires the record buffer, not a pointer to it.
C$RERR	The STATUS-TYPE parameter is not supported.
C$RERRNAME	
C$RUN	
C$SLEEP	
C$SOCKET	C$SOCKET LAST-ERROR op-code returns a limited set of values. See CSOCKET-LAST-ERROR for details.
C$SYSTEM	CSYS-NO-IO and CSYS-COMPATIBILITY flags are not supported.
C$TOUPPER	
C$TOLOWER	
C$XML	The following op-codes are not supported: CXML_GET_RAW_DOCTYPE_LEN, CXML_GET_RAW_DOCTYPE, CXML_SET_RAW_DOCTYPE. Regular expressions are not supported in the op-codes that looks for a elements or attributes by name. The standalone pseudo-attribute is always generated in the XML documents produced by the routine.
CBL_AND	
CBL_CLOSE_FILE	
CBL_COPY_FILE	Possible return codes are the same as the Micro Focus implementation, not just 0 and 1. "@[DISPLAY]" is not supported .
CBL_CREATE_FILE	
CBL_CREATE_DIR	Possible return codes are the same as the Micro Focus implementation, not just 0 and 1. "@[DISPLAY]" is not supported .

Library Routine	Known Limitations and differences
CBL_DELETE_DIR	Possible return codes are the same as the Micro Focus implementation, not just 0 and 1. "@[DISPLAY]" is not supported .
CBL_DELETE_FILE	Possible return codes are the same as the Micro Focus implementation, not just 0 and 1. "@[DISPLAY]" is not supported .
CBL_ERROR_PROC	
CBL_EXIT_PROC	The first parameter can be only 0 or 1. Priorities are not supported.
CBL_FLUSH_FILE	
CBL_NOT	
CBL_OPEN_FILE	
CBL_OR	
CBL_READ_FILE	
CBL_READ_SCR_CHARS	
CBL_READ_SCR_CHATTRS	It handles only the attributes HIGHLIGHT, UNDERLINE and REVERSE-VIDEO assuming that colours are not used.
CBL_WRITE_FILE	
CBL_WRITE_SCR_CHARS	
CBL_WRITE_SCR_CHATTRS	It handles only the attributes HIGHLIGHT, UNDERLINE and REVERSE-VIDEO assuming that colours are not used.
CBL_XOR	
HEX2ASCII	
I$IO	f_errno and other runtime variables have a different declaration, so you must use *isfilesys.def* copybook instead of *filesys.def* provided by ACUCOBOL-GT for the routine to work correctly. isCOBOL I$IO routine does not use the f_int_errno variable. New variables are provided. f_syserr contains the extended description of the error, and f-errmsg contains the error message, if available. op-code 4 (info-function) returns only the following information: logical-params, number of records, collating sequence, keys structure. op-code 15 (execute-function) is not supported.
M$ALLOC	
M$COPY	
M$FILL	

Library Routine	Known Limitations and differences
M$FREE	
M$GET	
M$PUT	
OCTAL2ASCII	
REG_CLOSE_KEY, DISPLAY_REG_CLOSE_KEY REG_CREATE_KEY, DISPLAY_REG_CREATE_KEY REG_CREATE_KEY_EX, DISPLAY_REG_CREATE_KEY_EX REG_DELETE_KEY, DISPLAY_REG_DELETE_KEY REG_DELETE_VALUE, DISPLAY_REG_DELETE_VALUE REG_ENUM_KEY, DISPLAY_REG_ENUM_KEY REG_ENUM_VALUE, DISPLAY_REG_ENUM_VALUE REG_OPEN_KEY, DISPLAY_REG_OPEN_KEY REG_OPEN_KEY_EX, DISPLAY_REG_OPEN_KEY_EX REG_QUERY_VALUE, DISPLAY_REG_QUERY_VALUE REG_QUERY_VALUE_EX, DISPLAY_REG_QUERY_VALUE_EX REG_SET_VALUE, DISPLAY_REG_SET_VALUE REG_SET_VALUE_EX, DISPLAY_REG_SET_VALUE_EX	
R$IO	f_errno and other runtime variables have a different declaration, so you must use *isfilesys.def* copybook instead of *filesys.def* provided by ACUCOBOL-GT for the routine to work correctly. isCOBOL R$IO routine does not use the f_int_errno variable. New variables are provided. f_syserr contains the extended description of the error, and f-errmsg contains the error message, if available. In ACUCOBOL-GT a keyval of "-1" means to write at the end of the file. In isCOBOL it has no special meaning. In ACUCOBOL-GT, successful read operations return the record size plus one. In isCOBOL they just return the record size.
RENAME	The RENAME routine as implemented by isCOBOL doesn't overwrite the destination file if it exists on Windows, but it fails instead.

Library Routine	Known Limitations and differences
S$IO	f_errno and other runtime variables have a different declaration, so you must use *isfilesys.def* copybook instead of *filesys.def* provided by ACUCOBOL-GT for the routine to work correctly. isCOBOL S$IO routine does not use the f_int_errno variable. New variables are provided. f_syserr contains the extended description of the error, and f-errmsg contains the error message, if available. The S-PRINT file type isnot supported. S-SEEK-FUNCTION is not supported.
SYSTEM	The SYSTEM library routine has been implemented using Java API and differs from the one available in other COBOLs that usually reflects the system() C function. The main difference is that isCOBOL performs a redirection of the standard input, output and error, so, if the command passed to the routine contains > or <, it will not work. In order to make isCOBOL use the system() C function instead of the Java API, the following setting must appear in the configuration: `iscobol.system.exec=c`
W$BITMAP	The following op-codes are not supported: WBITMAP-LOAD-IMAGELIST, WBITMAP-DESTROY-IMAGELIST, WBITMAP-CAPTURE-IMAGE, WBITMAP-CAPTURE-DESKTOP, WBITMAP-LOAD-PICTURE. WBITMAP-CAPTURE-IMAGE and WBITMAP-CAPTURE-DESKTOP can be replaced by the use of W$CAPTURE. "@[display]:" in the bitmap name must be replaced by the use of the op-code LOAD-BITMAP-FROM-CLIENT.
W$FONT	Due to Java implementation only true type fonts can be managed.
W$KEYBUF	Only op-codes 1, 2, 3, 4, 5 and 9 are supported. Using op-codes 4 and 5, the recorded keycodes are moved to the buffer only when the registration is stopped.
W$FLUSH	Only op-codes 1, 256 and 257 are supported.
W$MENU	WMENU-POPUP op-code is asynchronous. The equivalent ACUCOBOL-GT feature is synchronous instead.
W$MOUSE	Only the SET-MOUSE-SHAPE and GET-MOUSE-STATUS op-codes are supported. Setting the mouse shape to HELP-POINTER changes the mouse pointer to a hand instead of a question mark and doesn't allow to interact with the help automation. Bitmap-based interfaces can be handled with the MSG-MOUSE-CLICKED, MSG-MOUSE-DBLCLICK, MSG-MOUSE-ENTER, and MSG-MOUSE-EXIT events. Coordinates returned by the GET-MOUSE-STATUS op-code are always relative to the current graphical window. When the mouse pointer is placed on a subwindow, GET-MOUSE-STATUS returns coordinates relative to its parent graphical window.

Library Routine	Known Limitations and differences
W$PALETTE	Redefining a color will affect only the next DISPLAY statements and not the current state of the screen. The WPALETTE-UPDATE op-code is not supported.
W$PROGRESSDIALOG	The op-code WPROGRESSDIALOG-C-COPY and the flag WPROGRESSDIALOG-NOMINIMIZE are not supported. The ANIMATION-TYPE parameter in WPROGRESSDIALOG-CREATE it's also not supported, but you can replace it with the bitmap handle of an icon to be shown top-right of the progress dialog box.
W$TEXTSIZE	
WIN$PLAYSOUND	

Library Routine	Known Limitations and differences
WIN$PRINTER	some runtime variables have a different declaration (in isCOBOL they lack of the SYNC clause), so you must use *isprint.def* copybook instead of *winprint.def* provided by ACUCOBOL-GT for the routine to work correctly. The following op-codes are not supported: WINPRINT-GET-CAPABILITIES WINPRINT-GET-JOB-STATUS WINPRINT-GET-PAGE-COLUMN WINPRINT-GET-PRINTER-STATUS WINPRINT-GET-SETTINGS-SIZE WINPRINT-SET-PRINTER and WINPRINT-SET-PRINTER-EX settings are evaluated only at the OPEN OUTPUT of the print-file. Calling these op-codes in the middle of the print job has no effect. If WINPRINT-GET-PRINTER-INFO, WINPRINT-GET-CURRENT-INFO, WINPRINT-GET-PRINTER-INFO-EX, and WINPRINT-GET-CURRENT-INFO-EX op-codes are called before WINPRINT-SETUP, they return only the following information: winprint-name, winprint-no-of-printers, winprint-is-default, and winprint-job-title. All other fields are set to default values that may not match with the current printer settings. WPRTMARGIN-CELLS and WPRTMARGIN-PIXELS are not supported for WINPRINT-SET-MARGINS op-code. Due to different geometries, the border of the shapes created by WINPRINT-GRAPH-DRAW op-code is more marked in isCOBOL. The difference is more evident with the increasing of the value of WPRTDATA-PEN-WIDTH. WINPRINT-GRAPH-DRAW, WINPRINT-PRINT-BITMAP, WINPRINT-SET-PAGE-COLUMN and WINPRINT-SET-CURSOR coordinates are always absolute (e.g. *wprtunits-centimeters* and *wprtunits-centimeters-abs* have the same meaning). Margins set by WINPRINT-SET-MARGINS are applied at the close of the print file and affect all the pages of the print job. If the WINPRINT-SET-MARGINS function is called multiple times within the same print job, only margins set by the last call are considered. In Acucobol the coordinates of a shape drawn by WINPRINT-GRAPH-DRAW are relative to the paper border. In isCOBOL instead, they're relative to the margis set by WINPRINT-SET-MARGINS. WINPRINT-SET-CURSOR can't be used to retrieve the new cursor position after a WRITE statement. Bitmaps and fonts must not be destroyed before the close of the print file, otherwise they will not appear in the printer output.
WIN$VERSION	

All the other routines must be replaced with the corresponding isCOBOL routine or with an isCOBOL syntax that has the same effect, if possible.
The following table lists the known solutions to obtain the same effect of ACUCOBOL-GT routines.

Library routine	isCOBOL solution
C$CHAIN	Use the CHAIN verb.
C$JAVA	Use object oriented syntax to invoke Java methods.
W$BROWSERINFO	Only in Web Direct 2.0 environment, call WD2$CLIENT_INFO.

If the routine that you're looking for doesn't appear in the above list, contact Veryant's support to discuss about possible solutions.

Integrated Development Environments

AcuBench and Totem projects can be imported in the isCOBOL IDE and maintained with it.

See Importing programs from AcuBench and Importing programs from Totem for more information.

Web

- CGI programs developed with ACUCOBOL-GT can be easily transformed into Java Servlets. See Conversion of the ACUCOBOL-GT Oscars sample in the EIS documentation for details.
- COBOL program working with the ACUCOBOL-GT Web Runtime can be run using isCOBOL Web Direct 2.0. Check Web Direct 2.0 Technical Notes in the EIS documentation for details.

Transitioning from IBM COBOL

The following sections describe the key areas which need to be addressed during the transition from IBM COBOL to isCOBOL.

Compiler options

- When compiling source files of a IBM COBOL application, the following options should always be used:

```
-cv -cva
```

- In addition, depending on the way you need COMP fields to be represente, one of these two options should be used:

`-dci`	Use IBM sign encoding and IBM COMP sizes. COMP sizes are 1, 2, 4, 8, 12 or16 depending on the item picture. See USAGE clause for details about how numeric data items are affected by this option.
`-dcii`	Use IBM sign encoding and IBM COMP sizes. COMP sizes are 2, 4, 8 or16 depending on the item picture. See USAGE clause for details about how numeric data items are affected by this option.

Data access

- Fixed length sequential and relative files coming from IBM COBOL are fully supported by isCOBOL.
- Indexed files from IBM COBOL must be converted either to c-tree or JIsam with the following steps:
  a. unload file data to a raw binary file with IBM COBOL tools
  b. create an empty c-tree or JIsam indexed file by running a program that performs opens the file for output with isCOBOL
  c. load data from the raw binary file into the empty indexed file with ctutil or JUTIL.

Syntax and Behavior

- The CHANGED flag in the EXHIBIT statement is ignored.

Transitioning from ICOBOL

The following sections describe the key areas which need to be addressed during the transition from ICOBOL to isCOBOL.

Compiler options

- When compiling source files of an ICOBOL application, the following options should always be used:

```
-ci -dcd -dz
```

Run time and configuration

- By default, isCOBOL uses 2002 ANSI standard file status codes. These codes differ from those usually used by ICOBOL. To have the same file status codes, add the following line to the isCOBOL configuration:

```
iscobol.file.status=com.iscobol.io.FileStatusDG
```

- The following entry should be used in the configuration in order to have the same screen positioning rules of ICOBOL:

```
iscobol.gui.screen_col_plus_base=0
```

Data access

- Fixed length sequential and relative files coming from ICOBOL are fully supported by isCOBOL.
- Indexed files from ICOBOL must be converted either to c-tree or JIsam with the following steps:
  a. unload file data to a raw binary file with ICOBOL tools
  b. create an empty c-tree or JIsam indexed file by running a program that performs opens the file for output with isCOBOL
  c. load data from the raw binary file into the empty indexed file with ctutil or JUTIL.

Library routines

ICOBOL contains several library routines that can be called. These routines start with a "#" character. isCOBOL doesn't support these routines. You will have to rename the routines and code them in C, Java or COBOL if you want to use them.

Transitioning from MBP COBOL

The following sections describe the key areas which need to be addressed during the transition from MBP COBOL to isCOBOL.

Compiler options

- When compiling source files of a MBP COBOL application, the following option should always be used:

```
-dcb
```

Data access

- Fixed length sequential and relative files coming from MBP COBOL are fully supported by isCOBOL.
- Indexed files from MBP COBOL must be converted either to c-tree or JIsam with the following steps:
  a. unload file data to a raw binary file with MBP COBOL tools
  b. create an empty c-tree or JIsam indexed file by running a program that performs opens the file for output with isCOBOL
  c. load data from the raw binary file into the empty indexed file with ctutil or JUTIL.

Transitioning from MicroFocus

isCOBOL Evolve has a good compatibility with Micro Focus syntax and features.

The following sections describe the key areas which need to be addressed during the transition.

Compiler options

- When compiling source files of a Micro Focus application, the following options should always be used:

```
-cm -dcm
```

- If you were using some Microsoft Cobol extensions, then the following option should be used as well:

```
-cms
```

- Micro Focus compile flags are not supported by isCOBOL, but in most cases their effect can be obtained using one of the isCOBOL Compiler options or in another way. The following table lists the Micro Focus compile flags whose effect can be obtained also with isCOBOL:

Micro Focus flag	**isCOBOL solution**
ADDSYN	Use -rm= compiler option
ALIGN	Use -align= compiler option
ANIM	Use -d compiler option
APOST	Use -apost compiler option
ARITHMETIC=OSVS	Use -cva compiler option
ASSIGN	Use -cax compiler option
BOUND	Set iscobol.array_check=1 in the configuration and compile with -m1 option
CHANGE-MESSAGE	Set iscobol.compiler.messagelevel properly in the configuration
CHECKDIV	Default behavior using isCOBOL
CHECKDIV(*dialect*)	Set iscobol.checkdiv=1 in the configuration if *dialect* is one of the following: COBOL370, ENTCOBOL, MVS, OS390, OSVS or VSC2. Set iscobol.checkdiv=0 (or don't set iscobol.checkdiv) in the configuration if *dialect* is either ANSI or ISO2002.
CHECKNUM	Set iscobol.check.numeric_content=1 in the configuration
COBFSTATCONV	Set iscobol.file.status properly in the configuration

Micro Focus flag	isCOBOL solution
COBOL370	Use -cv compiler option
CONSTANT	Set iscobol.compiler.constant properly in the configuration
CONVERTRET	Use -d5 compiler option
COPYEXT	Use -ce= compiler option
COPYLIST	Use -lf compiler option
CURRENT-DATE	Take advantage of the CurrentDate Class (com.iscobol.rts.CurrentDate) feature
DATAMAP	Use -ld compiler option
DEFAULTBYTE	Use -dv= compiler options
DOS/VS	Use -cv compiler option
ERRLIST	Use -ef compiler option
FAULTFIND	Set iscobol.exception.dump=1 and iscobol.display_message=3 in the configuration
FLAGQ	Use -es compiler option
FOLD-COPY-NAME	Use -scnl and -scnu compiler options
FOLD-CALL-NAME	Use -ssnl and -ssnu compiler options
HIDE-MESSAGE	Set iscobol.compiler.messagelevel properly in the configuration
IBMCOMP	Use -dcmi compiler option instead of -dcm
INDD	Set iscobol.compiler.indd in the configuration
KEYCHECK	Set iscobol.file.extra_keys_ok properly in the configuration
LIST, LISTPATH, LISTWIDTH, LW	Use -lf and -lo= compiler options
MAKESYN	Use -rm compiler option
NOTRUNC	Use -dznt compiler option
NSYMBOL	If "DBCS", use -cndbcs compiler option If "NATIONAL", no action required
ODOSLIDE	Use -cod1 compiler option
OPTIONAL-FILE	Set iscobol.file.io_creates=1 and iscobol.file.extend_creates=1 in the configuration
OVERRIDE	Use -rc compiler option
OSVS	Use -cv compiler option

Micro Focus flag	isCOBOL solution
OUTDD	Set iscobol.compiler.outdd in the configuration
PERFORM-TYPE	Use either -pt0 or -pt1 or -pt2 compiler options
PRINT	Use -lo compiler option
REMOVE	Use -rw= compiler option
RM	Use -dcii compiler option
SEQUENTIAL	Sequential files type is controlled by -cfl compiler option
SIGN	If "ASCII", use -dcm compiler option If "EBCDIC", use -dci or -dcii compiler option
SOURCEFORMAT	If "FIXED", use -sa compiler option If "FREE", use -sf compiler option If "VARIABLE", use -sv compiler option
SPZERO	Default behaviour using isCOBOL
TRACE	Set iscobol.tracelevel properly in the configuration
TRUNC"ANSI"	Use -dzta compiler option

Data access

- Fixed-length sequential and relative files created by Micro Focus programs are directly usable by isCOBOL programs.
- To manage variable-length sequential files, add the following entry to the configuration:

```
iscobol.file.sequential=mfsequential
```

- To manage line sequential files that might contain 0x0A in the record due to COMP fields, add the following entry to the configuration:

```
iscobol.file.linesequential=lseqmf_n
```

- Variable-length relative files must be converted to fixed-length relative files as isCOBOL doesn't support variable-length relative files. This kind of conversion can be done through a Micro Focus program that reads from the variable-length file and writes to the fixed-length file using the maximum record size from the input file as the record size of the output file.
- Indexed MF files are not handled by isCOBOL. They can be easily converted to JIsam or c-tree by ISMIGRATE (Index File Migration) utility. Choose "com.iscobol.io.ScanMF" as input file handler to let ISMIGRATE read Micro Focus files without the need of external components.
- By default, isCOBOL returns ANSI 2002 file status codes. In order to have the same Micro Focus file status code, add the following setting to the configuration:

```
iscobol.file.status=com.iscobol.io.FileStatusMF
```

- The file status "30" is returned when you you try to create a file where there are no write permissions. Micro Focus returns "37" instead.
- If the Micro Focus application worked on databases using the Database Connector feature, consider that the same feature can be obtained with isCOBOL by taking advantage of isCOBOL Database Bridge. If you choose the EDBI Generation with EDBIIS (two steps), then the -ca option should be used when launching the EDBI routine generator (edbiis). If you choose EDBI Generation at compile time (one step), then you should gather all your FDs in a dummy program and compile only that program with -ca Compiler option, as compiling all programs of a Micro Focus application with -ca is not suggested.

Syntax and Behavior

- The PROGRAM-ID paragraph must be explicitly declared, and the name of the program must match the name of the source file.
- If a paragraph is declared, then its parent section must be declared as well. For example, INPUT-OUTPUT SECTION must be declared above FILE-CONTROL.
- The $SET OVERRIDE directive that changes the meaning of a reserved word is not supported. The same effect can be obtained by compiling with -rc= option.
- The syntax for inline comments with isCOBOL is |*comment*, not *//comment*.
- The LOCAL-STORAGE SECTION is not supported by isCOBOL. The behavior of data-items in recursive programs is controlled by the configuration property iscobol.recursion_data_global.

- The syntax *z"String"* is not supported by isCOBOL. In order to append 0x00 to a text string you must take advantage of string concatenations. For example:

```
move z"my-literal" to an-item
```

  can be changed to:

```
move "my-literal" & x"00" to an-item
```

- Micro Focus provides the ability to define new COBOL types in a C language style using type definition, while isCOBOL not.
- EVENT-POINTER, MONITOR-POINTER, MUTEX-POINTER, SEMAPHORE-POINTER, PROCEDURE-POINTER, PROGRAM-POINTER and THREAD-POINTER are not supported by isCOBOL.
- COMMON and EXTERNAL clauses are not supported in PROGRAM-ID.
- CALL-CONVENTION is not supported. You can configure the call convention by setting iscobol.dll_convention in the configuration.
- The following syntaxes are not supported for ASSIGN TO phrase in FILE-CONTROL:
  - FROM dataname
  - LINE ADVANCING
  - LPT1, LPT2, ...
  - PRN
- SUPPRESS is not supported in file keys description.
- The RETURN clause is not supported in I-O CONTROL paragraph.
- The VALUE clause is not supported in REDEFINES.
- The COMMUNICATION SECTION is not supported by isCOBOL.
- BY REFERENCE/VALUE/CONTENT cannot be used for parameters list either after PROCEDURE DIVISION USING or ENTRY...USING or in CHAIN and INVOKE statements. This syntax is supported only by CALL statement.
- The ACCEPT of a data-item does not show the initial value of the item unless you specify the WITH UPDATE clause. You can assume this clause for all ACCEPT statements by compiling with -vu option.
- The ACCEPT of a non-edited data item fills the item differently. Suppose to have a variable defined as PIC 9(3) with a value of 123. When the ACCEPT is performed, if the user inputs 5, the value of the variable with Micro Focus becomes 523, while isCOBOL stores 005.
- ACCEPT FROM EXCEPTION STATUS is not supported. A similar effect can be obtained by specifying the ON EXCEPTION and NOT ON EXCEPTION clauses in the ACCEPT.
- ACCEPT FROM USER NAME is not supported. You can query the iscobol.user_id property using ACCEPT FROM ENVIRONMENT in order to retrieve this kind of information.
- ACCEPT FROM CRT is not supported.
- The MODE IS BLOCK clause is not supported in ACCEPT.
- The WITH TIMEOUT clause is not supported in ACCEPT, you must replace it with a BEFORE TIME clause in order to obtain the same effect.
- The ALTER statement is not currently supported by isCOBOL.
- ADDRESS OF data-item cannot be passed as parameter between programs or used as program exit status.
- The AS clause is not supported in CALL.

- The FOR REMOVAL clause in CLOSE statement is not supported. Replace it with WITH LOCK in order to obtain the same effect.
- The WITH DISP clause is not supported in CLOSE.
- OPEN and CLOSE statements cannot be used on pointers. You can use it only on files.
- DISPLAY SPACES doesn't clear the screen unless you specify the ERASE EOS clause.
- The ENTER statement, that allows to switch to another programming language, is not supported by isCOBOL.
- The CHANGED flag of the EXHIBIT statement is ignored.
- INITIALIZE category TO VALUE is not supported by isCOBOL.
- The ON statement is not supported by isCOBOL.
- The RESERVED clause is not supported in OPEN statement.
- The WITH WAIT clause is not supported in READ statement.
- START THREAD is not supported by isCOBOL.
- The ADVANCINT TAB/FORMATTED clause is not supported in WRITE statement.
- The $ character as first digit of physical file name is not supported by isCOBOL.
- VALUE NEXT syntax requires -m1 and -align=8 compile flags to reproduce the Micro Focus behavior.
- Copybooks whose content is not terminated by a dot are not allowed. There are two solutions:
    - add a dot after the last line of the copybook content, or
    - use two dots to close the COPY statement (e.g. `COPY "mycopy.prd"..`)

Configuration

- iscobol.substring.zero_len_all (boolean) * should be set to false since Micro Focus allowed reference modifiers with length = 0.
- iscobol.file.index.check_all_keys (boolean) should be set to true in the configuration in order to activate the check on keys structure.
- iscobol.memory.alpha_edited (boolean) should be set to true in the configuration for compatibility on alphabetic-edited and alphanumeric-edited items setting.
- iscobol.terminal.autowrap (boolean) should be set to true since Micro Focus automatically wraps text in the terminal.
- If you configured your Micro Focus runtime to have the maximum precision on calculations, then your programs should be compiled in isCOBOL with the option -cfp36 in order to have the same maximum precision.
- isCOBOL provides a special class, the CurrentDate Class (com.iscobol.rts.CurrentDate), that allows to obtain the same effect of the following Micro Focus configuration entries:
    - current_day
    - current_hour
    - current_minute
    - current_month
    - current_second

- current_year
- datewarp_dynamic

Library routines

- The following Micro Focus library routines are supported by isCOBOL:

Micro Focus routine	Known Limitations and differences
CBL_ALLOC_MEM	The flags parameter is ignored.
CBL_AND	
CBL_CHANGE_DIR	CBL_CHANGE_DIR as implemented by isCOBOL doesn't actually change the working directory, it just specifies path that the framework must put before relative file names during i/o. External functions like commands run through C$SYSTEM or system API functions don't use the directory set by CBL_CHANGE_DIR as working directory.
CBL_CHECK_FILE_EXIST	
CBL_CLEAR_SCR	
CBL_CLOSE_FILE	
CBL_COPY_FILE	
CBL_CREATE_FILE	
CBL_CREATE_DIR	
CBL_DELETE_DIR	
CBL_DELETE_FILE	
CBL_DIR_SCAN_END	
CBL_DIR_SCAN_READ	
CBL_DIR_SCAN_START	
CBL_EQ	

Micro Focus routine	Known Limitations and differences

CBL_ERROR_PROC	The usage in Micro Focus is:
	
	`77 proc usage procedure-pointer.`
	`set proc to entry "err-prog"`
	`call "cbl_error_proc" using flag, proc`
	
	The corresponding usage in isCOBOL is:
	
	`call "cbl_error_proc" using flag, "err-prog"`

CBL_EXEC_RUN_UNIT	With isCOBOL it's not possible to assign a dedicated console to the new run unit. The choice is between sharing stdout and stderr of the new run unit with the ones of the caller program or lose the messages that the new run unit prints on stdout and stderr.
CBL_EXIT_PROC	
CBL_FLUSH_FILE	
CBL_FREE_MEM	
CBL_GET_CURRENT_DIR	
CBL_GET_SCR_SIZE	
CBL_IMP	
CBL_JOIN_FILENAME	
CBL_NOT	
CBL_OPEN_FILE	
CBL_OR	
CBL_READ_DIR	
CBL_READ_FILE	
CBL_READ_SCR_CHARS	
CBL_READ_SCR_CHATTRS	It handles only the attributes HIGHLIGHT, UNDERLINE and REVERSE-VIDEO assuming that colours are not used.
CBL_RENAME_FILE	
CBL_SPLIT_FILENAME	
CBL_TOLOWER	
CBL_TOUPPER	
CBL_WRITE_FILE	
CBL_WRITE_SCR_CHARS	

Micro Focus routine	Known Limitations and differences
CBL_WRITE_SCR_CHATTRS	It handles only the attributes HIGHLIGHT, UNDERLINE and REVERSE-VIDEO assuming that colours are not used.
CBL_WRITE_SCR_N_CHAR	
CBL_WRITE_SCR_N_CHATTR	It handles only the attributes HIGHLIGHT, UNDERLINE and REVERSE-VIDEO assuming that colours are not used.
CBL_XOR	
SYSTEM	

All the other routines must be replaced with the corresponding isCOBOL routine or with an isCOBOL syntax that has the same effect, if possible.
The following table lists the known solutions to obtain the same effect of Micro Focus routines.

Micro Focus routine	isCOBOL solution
_CODESET	use C$CODESET routine
CBL_ABORT_RUN_UNIT	use STOP RUN statement
CBL_ALLOC_DYN_MEM CBL_ALLOC_SHMEM CBL_ALLOC_THREAD_MEM	use M$ALLOC routine
CBL_CANCEL	use CANCEL statement
CBL_DEBUGBREAK	A similar feature can be obtained with the STOP statement.
CBL_FILENAME_CONVERT	Use INSPECT... REPLACING statement
CBL_FREE_DYN_MEM	use M$FREE routine
CBL_FREE_RECORD_LOCK	use R$IO routine
CBL_GET_CSR_POS	test the CURSOR Special Name
CBL_GET_MOUSE_STATUS	use W$MOUSE routine, GET-MOUSE-STATUS function
CBL_GET_OS_INFO	use C$SYSINFO routine
CBL_INIT_MOUSE CBL_TERM_MOUSE	no need to enable/disable the mouse support. It's always enabled by default in isCOBOL
CBL_LOCATE_FILE	use C$FULLNAME or C$FSFULLNAME routines depending on the file type
CBL_MANAGED_SESSION_GET_USER DATA CBL_MANAGED_SESSION_SET_USER DATA	the closer thing to this feature are the environment variables that you can set with SET ENVIRONMENT and inquire with ACCEPT FROM ENVIRONMENT

Micro Focus routine	isCOBOL solution
CBL_NLS_INFO	it's possible to get (but not to set) national language information by calling the C$GETENV routine
CBL_READ_KBD_CHAR	use ACCEPT... NO ECHO
CBL_TEST_RECORD_LOCK	use R$IO routine
CBL_TOLOWER	use C$TOLOWER routine
CBL_TOUPPER	use C$TOUPPER routine
JVM_LOAD_NATIVE	use CALL statement
PC_PRINTER_CLOSE	use CLOSE statement on the print-file[A]
PC_PRINTER_CONTROL	use WRITE statement on the print-record[B] with BEFORE and AFTER clauses
PC_PRINTER_DEFAULT_FONT	use WIN$PRINTER routine, WINPRINT-SET-FONT function
PC_PRINTER_DEFAULT_NAME	use WIN$PRINTER routine, WINPRINT-SET-PRINTER function
PC_PRINTER_DEFAULT_PROPERTIES	use WIN$PRINTER routine, WINPRINT-SET-PRINTER-EX function
PC_PRINTER_FREE_BMP	use W$BITMAP routine, WBITMAP-DESTROY function (to be done after closing the print-file[A])
PC_PRINTER_INFO	use WIN$PRINTER routine, WINPRINT-GET-CURRENT-INFO-EX and WINPRINT-GET-PRINTER-INFO-EX functions
PC_PRINTER_LOAD_BMP	use W$BITMAP routine, WBITMAP-LOAD function
PC_PRINTER_OPEN	use OPEN OUTPUT statement on the print-file[A]
PC_PRINTER_SET_COLOR	use WIN$PRINTER routine, WINPRINT-SET-BACKGROUND-COLOR and WINPRINT-SET-TEXT-COLOR functions
PC_PRINTER_SET_DEFAULT	use WIN$PRINTER routine, WINPRINT-SET-PRINTER function
PC_PRINTER_SET_FONT	use WIN$PRINTER routine, WINPRINT-SET-FONT function
PC_PRINTER_WRITE	use WRITE statement on the print-record[B]
PC_PRINTER_WRITE_BMP	use WIN$PRINTER routine, WINPRINT-PRINT-BITMAP function
PC_READ_DRIVE	retrieve the current directory (see CBL_GET_CURRENT_DIR above) and consider only the first byte.
X"91" function 16	use C$NARG routine
X"AF" function 22 X"E5"	use WIN$PLAYSOUND routine

[A] *print-file* is a sequential file defined as follows

```
SELECT print-file ASSIGN TO PRINT "-P SPOOLER".
```

[B] *print-rec* is the record definition for print-file, it's size should match the max number of digits that can be printed on a line. E.g.

```
FD print-file.
01 print-rec PIC X(80).
```

If the routine that you're looking for doesn't appear in the above list, contact Veryant's support to discuss about possible solutions.

Intrinsic Functions

The following intrinsic functions are not supported by isCOBOL:

- CHAR-NATIONAL
- LENGTH-AN

CGI

See Conversion of the Micro Focus sample in the EIS documentation for information about how to convert Micro Focus CGI programs to isCOBOL Servlets.

Transitioning from Microsoft COBOL

The following sections describe the key areas which need to be addressed during the transition from Microsoft COBOL to isCOBOL.

Compiler options

- When compiling source files of a Microsoft COBOL application, the following option should always be used:

```
-cms
```

Run time and configuration

- By default, isCOBOL uses 2002 ANSI standard file status codes. These codes differ from those usually used by Microsoft COBOL. To have the same file status codes, add the following line to the isCOBOL configuration:

```
iscobol.file.status=com.iscobol.io.FileStatusMS
```

Data access

- Fixed length sequential and relative files coming from Microsoft COBOL are fully supported by isCOBOL.
- Indexed files from Microsoft COBOL must be converted either to c-tree or JIsam with the following steps:
  a. unload file data to a raw binary file with Microsoft COBOL tools
  b. create an empty c-tree or JIsam indexed file by running a program that performs opens the file for output with isCOBOL
  c. load data from the raw binary file into the empty indexed file with ctutil or JUTIL.

Transitioning from NCR COBOL

The following sections describe the key areas which need to be addressed during the transition from NCR COBOL to isCOBOL.

Compiler options

- When compiling source files of a NCR COBOL application, the following option should always be used:

```
-dcn
```

Data access

- Fixed length sequential and relative files coming from NCR COBOL are fully supported by isCOBOL.
- Indexed files from NCR COBOL must be converted either to c-tree or JIsam with the following steps:
  a. unload file data to a raw binary file with NCR COBOL tools
  b. create an empty c-tree or JIsam indexed file by running a program that performs opens the file for output with isCOBOL
  c. load data from the raw binary file into the empty indexed file with ctutil or JUTIL.

Transitioning from Realia COBOL

The following sections describe the key areas which need to be addressed during the transition from Realia COBOL to isCOBOL.

Compiler options

- When compiling source files of a Realia COBOL application, the following option should always be used:

```
-dcr
```

Data access

- Fixed length sequential and relative files coming from Realia COBOL are fully supported by isCOBOL.
- Indexed files from Realia COBOL must be converted either to c-tree or JIsam with the following steps:
  a. unload file data to a raw binary file with Realia COBOL tools
  b. create an empty c-tree or JIsam indexed file by running a program that performs opens the file for output with isCOBOL
  c. load data from the raw binary file into the empty indexed file with ctutil or JUTIL.

Transitioning from RM/COBOL

The following sections describe the key areas which need to be addressed during the transition from RM/COBOL to isCOBOL.

Compiler options

- When compiling source files of a RM/COBOL application, the following options should always be used:

```
-cr -ce=cbl -cko -crlk -dcii -dvext=32 -pt0 -va
```

- Many RM/COBOL applications imply the SIGN IS TRAILING SEPARATE clause for signed data items. RM/COBOL version 1.6 behaved this way. RM/COBOL version 2.x can optionally behave this way. If the application was written originally with version 2.0 or later, then it might not behave this way.
  You will need to examine the source to see if the "S" picture element counts in the size of an item or not. It is important to determine whether or not the sign counts in the size, particularly if you plan to convert existing data files, otherwise records loaded from RM/COBOL files may not match the record layout described in the program.
  To imply the SIGN IS TRAILING SEPARATE clause with isCOBOL, use the following compiler option:

```
-ds
```

- With RM/COBOL, the default intensity during ACCEPT for UNIX machines is high intensity and for MS-DOS machines is low intensity. With isCOBOL every ACCEPT uses low intensity by default. If you need to force a default high intensity on ACCEPT, use the following compiler option:

```
-vh
```

- Many RM/COBOL applications use a particular mode of memory management. In this mode, all of the currently active programs must fit in 64K bytes, but programs that are not currently active need not. This has the effect of dynamically canceling inactive subprograms as needed to grain memory. To obtain a similar behavior with isCOBOL, use the following compiler option:

```
-zi
```

Run time and configuration

- iscobol.file.index.check_all_keys (boolean) should be set to true in the configuration in order to activate the check on keys structure.
- iscobol.gui.screen_col_zero (boolean) should be set to true in the configuration for compatibility on the DISPLAY at COLUMN 0.
- iscobol.memory.alpha_edited (boolean) should be set to true in the configuration for compatibility on alphabetic-edited and alphanumeric-edited items setting.
- iscobol.terminal.autowrap (boolean) should be set to true since RM/COBOL automatically wraps text in the terminal.

- By default, isCOBOL uses 2002 ANSI standard file status codes. These codes differ from those usually used by RM/COBOL. If RM/COBOL programs were written to the 1974 RM/COBOL standard, you should add the following line to the configuration:

```
iscobol.file.status=com.iscobol.io.FileStatus74
```

  If RM/COBOL programs were written in RM/COBOL-85, instead, set:

```
iscobol.file.status=com.iscobol.io.FileStatus85
```

- RM/COBOL (but not RM/COBOL-85) automatically closes all non-print files in a subprogram when that program exits. This is not default behavior with isCOBOL, but you can obtain it by adding the following entry to the configuration:

```
iscobol.file.close_on_exit=1
```

  or by compiling programs with the following compiler option:

```
-coe
```

- Under RM/COBOL, if the main program has a Linkage Section, it is initialized by the parameter passed on the command line. Under isCOBOL, it is illegal to reference Linkage Section items in the main program. You can get rid of this difference by using a stub main program that takes care of intercepting the command-line and pass it to the original main program through a CALL statement.
  A basic implementation of this stub main program is installed with isCOBOL in the *sample/custom-startup/rm-run* folder. The program is intended to be used as replacement of RM/COBOL runcobol command and supports the following command-line flags: A for arguments, S for switches and T for memory allocation.
  If you used a command like this in RM/COBOL:

```
runcobol test1 a = "a b c" S = 10101111 -T = 512000
```

  You can replace it with:

```
ISCRUN RUNCOBOL test1 a = "a b c" S = 10101111 -T = 512000
```

Data access

- RM/COBOL indexed files can be easily converted to JIsam, c-tree or other supported file systems using the ISMIGRATE (Index File Migration) utility. Choose "com.iscobol.io.ScanRMKF" as input file handler to let ISMIGRATE read RM/COBOL files without the need of external components.
- Fixed-length sequential and relative files created by 1974 RM/COBOL programs are directly usable by isCOBOL programs. To convert RM/COBOL-85 relative files with fixed-length records, write an RM/COBOL program that reads the file and writes the records to a fixed-length binary sequential file. The resulting relative file is usable by isCOBOL.
- Set the following configuration property for a RM/COBOL compatible line separator handling:

```
iscobol.file.linesequential=lseqacu
```

- Variable-length sequential files must be converted through a RM/COBOL program that reads from the variable-length file and writes the records to a new fixed-length record binary sequential output file but begin the output file's FD with a two byte COMP-4 item that is filled in with the record size from the input file's depend-variable. Use this new file as the variable-length record binary sequential file with isCOBOL.
- Variable-length relative files must be converted to fixed-length relative files as isCOBOL doesn't support variable-length relative files. This kind of conversion can be done through a RM/COBOL program that reads from the variable-length file and writes to the fixed-length file using the maximum record size from the input file as the record size of the output file.

Syntax and Behavior

In RM COBOL the PROGRAM-ID special register returns the program name exactly as it appears in the PROGRAM-ID paragraph. In isCOBOL instead it returns the name of the program class stripped of the class extension.

The "SET POINTER TO ADDRESS OF *DataItem*" statement is not supported. There are two possible solutions:

- compile the program with the *-cp* option. This is suggested if you need to pass the pointer to external C functions.
- compile the program with *-rm=handle,address*. This is suggested if you use the pointer internally in the COBOL program.

Library routines

The following RM/COBOL library routines are supported by isCOBOL:

Library Routine	Known Limitations and differences
C$CARG	
C$CENTURY	
C$DARG	
C$DELAY	

Library Routine	Known Limitations and differences
C$GETENV	
C$GETLASTFILENAME	
C$GETLASTFILEOP	
C$GUICFG	
C$MBAR	Hovering the mouse over the menu items doesn't update the status bar, but shows a tool tip instead.
C$NARG	
C$RBMENU	Hovering the mouse over the menu items doesn't update the status bar, but shows a tool tip instead.
C$RERR	
C$SBAR	
C$SCRD	
C$SCWR	The optional parameters ATTRIBUTE-CODES and PALETTE-TABLE are not supported.
C$SETDEVELOPMENTMODE	Mode must be "Absolute".
C$SETENV	
C$SHOW	Only the flags SW-HIDE and SW-SHOWNORMAL are supported. Any other flag is treated as SW-SHOWNORMAL.
C$TBAR	Icon files are not included in the runtime, they must be provided separately in the form of png files. Hovering the mouse over the buttons doesn't update the status bar, but shows a tool tip instead.
C$WRU	The second and third parameters are always 0. In isCOBOL the routine returns the class name stripped of the "class" extension, not the Program-Id.
DELETE	
P$CLEARDIALOG	
P$CLEARFONT	
P$DISABLEDIALOG	
P$DISPLAYDIALOG	
P$DRAWBITMAP	Mode must be "Absolute".
P$DRAWBOX	Mode must be "Absolute".
P$DRAWLINE	Mode must be "Absolute".

Library Routine	Known Limitations and differences
P$ENABLEDIALOG	
P$GETDEVICECAPABILITIES	Only these attributes are supported: "Driver Version" return 0; "Technology" returns 2 (Printer) "Horizontal Size" horizontal size of the biggest supported page "Vertical Size" vertical size of the biggest supported page "Horizontal Resolution" Width of the biggest printable area, in dots. "Vertical Resolution" Height of the biggest printable area, in dots. "Logical Pixels X" maximum horizontal resolution in DPI "Logical Pixels Y" maximum vertical resolution in DPI "Aspect X" same as "Logical Pixels X" "Aspect Y" same as "Logical Pixels Y" "Aspect XY" Diagonal of a square whose sides are the 2 above "Physical Width" same as "Horizontal Size" "Physical Height" same as "Vertical Size" "Physical Offset X" starting x offset of the printable are "Physical Offset Y" starting y offset of the printable are "Scaling Factor X" always 0 "Scaling Factor Y" always 0.
P$GETDIALOG	Only these attributes are supported: "Print Dialog Copies" "Device Name" "Orientation" "Paper Size" "Device Mode Copies" "Default Source" "Print Quality" "Color"
P$GETFONT	Only these attributes are supported: "Height" "Width" "Escapement" "Weight" "Italic" "Underline" "Strike Out" "Pitch" "Face Name"
P$GETTEXTMETRICS	Only these attributes are supported: "Height" "Ascent" "Descent" "Internal Leading" "External Leading" "Average Character Width" "Maximum Character Width" "Weight" "Italic" "Underlined" "Struck Out" "Pitch"

Library Routine	Known Limitations and differences
P$NEWPAGE	Can't specify orientation.
P$SETDEFAULTMODE	
P$SETDEFAULTUNITS	
P$SETDIALOG	Only these attributes are supported: "Print Dialog Copies" "Device Name" "Orientation" "Paper Size" "Device Mode Copies" "Default Source" "Print Quality" "Color"
P$SETDOCUMENTNAME	
P$SETFONT	Only these attributes are supported: "Height" "Width" "Escapement" "Weight" "Italic" "Underline" "Strike Out" "Pitch" "Face Name"
P$SETPEN	
P$SETPOSITION	Mode must be "Absolute".
P$SETTEXTCOLOR	
P$SETTEXTPOSITION	Mode must be "Absolute".
P$SETTOPMARGIN	The measure unit in "Character" is not supported
P$TEXTOUT	Mode must be "Absolute".
RENAME	
SYSTEM	

All the other routines must be replaced with the corresponding isCOBOL routine or with an isCOBOL syntax that has the same effect, if possible.

The following table lists the known solutions to obtain the same effect of RM/COBOL routines.

RM/COBOL routine	isCOBOL solution
C$BITMAP	Use W$BITMAP with WBITMAP-DISPLAY op-code

RM/COBOL routine	isCOBOL solution
C$GETSYSINFO	Use WIN$VERSION for Windows information. On other systems, call system APIs. Use C$GETPID to retrieve the process id. Use J$NETADDRESS to retrieve the computer name.
C$LOGICALAND	Use CBL_AND
C$LOGICALOR	Use CBL_OR
C$LOGICALXOR	Use CBL_XOR
C$MEMORYALLOCATE	Use M$ALLOC
C$MEMORYDEALLOCATE	Use M$FREE
C$PLAYSOUND	Use WIN$PLAYSOUND
C$SECUREHASH	Use A$ENCRYPT
C$TBAREN	Modify the ENABLE property of buttons in the TOOL-BAR
C$TBARSEQ	Modify BITMAP and BITMAP-NUMBER properties of buttons in the TOOL-BAR
C$TITLE	Modify the TITLE property of the WINDOW

If the routine that you're looking for doesn't appear in the above list, contact Veryant's support to discuss about possible solutions.

Cobol-WOW

Cobol-WOW is Liant Software Corporation's graphical user interface development tool for RM/COBOL. The goal of Veryant's Cobol-WOW replacement is to provide a new and modern environment based on Java. The WOW developer will find the same concepts used in Cobol-WOW designer in Veryant's solution.

All the standard GUI widgets were rewritten in Java and are supported. All the GUI widget behaviors were implemented so the WOW migrated application behaves as originally designed by the COBOL developer.

Veryant's IDE is able to import a Cobol-WOW project and allows the user to maintain the programs via the WOW screen designer and Cobol Source Editor.

Developer productivity has been enhanced by providing improved code navigation while editing in the IDE and debugging programs, and new debugger features have been implemented.

isCOBOL IDE-generated source code mimics the code you are familiar with from the RM/COBOL WOW Designer. COBOL developers have a choice between the WOW GUI and the isCOBOL GUI when designing new application features. WOW Screens and standard isCOBOL Screens can work together as long as they're handled by separate programs.

The isCOBOL runtime provides most of the WOW library routines to manage GUI widgets.

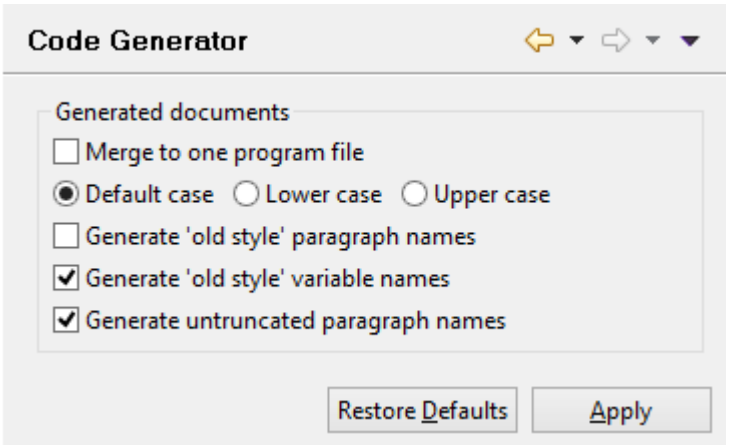
Cobol-WOW programs converted to isCOBOL can also run in thin client mode in an Application Server environment.

Before you start – Initial IDE configuration and creation of a new project

Before importing your Cobol-WOW projects in Veryant's IDE you should review the global preferences set.

1. Click on *Window* in the IDE menu bar and choose *Preferences*
2. Expand *isCOBOL* in the tree, then expand *WOW*

Code Generator

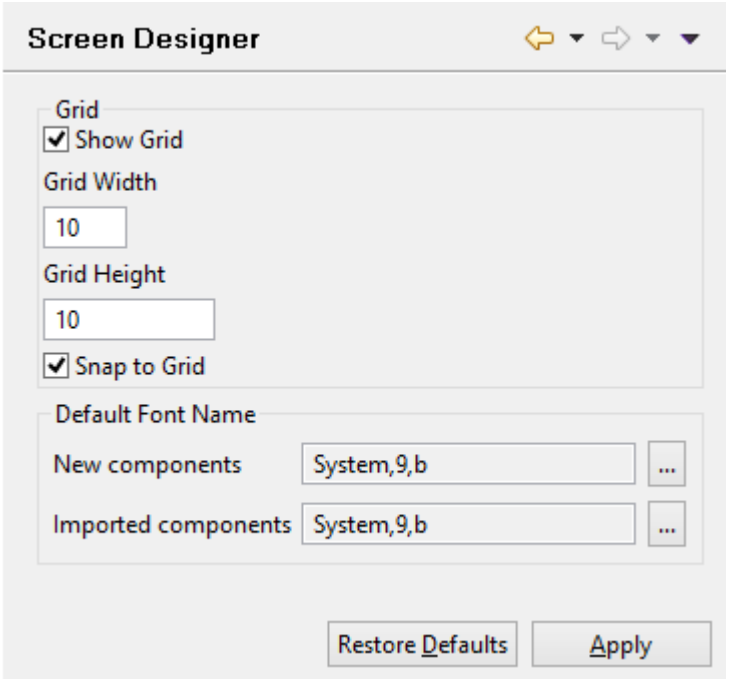


Here you can customize the way isCOBOL IDE generates the source code of your Cobol-WOW programs:

Merge to one program file	If checked, a single CBL file is generated. Otherwise, a CBL file and multiple copybooks are generated.
Default case | Lower case | Upper case	The case used for COBOL identifiers and keywords. By default, a lower case is used.
Generate 'old style' paragraph names	This setting reflects the same setting available in Cobol-WOW. [1]
Generate 'old style' variable names	This setting reflects the same setting available in Cobol-WOW. [1]
Generate untruncated paragraph names	This setting reflects the same setting available in Cobol-WOW. [1]

[1]These settings affect only new programs created in the isCOBOL IDE. Programs imported from existing Cobol-WOW projects maintain the settings they had in Cobol-WOW.

Screen Designer



Here you can configure the Screen Designer:

Show Grid	If checked, a grid is shown as background of the window in the Screen Designer. The width and height in pixels can be configured in the fields below. The grid helps in dimensioning controls when you draw them with the mouse.
Snap to Grid	If checked, the Screen Designer automatically aligns controls to the nearest cell. It helps in having controls aligned to each other.
Default font name	Specifies the font that is used by default for the new controls drawn in the Screen Designer as well as for the controls imported from existing Cobol-WOW projects if they lack font attributes.

Before importing Cobol-WOW projects you should create an empty isCOBOL Project in the IDE.

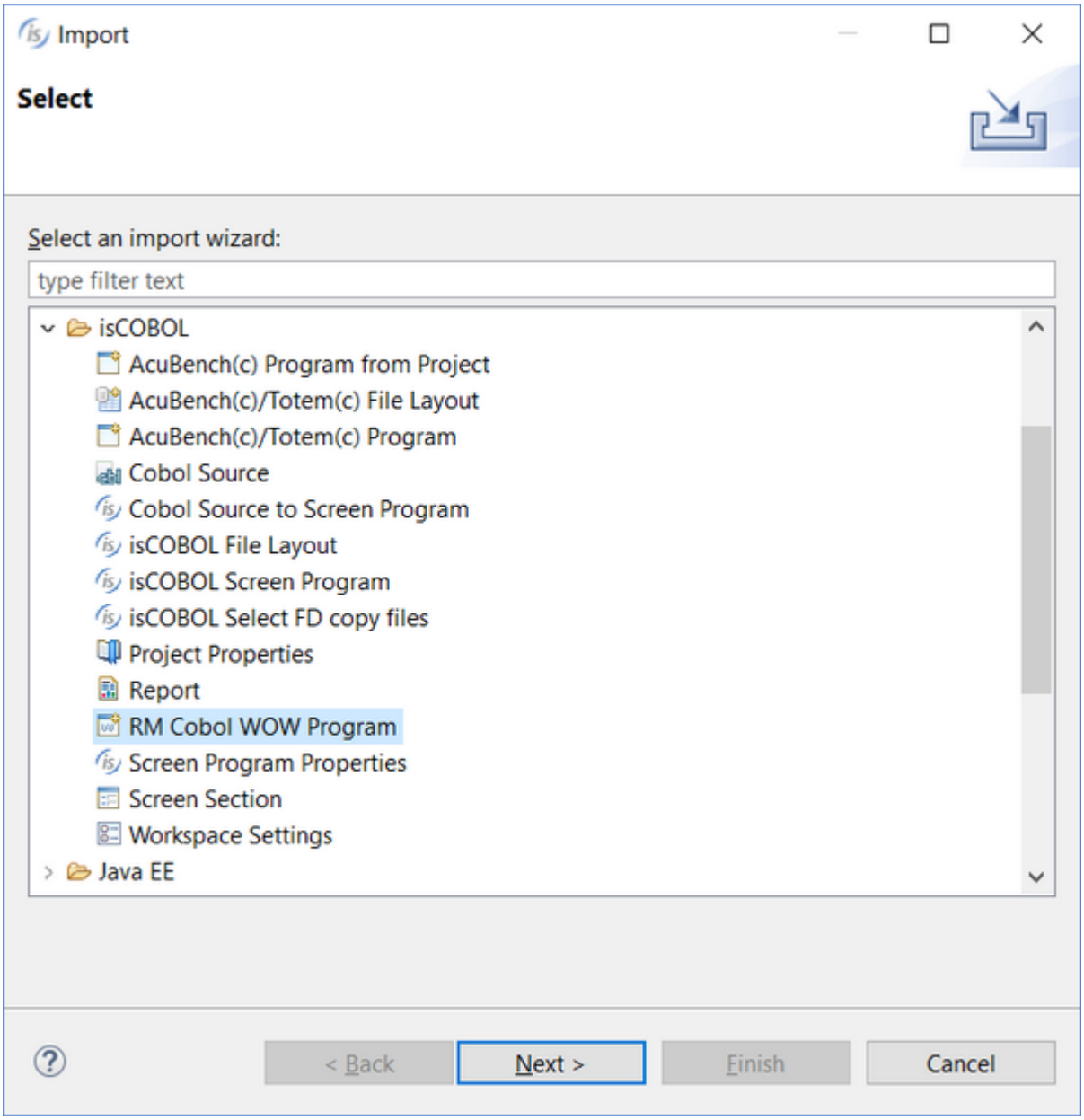
1. Click on *File* in the menu bar and choose *New*
2. Select *isCOBOL Project* from the sub menu
3. Follow the wizard procedures tothe end.

Importing projects from Cobol-WOW in isCOBOL IDE

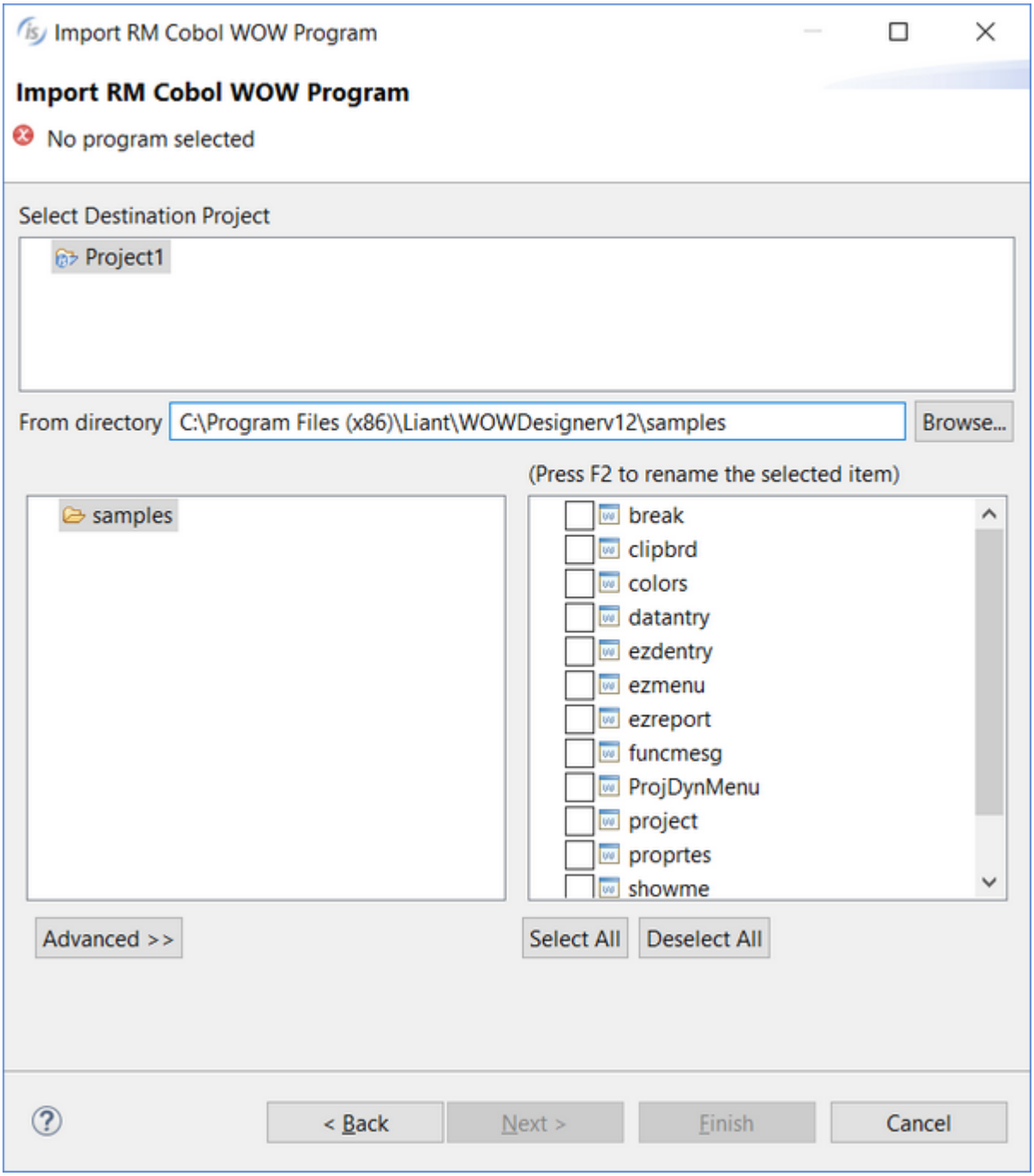
Veryant's IDE is able to import programs from existing Cobol-WOW projects.

In order to import projects from Cobol-WOW in the IDE:

1. Click on *File* in the menu bar and choose *Import*
2. Expand the *isCOBOL* group and select *RM Cobol WOW Program*



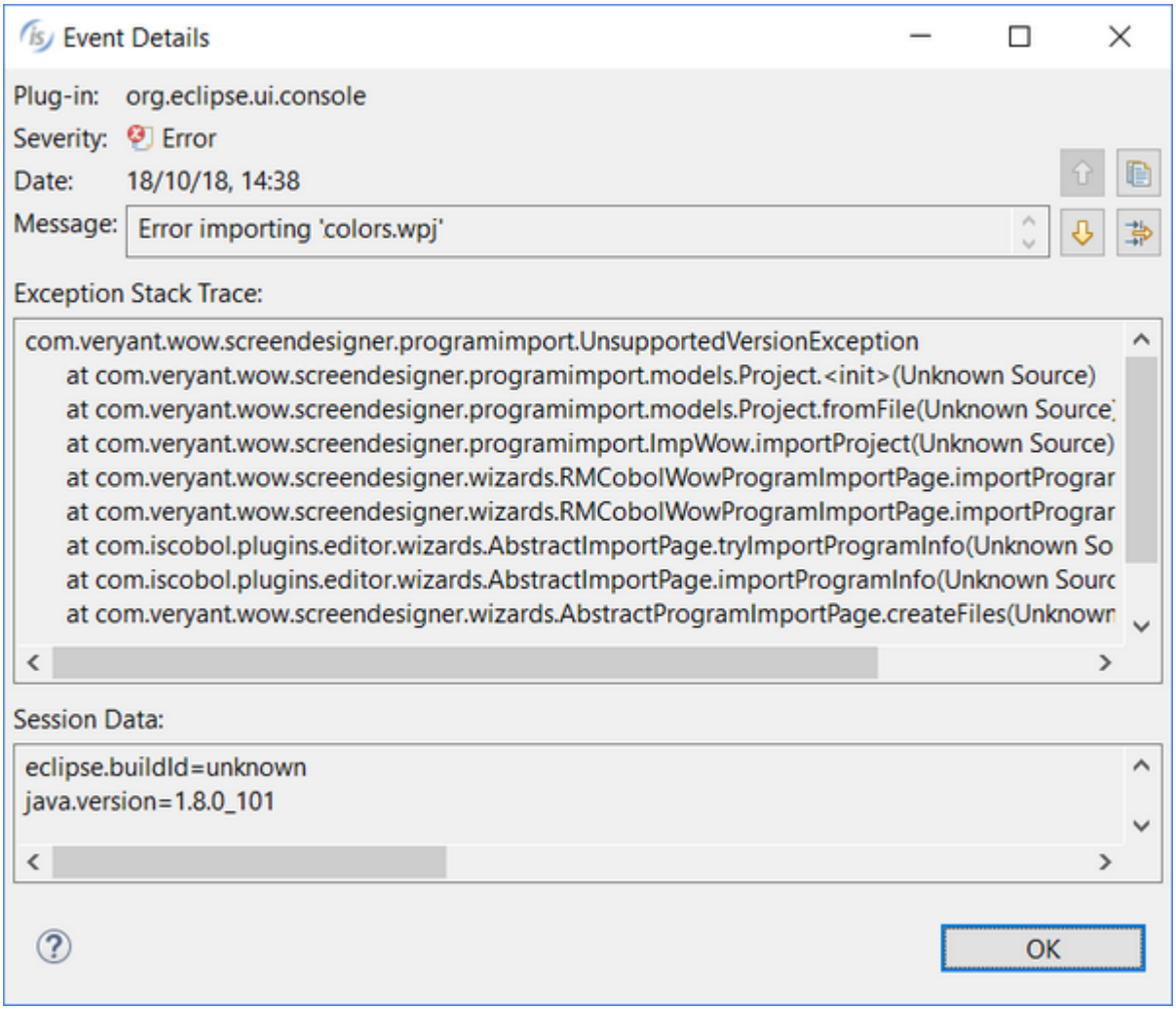
3. Browse for the folder where the Cobol-WOW projects are stored and press Enter to load the list of available programs:



The *Advanced* button allows you to choose which character set was used in the code of the Cobol-WOW projects. Use it if you notice wrongly decoded characters on the screen after the import.

4. Select the desired programs and click *Finish*.

Note that only projects of version 12 are supported. If you try to import a Cobol-WOW project whose version is unsupported, then the program doesn't appear in the IDE and the following exception is shown in the *Error Log* view:

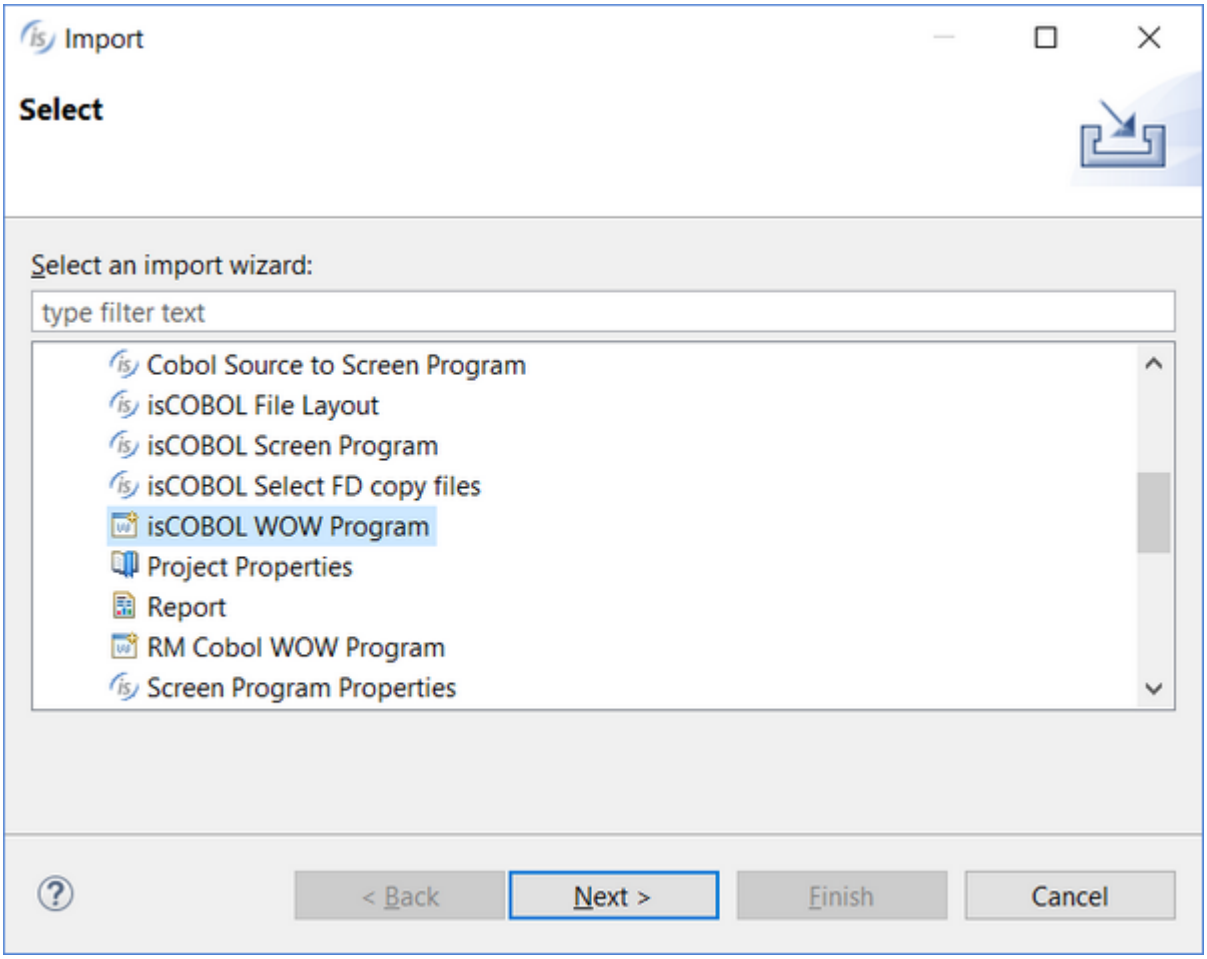


In order to get rid of the error, you can open the Cobol-WOW project with a Cobol-WOW designer version 12, make a small useless modification and save. In this way the project files will be compatible with isCOBOL IDE.

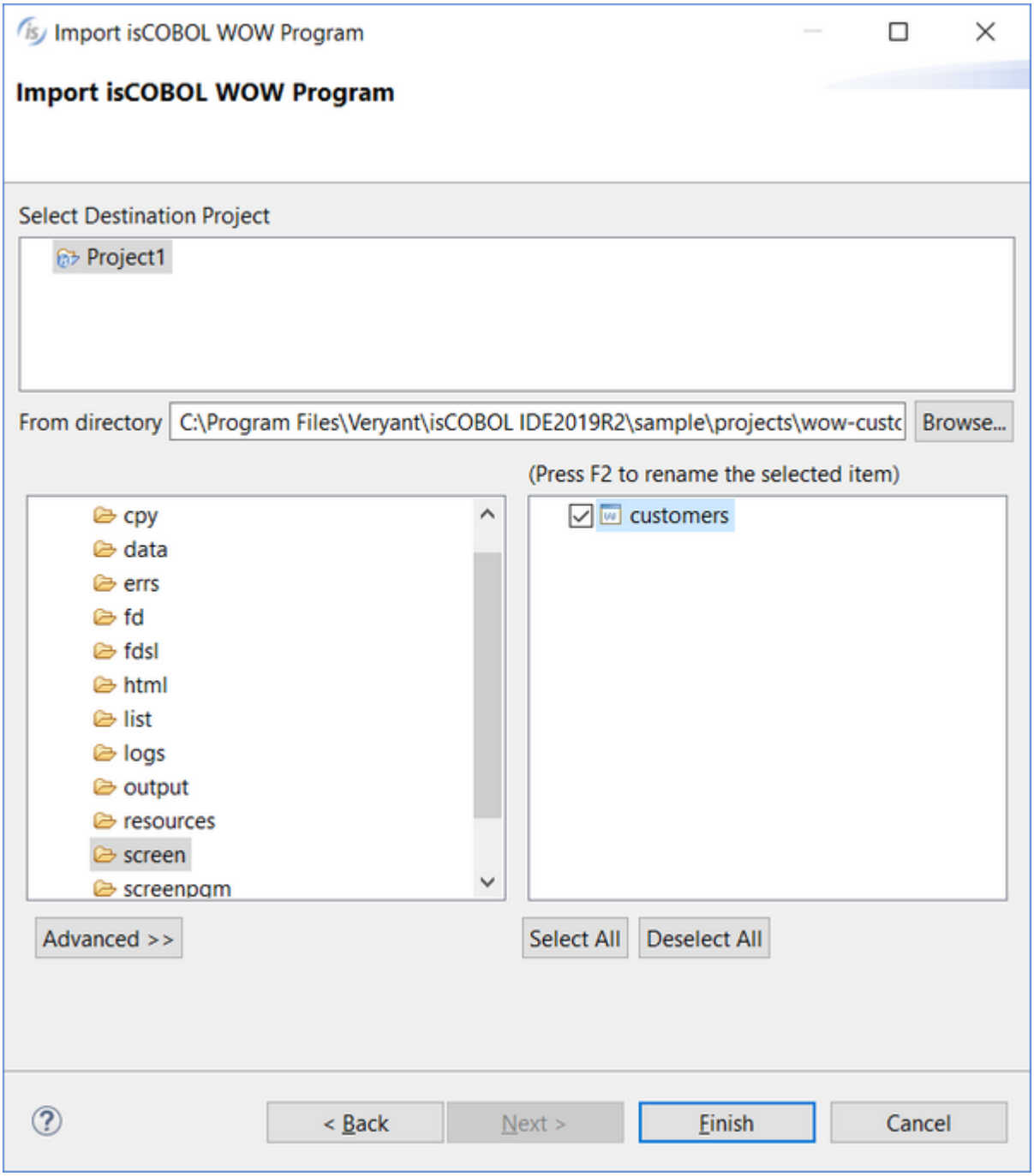
Adding existing WOW Programs to the current Project

isCOBOL IDE can also import an existing WOW Program that is part of another project or workspace into the current project.

In order to add existing WOW Programs to the current Project, right click on the project name in the isCOBOL Explorer and choose Import from the pop-up menu. Then select *isCOBOL / isCOBOL WOW Program* from the tree.



Use the *Browse* button to find the folder with the program that you wish to import. Select the desired item from the list on the right. Click on the *Advanced* button for advanced options.



Advanced options for WOW Program are:

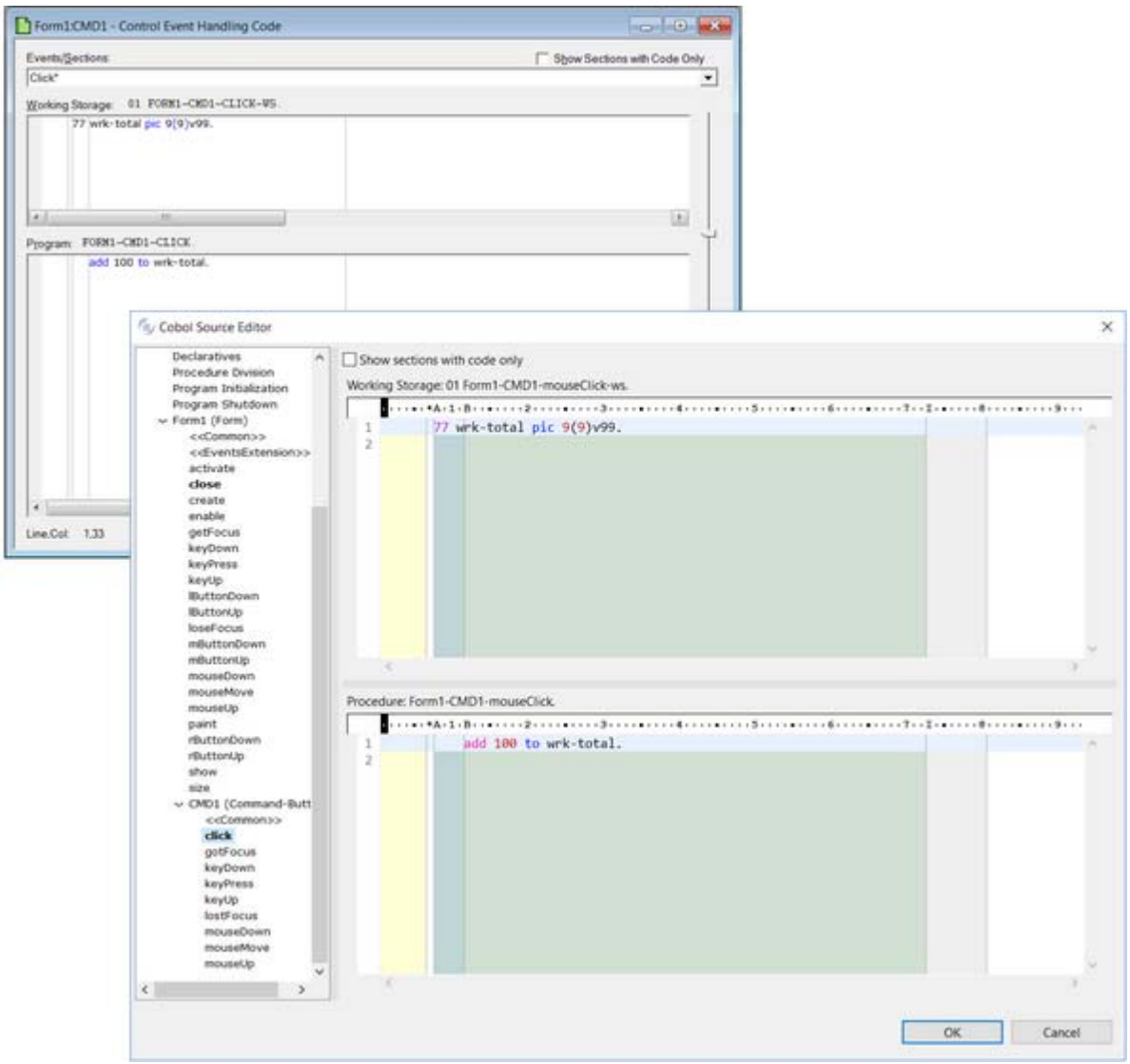
Create links in workspace	By default the imported resource is copied in the project folder and modifications are done on the copied resource, leaving the original resource unchanged. If you check this option instead, the resource is linked and modifications are done on the original resource.

Editing, compiling and running Cobol-WOW programs

Once the Cobol-WOW project has been imported in the IDE, the user can edit it in the same way he did with the Cobol-WOW Designer: draw controls by dragging them from the palette to the screen and edit the code by double clicking on the control.

The code editor in isCOBOL's IDE is slightly different from Cobol-WOW's Designer.

Cobol-WOW allows you to choose the section from a combo box, while isCOBOL IDE shows all the sections in a tree view on the left:



In order to generate the source code of your programs:

1. Right click on the program name in the Project Explorer's Structural View
2. Select *Generate Program* from the menu

Or

1. Left click on the program name in the Project Explorer's Structural View
2. Click on the *Generate Program* button in the IDE's tool-bar

In order to compile a program:

1. Right click on the program name in the Project Explorer's Structural View
2. Select *Compile* from the menu

Or

1. Left click on the program name in the Project Explorer's Structural View
2. Click on the *Compile* button in the IDE's tool-bar

Note that, if you compile a program without generating it first, the IDE generates it automatically before compiling.

In order to execute a program:

1. Right click on the program name in the Project Explorer's Structural View
2. Choose *Run As* from the menu
3. Choose *isCOBOL Application* from the sub menu[1]

Or

1. Left click on the program name in the Project Explorer's Structural View
2. Click on the *Run* button in the IDE's tool-bar

[1]Note that Cobol-WOW programs can't run in webDirect environment. If you select *isCOBOL EIS webDirect* from the *Run As* menu, the effects are unpredictable.

In order to debug a program:

1. Right click on the program name in the Project Explorer's Structural View
2. Choose *Debug As* from the menu
3. Choose *isCOBOL Application* from the sub menu[1]

Or

1. Left click on the program name in the Project Explorer's Structural View
2. Click on the *Debug* button in the IDE's tool-bar

[1]Note that Cobol-WOW programs can't run in webDirect environment. If you select *isCOBOL EIS webDirect* from the *Debug As* menu, the effects are unpredictable.

The isCOBOL Debugger is different than the one in Cobol-WOW. Refer to Debugger for details about the usage of the isCOBOL Debugger.

Note that pressing the *Pause* button on the keyboard while debugging a WOW program doesn't enter the debugger like it does with standard GUI COBOL programs. In order to enter the debugger, you need to click on the *Pause* button in the Debugger tool-bar.

Known differences and things to know

Concepts

Cobol-WOW project management is very simple: all files are generated in the same folder, there is no structure. There is a project for each GUI program. Each project references one or more forms and menus.

The Eclipse-based isCOBOL IDE from Veryant instead provides a structured management of the projects. There is a workspace, a disk folder where project files are physically saved. The workspace can host one or more projects. Each project can host one or more programs. All the program items (e.g. form and menu) are included in the program instead of being referenced as separate files.

	Cobol-WOW	**isCOBOL IDE**
Programs container	Disk folder	Project
Program entity	Project	Program
Form entity	Form	n/a[1]
Menu entity	Menu	n/a[1]

[1]The form and the menu are included in the program.

Project files

Veryant's IDE uses different file formats and extensions for the project's file.

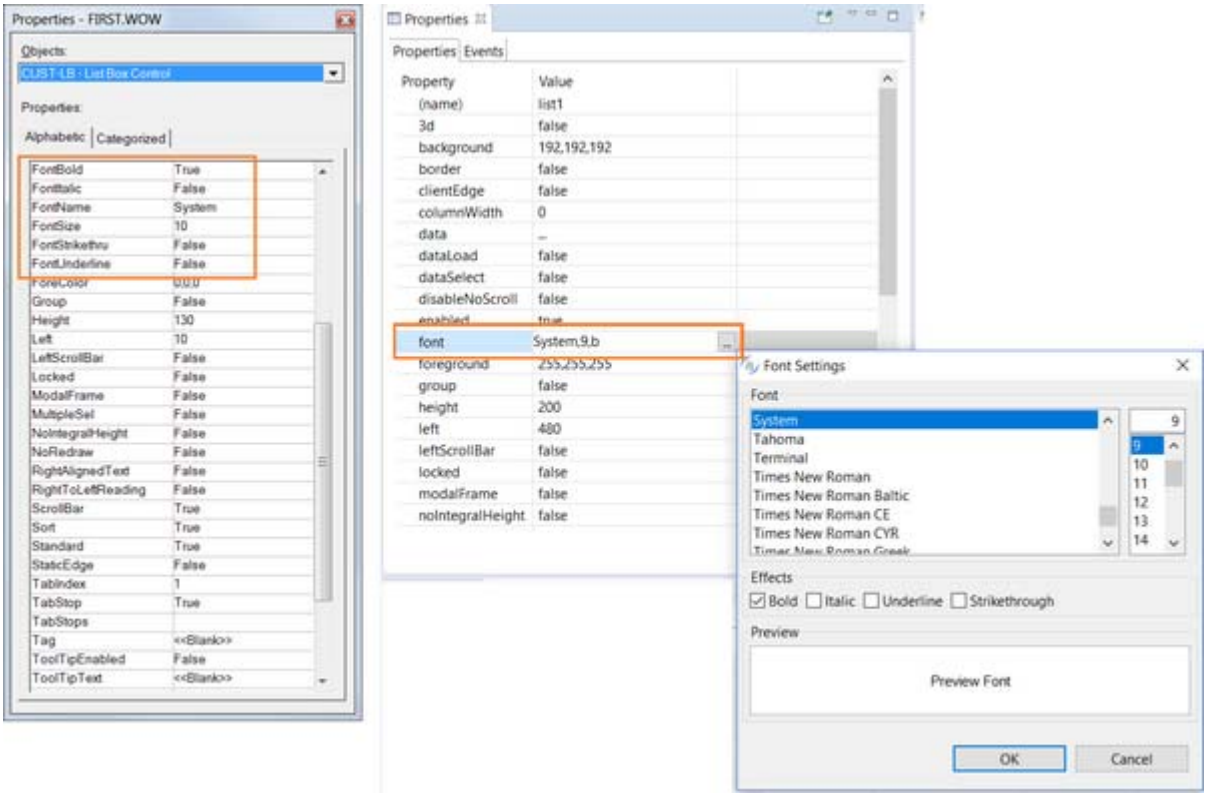
	Cobol-WOW	**isCOBOL IDE**
Program file extension	Wjp	wsp
Form file extension	Wow	n/a[1]
Menu file extension	wmn	n/a[1]

[1]The information is included in the wsp file.

Screen Designer

There are some differences between Veryant's IDE screen painter and Cobol-WOW's screen designer.

The main difference is in the list of attributes of controls. When you select a graphical control on the screen, the list of attributes is populated in the Properties view. Cobol-WOW shows several attributes while Veryant's IDE tries to group them where applicable. For example, for most of the controls you can set the font. The screenshot below shows a comparison between the multiple font properties shown by Cobol-WOW (on the left) and the single font property shown by Veryant's IDE (on the right).



List-Box and Status-Bar have properties to define the multiple items in WOW designer. In Veryant's IDE instead there is an item designer that shows a pop-up dialog in which the user can configure the items contained in the control.

Bitmaps

When you run a WOW program from Veryant's IDE, bitmaps are loaded from the working directory and from the Java Classpath.

Form and menu sharing

In Cobol-WOW forms and menus are identified by separate files and they're referenced by the Cobol-WOW project. This means that multiple projects can reference the same form or the same menu and altering such form or menu affects all the projects in which the form or menu is referenced.

The same concept is not applicable to the isCOBOL IDE. The IDE keeps the information about form and menu inside the program file. If you edit a form or a menu, only the program you're editing is affected. However, when the source code is generated, a copybook with the same name of the form is generated on disk, so if multiple programs include a form with the same name, an overwrite of the copybook may occur. The same happens with menus.

Menu editor

The maintenance of the menu bar of the form in Cobol-WOW is done via the menu designer, activated by a dedicated button in the Designer tool-bar. In isCOBOL IDE instead the menu editor is activated by editing the menu property of the form.

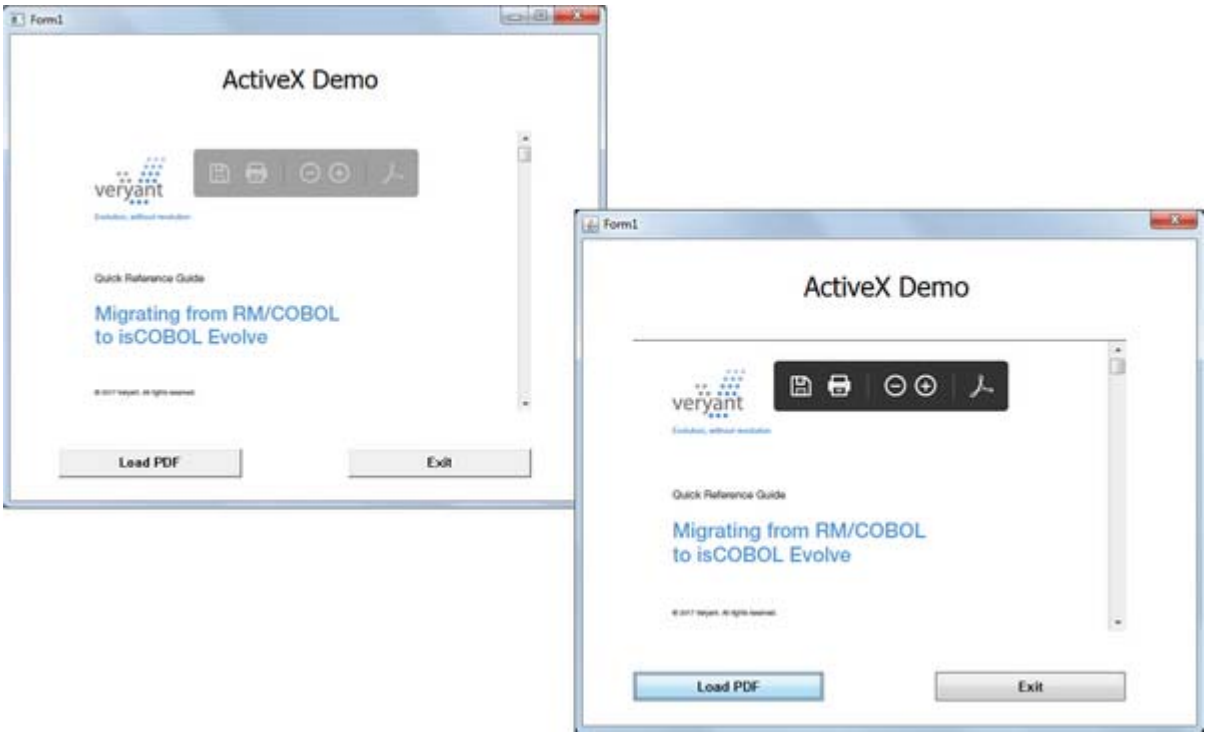
ActiveX

The ActiveX technology is in legacy status and is no longer supported by Microsoft.

Veryant's approach on ActiveX is to provide tools and migration service to re-implement all ActiveX controls as Java components

isCOBOL IDE's import wizard reads the ActiveX resources used in a WOW project and imports all its properties with the correct values. The IDE relies on the isCOBOL utility AxTools to generate a Java template for each ActiveX control to be implemented. AxTools reads all properties, events and methods available from ActiveX and OCX controls and generates a Java source skeleton that users can modify in order to implement the properties, methods and events of the ActiveX in Java.

The screenshot below shows how the IExplorer ActiveX used in a Cobol-WOW form (on the left) was reimplemented using the Java component DJBrowser (on the right):



Veryant will provide some source code examples of ActiveX controls from DBI-Technologies rewritten in Java.

Contact Veryant for more information about ActiveX replacement.

Supported library routines

isCOBOL supports the following library routines of Cobol-WOW:

- AXBINDEVENTARGUMENTS
- AXDOMETHOD
- AXUNBINDEVENTARGUMENTS
- CHECKMENUITEM
- CLOSEWINDOW
- DELETEMENU
- DRAWMENUBAR
- ENABLEMENUITEM
- ENABLEWINDOW
- FINDWINDOW
- GETACTIVEWINDOW
- GETCURSORPOS
- GETENVIRONMENTVARIABLE
- GETFOCUS
- GETMENU
- GETSUBMENU
- GETWINDOWSDIRECTORY
- ISCHILD
- ISWINDOW
- MESSAGEBEEP
- MESSAGEBOX
- MODIFYMENU
- OPENICON
- SENDMESSAGE
- SETACTIVEWINDOW
- SETFOCUS
- SHOWWINDOW

- WINEXEC
- WOWADDITEM
- WOWCLEAR
- WOWCLEARWAITCURSOR
- WOWCREATEWINDOW
- WOWDESTROYWINDOW
- WOWDISCARDEVENTS
- WOWGETFOCUS
- WOWGETINDEXPROP
- WOWGETMESSAGE
- WOWGETNUM
- WOWGETPROP
- WOWGETWINDOWTYPE
- WOWINITALLCONTROLS
- WOWINITCONTROL
- WOWMESSAGEBOX
- WOWMOVE
- WOWMULTICONTROLGETPROP
- WOWMULTICONTROLSETPROP
- WOWPEEKMESSAGE
- WOWREFRESH
- WOWREMOVEITEM
- WOWRESETWAITCURSOR
- WOWSETFOCUS
- WOWSETINDEXPROP
- WOWSETNEXTCTRL
- WOWSETNUM
- WOWSETPREVCTRL
- WOWSETPROP
- WOWSETSTRIPTRAILING

- WOWSETWAITCURSOR
- WOWVERSION[1]

[1]WOWVERSION always returns 0.

isCOBOL also provides a replacement for the following Windows API invoked by Cobol-WOW:

- GetNextDlgTabItem
- GetWindow

Transitioning from other COBOLs

The isCOBOL Compiler and Framework have been proven to compile and run successfully source code from the following COBOLs that were not discussed in the previous transitioning guides:

COBOL	Advice
Fujitsu NetCOBOL	
Hitachi's COBOL	
HP COBOL	
LPI-COBOL	This COBOL is very similar to RM/COBOL, so you should consider following the advices provided in the Transitioning from RM/COBOL guide.
OpenCOBOL/GnuCOBOL	
VMS COBOL	
COBOL-IT	This COBOL is very similar to MicroFocus, so you should consider following the advices provided in the Transitioning from MicroFocus guide.

Data Access

- Fixed length sequential and relative files coming from the above COBOLs are fully supported by isCOBOL.
- Indexed files from the above COBOLs must be converted either to c-tree or JIsam with the following steps:
  a. unload file data to a raw binary file with the tools provided by the other COBOL vendor
  b. create an empty c-tree or JIsam indexed file by running a program that performs opens the file for output with isCOBOL
  c. load data from the raw binary file into the empty indexed file with ctutil or JUTIL.

Transitioning from Pro*COBOL

The following guide describes the key areas which need to be addressed during the replacement of Pro*COBOL with JDBC.

Why replace Pro*COBOL?

Pro*COBOL is a precompiler that translates Embedded SQL code into calls to dedicated API functions written in C.

Since isCOBOL is able to call C functions, you could consider maintaining Pro*COBOL and using the isCOBOL Compiler to compile the translated source file (program.cbl) instead of the original source file (program.pco).

However, with isCOBOL you can execute Embedded SQL statements via JDBC, which is more natural with the Java core of isCOBOL.

Here is a list of advantages to discarding the Pro*COBOL precompiler in favor of a direct support of Embedded SQL via JDBC:

One step compilation

The isCOBOL Compiler understands the Embedded SQL code without the need of precompiling. You can directly compile the original source file (program.pco).

Clearer debug

Since no code translation occurred, the isCOBOL Graphical Debugger shows the Embedded SQL code exactly as you wrote it, allowing you to monitor the host variables used in the EXEC SQL blocks.

Simpler installation

Pro*COBOL requires the Oracle Client installed on the machine where the program runs. The Oracle Client includes native libraries and therefore it must be of the same architecture of the runtime system (e.g. a 64 bit isCOBOL requires a 64 bit Oracle client).

When working with JDBC instead, you just need the JDBC driver library. This is usually a single jar file written in Java without native dependencies, so it is portable on every system and architecture.

Thread safety

Pro*COBOL is not thread safe so it's not advisable to use it in application server environments like the isCOBOL Thin Client or Tomcat.

JDBC drivers instead are thread safe and therefore suitable to be used in application server environments.

Portability to other RDBMS

If the COBOL application uses standard SQL rather than Oracle specific SQL syntax, you can easily run it on other RDBMS (for example MySQL, PostgreSQL or Microsoft SQL Server). Just add the proper JDBC driver library to the Classpath and adjust the value of iscobol.jdbc.driver and iscobol.jdbc.url configuration properties.

Programs precompiled with Pro*COBOL instead are suitable only for the Oracle database.

Issues you should be aware of

The isCOBOL SQL support based on JDBC has some differences with Pro*COBOL. Below you find a list of the changes required to the configuration in order to obtain the same behavior between Pro*COBOL and isCOBOL on JDBC.

Required items and configuration

Pro*COBOL relies on the Oracle Client that must be installed on the same computer where the runtime is executed. The information for the database connection (such as database IP address and port) is retrieved from the configuration files of the Oracle Client.

isCOBOL instead creates a JDBC connection. There is no need of the Oracle Client, you just need the Oracle JDBC driver library (e.g. *ojdbc7.jar*). You can find JDBC driver libraries at https://www.oracle.com/database/technologies/appdev/jdbc-downloads.html.

These steps are required:

1. Add the Oracle JDBC driver to the Classpath.

Example for Windows:

```
set CLASSPATH=%CLASSPATH%;\path\to\ojdbc7.jar
```

Example for Linux/Unix:

```
export CLASSPATH=$CLASSPATH:/path/to/ojdbc7.jar
```

2. Define the JDBC driver class in the runtime configuration:

```
iscobol.jdbc.driver=oracle.jdbc.OracleDriver
```

3. Define the connection URL in the runtime configuration. Example:

```
iscobol.jdbc.url=jdbc:oracle:thin:scott/tiger@192.168.0.251:1521:
```

Negative SQLCODE

Pro*COBOL returns the negative version of the Oracle error number in the SQLCODE data item. By default isCOBOL returns the absolute value of the Oracle error number instead. In order to have negative error codes, add the following entry to the configuration:

```
iscobol.esql.error.negative=true
```

SQLSTATE

Pro*COBOL doesn't define SQLSTATE in the SQLCA structure, like isCOBOL does, but it lets you declare SQLSTATE among host variables. Therefore, in order to run without issues with isCOBOL, it is necessary to comment or remove the SQLSTATE declaration from the source code and refer to the SQLSTATE data item defined in the SQLCA structure.

Different SQLCODE for NOTFOUND

Unless you used MODE=ANSI in your Pro*COBOL options, the SQLCODE returned by Pro*COBOL for the NOTFOUND condition is 1403. By defeault isCOBOL sets SQLCODE=100 for the NOTFOUND condition. If you wish to have 1403, add the following entry to the configuration:

```
iscobol.esql.sqlcode.100=1403
```

Error for too many rows

When a SELECT query is performed directly, without using cursors, the program expects to read only one record. Pro*COBOL returns an error if more than one record is returned (SQLCODE=-2112, SQLERRMC="SQL-02112: SELECT..INTO returns too many rows"). isCOBOL by default doesn't return any error and sets the host variables of the INTO clause with the content of the first record in the resultset. In order to have an error also with isCOBOL, add the following entry to the configuration:

```
iscobol.esql.value_too_many_rows=-2212
```

Error for NULL value fetched

Unless you used UNSAFE_NULL=YES in your Pro*COBOL options, when a NULL value is fetched and no indicator variable was used, Pro*COBOL returns an error (SQLCODE=-1405, SQLERRMC="ORA-01405: fetched column value is NULL"). By default isCOBOL doesn't return errors in this case, it just sets the host variable to zero or spaces depending on the picture. In order to have the error number 1405 also with isCOBOL, compile the program with the -csqn option.

Error for no data during DELETE, INSERT and UPDATE

Pro*COBOL returns a NOT_FOUND condition (e.g. it sets SQLCODE either to 100 or 1043, depending on the mode) when an UPDATE statement or a DELETE statement doesn't affect any record. The same NOT_FOUND condition is returned for INSERT INTO SELECT statements, when the inner SELECT doesn't find any record.

isCOBOL doesn't do the same, by default.

In order to have the same behavior of Pro*COBOL with isCOBOL, set iscobol.esql.value_sqlcode_on_no_data in the configuration.

Mapping the PICX option

Pro*COBOL has an option named PICX that specifies the default datatype of PIC X variables.

The possible values for this option are CHARF and VARCHAR2. The default value is CHARF (was VARCHAR2 before release 8.0).

To have the same behavior of PICX=CHARF with isCOBOL, add these entries to the configuration:

```
iscobol.jdbc.kept_spaces=-1
iscobol.jdbc.options=fixedString=true
```

To have the same behavior of PICX=VARCHAR2 with isCOBOL, add these entries to the configuration:

```
iscobol.jdbc.kept_spaces=0
iscobol.jdbc.options=fixedString=true
```

Issues with Stored Procedures

When a data truncation occurs while setting an host variable to the output parameter of a stored procedure, Pro*COBOL doesn't report it in the indicator variable. In order to have the same behavior with isCOBOL, add the following entry to the configuration:

```
iscobol.esql.indicator_trunc_on_call=false
```

Due to limitations in the JDBC API, isCOBOL is not able to understand the stored procedure signature, so it sets parameters in a generic way. It works fine in most of the cases, but in some cases Oracle may reject the parameters because they're not of the expected type. If you experience errors related to a wrong parameter type, consider to describe the stored procedure signature. It can be done in the Compiler configuration by setting iscobol.compiler.esql.procedure.ProcedureName or in the COBOL program by using the HOSTVAR Directive.

Using Pro*COBOL from isCOBOL

As stated in the introduction, isCOBOL is able to compile and run programs precompiled by Pro*COBOL. If you wish to maintain Pro*COBOL instead of moving to JDBC, follow these steps:

1. Ensure that the architecture of isCOBOL matches with the architecture of Pro*COBOL. They must be both 32 bit or both 64 bit.
2. Compile the precompiled source (program.cbl) with the -cp option.

Transitioning from the IBM DB2 Preprocessor

The following guide describes the key areas which need to be addressed during the replacement of the IBM DB2 preprocessor with JDBC.

Why replace the IBM DB2 preprocessor?

The IBM DB2 preprocessor translates Embedded SQL code into calls to dedicated API functions written in C.

Since isCOBOL is able to call C functions, you could consider maintaining this approach and using the isCOBOL Compiler to compile the translated source file (program.cbl) instead of the original source file (program.sqb).

However, with isCOBOL you can execute Embedded SQL statements via JDBC, which is more natural with the Java core of isCOBOL.

Here is a list of advantages to discarding the IBM DB2 preprocessor in favor of a direct support of Embedded SQL via JDBC:

One step compilation

The isCOBOL Compiler understands the Embedded SQL code without the need of precompiling. You can directly compile the original source file (program.sqb).

Clearer debug

Since no code translation occurred, the isCOBOL Graphical Debugger shows the Embedded SQL code exactly as you wrote it, allowing you to monitor the host variables used in the EXEC SQL blocks.

Simpler installation

Programs generated by the IBM DB2 preprocessor require the IBM DB2 client libraries installed on the machine where the program runs. These libraries are native libraries and therefore they must be of the same architecture of the runtime system (e.g. a 64 bit isCOBOL requires a 64 bit DB2 client libraries).

When working with JDBC instead, you just need the JDBC driver library. This is usually a single jar file written in Java without native dependencies, so it is portable on every system and architecture.

Thread safety

Programs generated by the IBM DB2 preprocessor are not thread safe so it's not advisable to use them in application server environments like the isCOBOL Thin Client or Tomcat.

JDBC drivers instead are thread safe and therefore suitable to be used in application server environments.

Portability to other RDBMS

If the COBOL application uses standard SQL rather than DB2 specific SQL syntax, you can easily run it on other RDBMS (for example MySQL, PostgreSQL or Microsoft SQL Server). Just add the proper JDBC driver library to the Classpath and adjust the value of iscobol.jdbc.driver and iscobol.jdbc.url configuration properties.

Programs generated by the IBM DB2 preprocessor instead are suitable only for the IBM DB2 database.

Issues you should be aware of

The isCOBOL SQL support based on JDBC has some differences with the IBM DB2 preprocessor. Below you find a list of the changes required to the configuration in order to obtain the same behavior between the IBM DB2 preprocessor and isCOBOL on JDBC.

Enabling the DB2 compatibility mode

For the best compatibility with the IBM DB2 rules and behaviors, you should enable the DB2 compatibility mode in the Compiler before compiling your programs with ESQL. In order to enable the DB2 compatibility mode, use the following setting in the Compiler configuration:

```
iscobol.compiler.esql.db2=1
```

When this property is set to true, the Compiler generates specific code to return the result sets in the same format that would be produced when using the IBM DB2 preprocessor. In particular, it supports the SQLDA structure and the use of date, time and timestamp as function parameters, like for example DAYOFWEEK or WEEKS_BETWEEN.

JDBC Connection

isCOBOL creates a JDBC connection. There is no need of native DB2 client libraries; you just need the IBM DB2 JDBC driver library (e.g. *db2jcc4.jar.jar*). You can find JDBC driver libraries at https://www.ibm.com/support/pages/db2-jdbc-driver-versions-and-downloads.

These setup steps are required:

1. Add the IBM DB2 JDBC driver to the Classpath.

Example for Windows:

```
set CLASSPATH=%CLASSPATH%;\path\to\db2jcc4.jar
```

Example for Linux/Unix:

```
export CLASSPATH=$CLASSPATH:/path/to/db2jcc4.jar
```

2. Define the JDBC driver class in the runtime configuration:

```
iscobol.jdbc.driver=com.ibm.db2.jcc.DB2Driver
```

3. Define the connection URL in the runtime configuration. Example:

```
iscobol.jdbc.url=jdbc:db2://localhost:50000/SAMPLE
```

Different format string from DATE, TIME and TIMESTAMP

The format string used in JDBC to represent the value of DATE, TIME and TIMESTAMP columns may be different than the format string used when working with the IBM DB2 preprocessor. In order to make isCOBOL use the same format strings as the IBM DB2 preprocessor, you can set the following runtime configuration properties:

- iscobol.jdbc.dateformat *
- iscobol.jdbc.timeformat *
- iscobol.jdbc.timestampformat *

Error for NULL value fetched

When a NULL value is fetched and no indicator variable was used, you expect the error code -305.

By default isCOBOL doesn't return errors in this case, it just sets the host variable to zero or spaces depending on the picture. In order to have the error number -305 also with isCOBOL

1. compile the program with the -csqn option,
2. add *iscobol.esql.value_sqlcode_on_null=-305* to the runtime configuration.

Using programs generated by the IBM DB2 preprocessor with isCOBOL

As stated in the introduction, isCOBOL is able to compile and run programs generated by the IBM DB2 preprocessor. If you wish to maintain the DB2 preprocessor instead of moving to JDBC, follow these steps:

1. Ensure that the architecture of isCOBOL matches with the architecture of the DB2 native client libraries. They must be both 32 bit or both 64 bit.
2. Compile the source file generated by the preprocessor (program.cbl) with the -cp option.