2020/11

ISSUE 03

TRIANNUAL JOURNAL OF VERYANT AND isCOBOL

veryant NEWS

THIS ISSUE

AT VERYANT, WE CARE ABOUT MAKING YOUR LIFE EASIER.

This issue highlights 2020R2, with lots of ways to work smarter, including

- An easier Profiler,
- A slimmed-down Debugger
- A call to arms to use the isCOBOL IDE.

cobol
Cool Business Oriented Language

Our thoughts are with all who have been impacted by the coronavirus.

WHAT'S NEW

2020R2 is here

Veryant is pleased to announce the latest release of isCOBOL™ Evolve, isCOBOL Evolve 2020 Release 2.

Each release is exciting to us, because we get to show our customers that we listen to them, and that we're working hard to make our products richer, easier, and more relevant to current trends. This release is no exception.

2020R2 makes profiling your application easier, both to run and to fine-tune to your needs. Debugging is easier and cleaner – see the article entitled "Saying Farewell to iscobol.debug.code_prefix" below.

The IDE has more integrated features, so you don't have to leave it for profiling, code coverage or unit testing. You asked to be able to modify your variables outside of the graphical editors, and this version we've done that. Don't use our IDE? See the reasons why you should in our article by Davide Spizzi, Veryant's Support Engineering Manager.

Graphical changes like a hamburger menu, creating bitmaps using glyph symbols, a new web browser class, and compiler performance have all been added in 2020R2. Read more here, and look for an upcoming video demonstration on our YouTube channel.

DOCUMENTATION HIGHLIGHTS

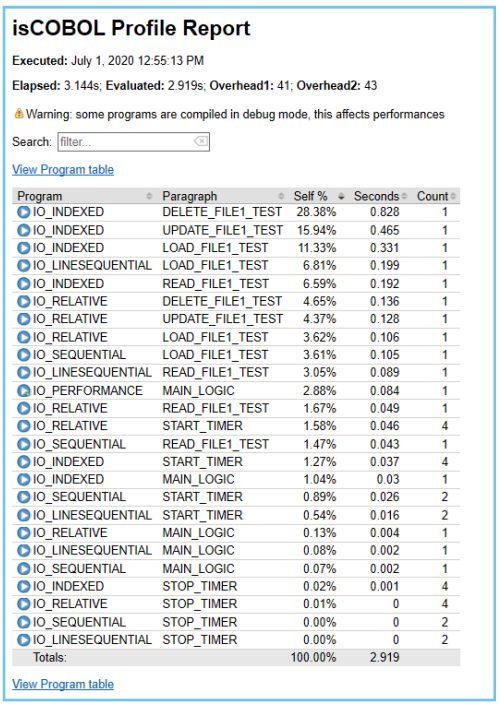
Performance Tuning

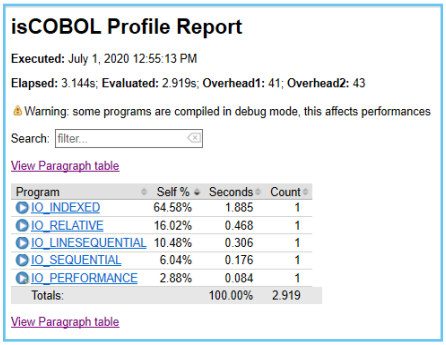
We've organized all our performance improvement suggestions in one place to make it easy for you to fine-tune your application's speed.
In the Appendices of the isCOBOL Evolve documentation is a book called Performance Tuning.
There are three chapters, "Guidelines for faster compilation", "Guidelines for better runtime performance", and "Profiling COBOL programs".

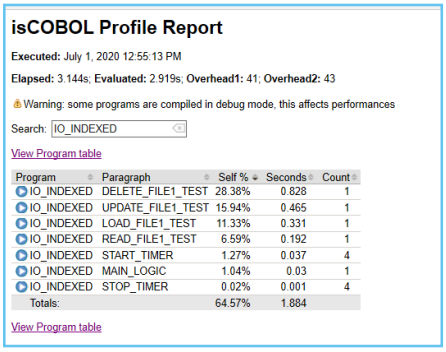
You'll find suggestions for faster compiling from the command line and the IDE, optimizations during compiling and running your application, and suggestions for Thin Client, Data access and printing performance improvement.

The Profiling chapter has some significant changes in 2020R2 – read more about it on this page.

Meet the new isCOBOL Profiler







2020R2 includes improvements to the isCOBOL Profiler, making the report easier to create and view. Running your programs with the "-profiler" switch will create an HTML report showing the time and percentage of total time for each program as well as each paragraph in the programs.

New options have been added to help you get a better report, including: C$PROFILER library -to turn the profiler on and off programmatically, especially around ACCEPT statements. This removes the time the user takes act. "Iscobol.profiler.excludes" and "iscobol. profiler.includes" – To restrict the classes profiled. This is useful if your UI is separate, or you have pinpointed a problem to one program.
The report shows a warning if the programs profiled were compiled in debug. You should only profile programs NOT compiled for debug.

Have You Seen This?

New YouTube videos:

New Features of isCOBOL Version 2020R1

isCOBOL for the Technologist

Modernization Methods with isCOBOL

isCOBOL and REST Services Demo

Mainframe Rehosting with HTWC

isCOBOL and zOS

New Knowledgebase articles:

How do I point my installed isCOBOL to a different Java location?

How to extend the length of AREA B in a program source code

How to run batch programs on a Linux server and compile and debug them from the IDE

How can I control or restrict access to my WebClient applications and see the user name displayed in the WebClient console

How to copy data from the clipboard

COBOL and our Universities

Marilyn Prince, Sales Engineer for North America, talks about the need for COBOL instruction in our Universities, how to keep students interested, and what Veryant is doing to help.

The shortage of COBOL programmers has been in the news a lot lately as these developers are aging out of the workforce. COBOL is taught in very few universities, and often as an elective. But COBOL code is still running our banks, insurance companies, and governments. To fix the problem, we need to look at why there's a huge gap between the need for COBOL programmers and the lack of priority for training them.

It could be because most COBOL programs were expected to be replaced by the year 2000 (leading to the Y2K problem). Or perhaps as COBOL programs purr merrily along, management gets complacent about the need for change – the old "if it isn't broken, don't fix it" maxim. Perhaps universities – for-profit companies subject to the whims of their student body – found COBOL was passé among their students, going the way of bell-bottoms and Beatlemania.

The answer is probably a combination of all three of these. But we've proven that COBOL is still here and probably here to stay in some form or another. We need not only more young COBOL programmers, but COBOL programmers that can bring our applications into the 21st century.

COBOL programmers can no longer get away with just knowing COBOL. For instance, COBOL and Java are being used together more frequently, in part because recent IBM enhancements allow COBOL and Java to run together on mainframes. The two languages fit well together, which is why isCOBOL is written in Java and compiles COBOL code to Java classes.

The capabilities of COBOL as REST or SOAP web services mean COBOL can easily continue to work as a data processing language, while newer languages can provide flashier interfaces that easily display data in screens with graphs, charts, and dashboards. Even a strictly COBOL application can be enhanced with OO Programming and the inclusion of JavaBeans, for instance. It's common to see a COBOL implementation with other languages or tools mixed in.

A commenter on Hacker News noted that a COBOL programmer spends their career "doing maintenance work rather than any greenfield development". That might be true in some cases, but to move into the future COBOL programs will have to evolve, and new development is a part of that – whether it takes the form of COBOL graphical screens, object-oriented COBOL programming, or partitioning COBOL into web services.

But how to we get students interested enough to pay universities to teach COBOL? New students are alienated by the "green screen" development environment, but why not try an IDE? I spoke recently with Joel Sweatte at the East Carolina University about teaching COBOL with isCOBOL's IDE. Veryant is happy to provide his students with free licenses for his class. He says using the Eclipse-based environment for his first-time programmers not only makes learning easier, it elevates COBOL to the same level as other languages they want to work on, such as Java and C++.

Veryant is doing our part to address the shortage of COBOL programmers by offering free licenses to universities

It's not just students who prefer a graphical development environment. We're seeing an increased interest in the isCOBOL IDE. There are even some mainframe developers that use the IDE to develop their COBOL programs, then move them to the mainframe to compile and run.

Veryant is doing our part to address the shortage of COBOL programmers by offering free temporary licenses to universities, like the East Carolina University and others through our connection with COBOL Cowboys. We're also making sure we keep our product technologically advanced and easy to use, so you can keep your COBOL, maintain it easily and keep it in modern applications.

Russel Kirsch

The inventor of the digital imaging pixel died August 11, 2020. He was 91. Mr Kirsch used a digital photograph of his 3 month old son in 1957 to demonstrate how computers could look at pictures.

Secure WebPages

Our websites are now https:// secure pages, including

www.veryant.com

and

support.veryant.com

NOW HIRING

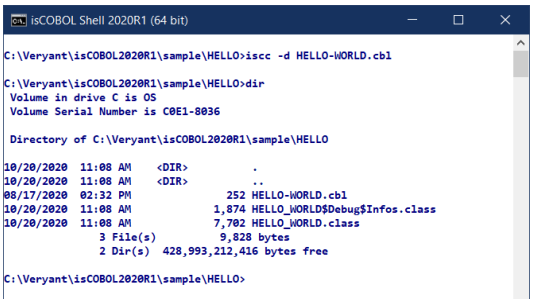
If you're looking for COBOL programmers, Try IBM's Community page "Calling all COBOL programmers". You'll find a mix of mainframe and non-mainframe programmers listed with credentials and contact information.

YouTube Scription Drive

Subscribe to our

YouTube Channel

Saying Farewell to iscobol.debug.code_prefix



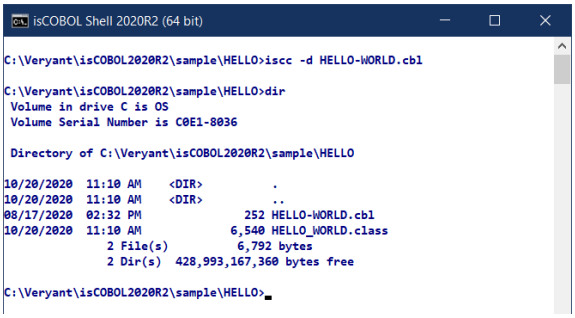
In 2020 R1 and earlier, debugging isCOBOL programs could only be done if a copy of the source code was made available to the runtime framework. Configuration variables would be set to point to these source files.

`iscobol.debug.code_prefix` (to find the files locally)
`iscobol.debug.remote_source` (when debugging remotely)

Compiling with -d also resulted in an additional class, in the format of <class name>$Debug$Infos.class.

We've made debugging much easier in 2020 R2 by removing the need for extra configuration and class files. Now, when you compile with -d or -dx the source code is included with the compiled class file. Here's some tips to remember when using this new debugging feature:

- To see if a class is compiled for debug, you can still run it with "isrun -info <class name>"
- The class compiled for debug will be bigger and run slower than one not compiled for debug.
- The source code in the class is encrypted
- You can use the same debugger commands ("br 7 HELLO-WORLD" still sets a breakpoint at line 7 of the program HELLO-WORLD).
- You can still use the configuration variables to load the source from the disk for backward compatibility



Don’t use the isCOBOL IDE? It’s easy to start.

Do you still use the command prompt to compile your programs? Notepad or vi to code? I think you would be surprised by how easy it is to move to isCOBOL’s IDE, and how useful you’ll find its features. Here are some of the features of the IDE that I think you’ll love:

- Keep your current file storage structure, and link the files to the IDE project folders – No need to change your current source structure or location.
- The IDE’s real time syntax checking feature lets you immediately see errors in the source code without having to compile – When you do compile you’ll know there are no syntax errors
- Use the Build Project feature to be sure you’ve compiled everything. For instance, if you have a project with 1,000 programs and you change a copy file used by 50 of those programs (some with nested copy files), Build Project will recompile just those programs that use the copy file. When the build is completed, it’s easy to update your production environment by ordering the output folder by date and moving only those 50 programs.
- You can easily create a process to automatically recompile all your programs during the night – for instance for internal tests needed by the test department. An example command line for this would be:

```
isIDE -data
workspaceLocation -nosplash
--launcher.suppressErrors
-application com.iscobol.
plugins.screenpainter.
IscobolScreenPainter.
builderApplication [project
projectName] refresh clean
build logfile logFilename
```

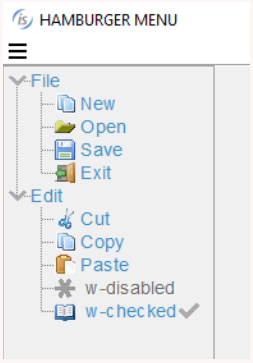
- The IDE easily integrates with code versioning software like SVN, so you can finally keep your changes organized and your code safe.
- You don’t need to work outside the IDE to access your command prompt-specific activities. Terminal Views connect locally or through SSH/Telnet from within the IDE
- Hyperlinks can move you around your code without constant scrolling
- It’s useful to view your code in different ways – the IDE shows you an outline view, lets you expand or collapse different sections of code, and split your screen to view two different parts of your code at the same time
- The IDE includes an advanced search and a quick access field so you can find anything in or outside the IDE.
- Use the IDE on Mac and Graphical Linux as well as Windows

The IDE is included in your Development system – no extra cost involved. Support is happy to help you move to the isCOBOL IDE. I think once you get used to it, you’ll wonder how you ever lived without it!

Davide Spizzi,
Support Engineering Manager

NEW HAMBURGER MENU

GUI Screens used to be considered a static environment – you only had to draw with one screen size in mind.
In 1999, Microsoft’s engineers said that computers were going to “become invisible”, getting hidden in common objects, and that’s certainly true of many COBOL programs.
It’s more common now to need to write responsive screens so they’ll display on monitors, smart phones, and tablets. At Veryant, we’re focused on making that as easy as possible for you.



Version 2020R2 introduces a hamburger menu that can be implemented by putting a configuration variable in your properties file rather than changing your code.

HOW TO CHANGE YOUR APPLICATION’S ICON

You may be using the ‘icon’ syntax in your display window statement to display an icon on the window. That icon also becomes the taskbar icon.

Another way of customizing your icon is using a configuration variable:

```
iscobol.gui.icon_file=<filename>
```

You don’t need to load or unload the icon with W$BITMAP, and updating your logo is as easy as replacing the icon file.

Evolution, without revolution

Veryant LLC

Veryant's dedicated and experienced support team is committed to providing the highest levels of customer care

For supported customer email us at support@veryant.com

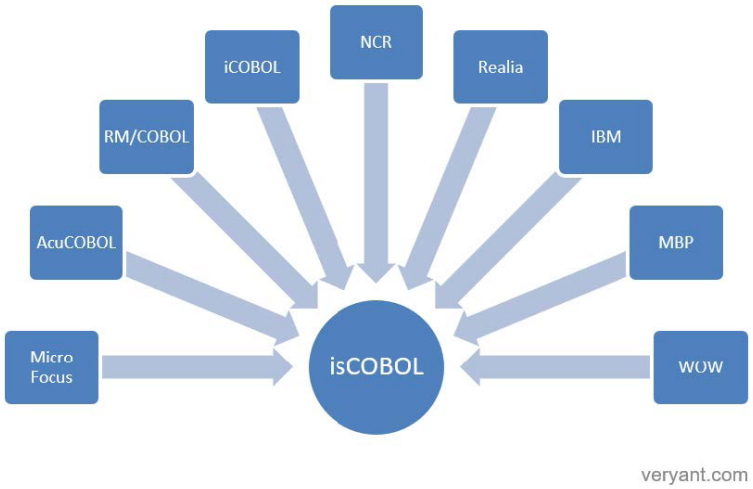
If you would like Veryant to contact you to schedule a technical product briefing, email us at info@veryant.com

If you would like Veryant to contact you for special quote or sales assistants email us at sales@veryant.com

Corporate Headquarters
6390 Greenwich Dr., Suite 225
San Diego, CA 92122 - USA
Tel (English): +1 619 797 1323
Tel (Español): +1 619 453 0914

European Headquarters
Via Pirandello, 29
29121 - Piacenza - Italy
Tel: +39 0523 490770
Fax: +39 0523 480784
emea@veryant.com

As always, 2020R2 contains multiple compatibility additions – as we continue to make your conversion process as smooth, quick, and pain-free as possible.



Offering choice, flexibility and cost-effective solutions to organizations with valuable COBOL assets

CONTACT US FOR A DEMONSTRATION

CLICK HERE