2024

ISSUE

12

QUARTERLY JOURNAL OF VERYANT AND isCOBOL

veryant NEWS

THIS ISSUE

We've rolled out our new customer support site. It's more intuitive to use, and has a friendlier UI.

If you've used our support site recently, your account has been moved over to the new site. Your login information is now your email, with the same password you've always used.

If you have trouble, or want access, contact your account manager at sales@veryant.com, or email support at support@veryant.com

2024 R2 and Happy Holidays

The release of 2024R2 with it's new Chips box control has been very popular with our customers.

After the recent release of isCOBOL Evolve 2024R2, we've jumped right into the upcoming 2025R1 We'll talk more about what's coming in the next newsletter.

We here at Veryant are settling into the holiday months with warm wishes for our customers' continued success and happiness.

This edition of the newsletter contains part 2 of Valerio's debugging tutorial, information about using JWT's (JSON Web Token) for advanced security, and the start of a discussion about AI and COBOL.

We hope you have a great holiday season and a healthy and prosperous new year!

Are you running an older version of isCOBOL? We're here to help you upgrade to stay with a supported versions and take advantage of new technology. Check out our end-of-life policy here.

How Can Artificial Intelligence Benefit our COBOL applications?

The discussion around AI's role in the future of COBOL underscores both its potential and its limitations. AI may never completely replace COBOL developers, but it will likely evolve to be a critical assistant, reducing their workload and enabling them to focus on high-value tasks.

Is AI ready to replace COBOL developers?

AI tools like IBM's WatsonX and other generative models (e.g., ChatGPT, GitHub Copilot) show promise but are not yet fully equipped to replace COBOL developers. Experienced COBOL programmers report incomplete or inaccurate AI-generated code on social media discussions.

AI can't replicate the deep institutional knowledge and creativity of human developers. However, it can assist in identifying areas of interest within legacy systems and provide recommendations and insights to streamline the development process.

What roles can AI currently play?

AI's strength lies in enhancing developer productivity through supportive roles:

- **Code Translation:** Tools like WatsonX achieve partial automation of COBOL-to-Java translations, handling 75-80% of the work, but still rely on human expertise to refine results. Though these tools are focused on mainframe enterprise COBOL, Veryant will soon be offering a 100% COBOL conversion to Java for open-system environments.
- **Documentation and Understanding:** AI can auto-generate documentation, map dependencies, and conduct impact analysis, which are crucial for maintaining large, opaque COBOL systems.
- **Bug Fixing and Debugging:** Developers spend about three quarters of their time locating code containing a bug or the location of additional code affected by a change in one location. Advanced AI systems can identify code sections requiring updates or corrections, minimizing time spent searching through vast codebases.
- **Reducing Technical Debt:** AI tools are being employed to identify inefficient or redundant code, helping companies like Wayfair manage technical debt.

Symbolic AI vs. Generative AI

To do all this, we need more than an AI that generates code, called Generative AI. A new type of AI called Symbolic AI is being used to give a more common-sense, human-like approach to AI. Here's a description of the differences between these two types of AI:

- **Generative AI (e.g., ChatGPT):** Focuses on creating content (code, comments, etc.). It's useful for prototyping but lacks the reasoning capabilities to tackle complex systems with interconnected dependencies.
- **Symbolic AI (e. g., PhaseChange's COBOL Colleague):** Designed for reasoning and understanding cause-effect relationships in code. Symbolic AI could aid in conceptualizing what legacy systems are doing and automating system-level insights that allow developers to make informed changes without deep-diving into the entire codebase.

This division suggests that the combination of both approaches could yield the most productive results.

Long-Term Predictions

By 2028, AI-human collaboration is expected to reduce development times by 30%, according to Gartner. This could lower the barrier to maintaining legacy COBOL systems, extend the lifecycle of these systems while companies modernize incrementally, and enable developers to focus on innovation rather than maintenance.

Conclusion

AI, whether generative or symbolic, isn't a silver bullet for COBOL application upkeep and development. Instead, it's a powerful ally that can streamline workflows, reduce technical debt, and augment the productivity of developers. The road ahead lies in a hybrid approach—leveraging both human expertise and AI's capabilities to ensure legacy systems continue to operate effectively while being modernized where necessary.

10,000.00 OR 10.000,00? DECIMAL POINTS FOR MULTIPLE COUNTRIES

When you write a COBOL program to run in Europe you would set the program to use a comma as a decimal separator by adding DECIMAL-POINT is comma to the special names section.

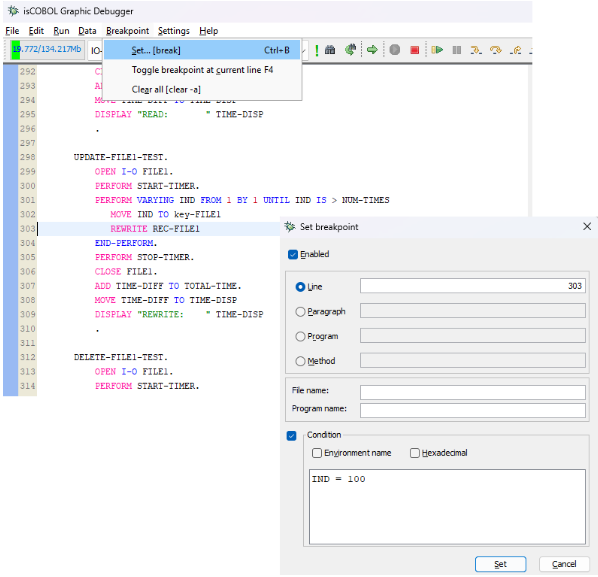
But if you need to execute the same program in both the USA and Europe you'll need to accommodate both types of decimal separators. This is easy in isCOBOL with no program changes. Here are the steps:

1. Compile with the option -sddp
2. When you run in Europe, change the decimal point behavior at runtime setting the configuration property iscobol.runtime.decimal_point_is_comma to true

Working in the Debugger

Once bugs are found, coders can begin the process of debugging and work towards ridding software of any errors. Let’s look at some of isCOBOL’ smart debugger functions that allow coders to identify bugs.

The BREAKPOINT is an intentional stopping or pausing place in a program put in a place for debugging purposes. In practice, a breakpoint consists of one or more conditions that determine when a program’s execution should be interrupted. You can set a breakpoint from the menu, by typing Control+B in the command field, or right-clicking on the code in the debugger. Here’s an example of how to stop at a specific line when a condition has been met (in this case when ind = 100)



```
           DISPLAY "READ:       " TIME-DISP
           .

       UPDATE-FILE1-TEST.
           OPEN I-O FILE1.
           PERFORM START-TIMER.
           PERFORM VARYING IND FROM 1 BY 1 UNTIL IND IS > NUM-TIMES
              MOVE IND TO key-FILE1
              REWRITE REC-FILE1
           END-PERFORM.
           PERFORM STOP-TIMER.
           CLOSE FILE1.
           ADD TIME-DIFF TO TOTAL-TIME.
           MOVE TIME-DIFF TO TIME-DISP
           DISPLAY "REWRITE:    " TIME-DISP
           .

       DELETE-FILE1-TEST.
           OPEN I-O FILE1.
           PERFORM START-TIMER.
```

Set breakpoint

☑ Enabled

● Line: 303

○ Paragraph

○ Program

○ Method

File name:

Program name:

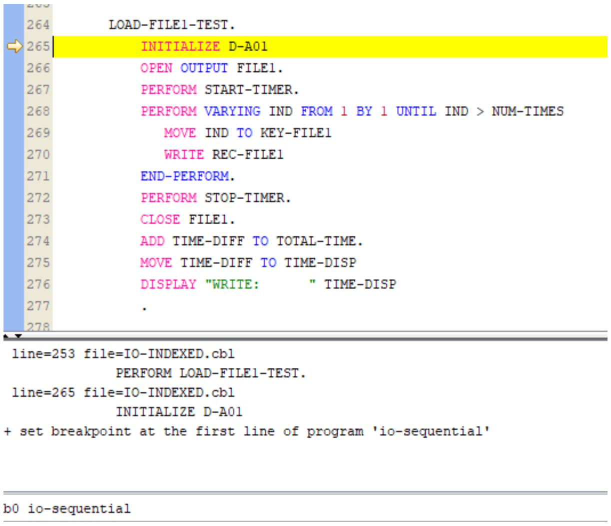
☑ Condition

☐ Environment name ☐ Hexadecimal

```
IND = 100
```

You can also set a breakpoint at the beginning of a called program, thru the command area typing:

```
b0 program name
```



```
       LOAD-FILE1-TEST.
           INITIALIZE D-A01
           OPEN OUTPUT FILE1.
           PERFORM START-TIMER.
           PERFORM VARYING IND FROM 1 BY 1 UNTIL IND > NUM-TIMES
              MOVE IND TO KEY-FILE1
              WRITE REC-FILE1
           END-PERFORM.
           PERFORM STOP-TIMER.
           CLOSE FILE1.
           ADD TIME-DIFF TO TOTAL-TIME.
           MOVE TIME-DIFF TO TIME-DISP
           DISPLAY "WRITE:       " TIME-DISP
           .
```

```
line=253 file=IO-INDEXED.cbl
            PERFORM LOAD-FILE1-TEST.
line=265 file=IO-INDEXED.cbl
            INITIALIZE D-A01
+ set breakpoint at the first line of program 'io-sequential'
```

```
b0 io-sequential
```

SCALABILITY FOR COBOL APPLICATIONS

Scalability ensures business infrastructure adapts to growth, improving efficiency, optimizing resources, and enabling quick adaptation. For COBOL systems, this means modernizing and extending capacity without replacing core code.

Veryant’s isCOBOL Solutions

Veryant offers tools to make COBOL applications scalable:

- Cloud Support: isCOBOL runs seamlessly in the cloud or any JRE-supported environment, enabling flexible, elastic scaling.
- isCOBOL LoadBalancer: Distributes ThinClient sessions across servers to optimize resources and prevent bottlenecks.
- Clustered WebClient: Distributes WebClient sessions across multiple pools, ensuring reliable, fast access during peak usage.

Steps to Scale COBOL Applications

1. Evaluate Systems: Assess performance under various workloads.
2. Adopt Cloud Solutions: Use isCOBOL’s cloud capabilities for flexibility.
3. Leverage Load Balancers and Clustering: Improve user interactions with tools like isCOBOL LoadBalancer.
4. Monitor Performance: Optimize systems continuously

Why Choose Veryant?

Veryant’s tools help modernize COBOL systems for scalability, supporting growth without overhauls. Contact a Veryant account manager to explore tailored solution

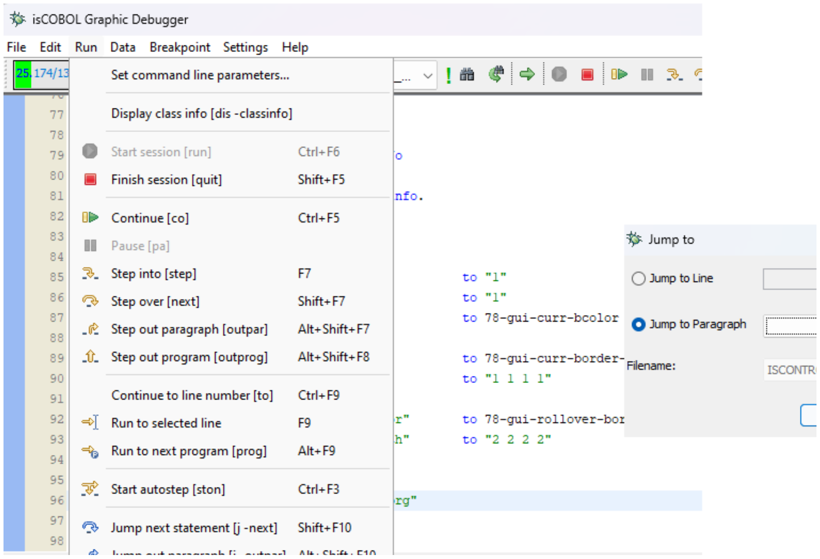
PLEASE JOIN US ON

Twitter, LinkedIn, or Facebook for up-to-date with Veryant's news

Watch our demonstration videos and Subscribe to our YouTube Channel

Working in the Debugger

The JUMP functionality, available only on programs compiled with the -dx option, lets the user jump to a specific line, skipping the code between the current line and the destination line.


isCOBOL Graphic Debugger
File Edit Run Data Breakpoint Settings Help
25. 174/13
Set command line parameters...
Display class info [dis -classinfo]
Start session [run] Ctrl+F6
Finish session [quit] Shift+F5
Continue [co] Ctrl+F5
Pause [pa]
Step into [step] F7
Step over [next] Shift+F7
Step out paragraph [outpar] Alt+Shift+F7
Step out program [outprog] Alt+Shift+F8
Continue to line number [to] Ctrl+F9
Run to selected line F9
Run to next program [prog] Alt+F9
Start autostep [ston] Ctrl+F3
Jump next statement [j -next] Shift+F10
Jump to
○ Jump to Line
● Jump to Paragraph
Filename:
ISCONTR

```
to "1"
to "1"
to 78-gui-curr-bcolor
to 78-gui-curr-border-
to "1 1 1 1"
to 78-gui-rollover-bor
to "2 2 2 2"
```

You can see all the debugger commands in our documentation here.

Keep in touch with the next newsletters where we'll see other amazing functionalities! Stay tuned!

.

Have You Seen This?

Newest Video:

Visual Studio Code, or VS Code, is a free editor from Microsoft. Combined with the isCOBOL VSCode extension, you can get the benefits like syntax checking, predictive typing, and in-app debugging in an easy-to-use environment.
Here's a video demonstrating how to get started.

New KB Articles:

How to automatically terminate a thin-client session that has been idle for a specific period of time.

Can a COBOL procedural program be INVOKED with an array of objects instead of a standard CALL with USING

Advanced Web API Authorization and Signature

When you want to implement an advanced API authentication you will need a program that can validate the authenticity and integrity of a JSON Web Token (JWT) signed with an RSA public key.

JWTs are a compact and self-contained way to securely transmit information between parties as a JSON object.

Let’s consider the following sample code:

```
program-id. CobJwtDecode.

 configuration section.
 repository.
     class PublicKey   as “java.security.PublicKey”
     class X509Certificate
                       as “java.security.cert.X509Certificate”
     class RSAPrivateKey
                       as “java.security.interfaces.RSAPrivateKey”
     class RSAPublicKey
                       as “java.security.interfaces.RSAPublicKey”
     class RSAKey      as “java.security.interfaces.RSAKey”
     class JWT           as “com.auth0.jwt.JWT”
     class JWTVerifier   as “com.auth0.jwt.JWTVerifier”
     class Algorithm     as “com.auth0.jwt.algorithms.Algorithm”
     class DecodedJWT    as “com.auth0.jwt.interfaces.DecodedJWT”
     class X509CertUtils as “com.nimbusds.jose.util.X509CertUtils”
 .
 working-storage section.
 77  token       pic x any length.
 77  certificate pic x any length.
 77  cert        object reference X509Certificate.
 77  pubKey      object reference PublicKey.
 77  o-publicKey object reference RSAPublicKey.
 77  o-algorithm object reference Algorithm.
 77  verifier    object reference JWTVerifier.
 77  o-jwt       object reference DecodedJWT.

 procedure division.
 main.
* TODO Auto-generated method stub
     move
     “eyJhbGciOiJSUzI1NiIsImtpZCI6IjYwZTQxMjczMzMwYTg2ZmRjMjhlMjgz
-    “MDVhNDRkYzlhODgzZTI2YTciLCJ0eXAiOiJKV1QifQ.eyJuYW1lIjoiQnJpY
-    “W4iLCJwaWN0dXJlIjoiaHR0cHM6Ly93d3cudzNzY2hvb2xzLmNvbS9ob3d0b
-    “y9pbWdfYXZhdGFyLnBuZyIsImlzcyI6Imh0dHBzOi8vc2VjdXJldG9rZW4uZ
-    “29vZ2xlLmNvbS9jb2xsZWN0aW9uLXBhcnRuZXIiLCJhdWQiOiJjb2xsZWN0a
-    “W9uLXBhcnRuZXIiLCJhdXRoX3RpbWUiOjE1NjMzNzM1MjUsInVzZXJfaWQiO
-    “iJIU0tSWTdrTGxHUU9GSkwzVlUxWXhiWFJNYk0yIiwic3ViIjoiSFNLUlk3a
-    “0xsR1FPRkpMM1ZVMVl4YlhSTWJNMiIsImlhdCI6MTU2MzM3MzUyNSwiZXhwI
-    “joxNTYzMzc3MTI1LCJlbWFpbCI6ImJyaWFuLnN1bGxpdmFuMUB5YWhvby5jb
-    “20iLCJlbWFpbF92ZXJpZmllZCI6dHJ1ZSwiZmlyZWJhc2UiOnsiaWRlbnRpd
-    “GllcyI6eyJlbWFpbCI6WyJicmlhbi5zdWxsaXZhbjFAeWFob28uY29tIl19L
-    “CJzaWduX2luX3Byb3ZpZGVyIjoicGFzc3dvcmQifX0.J6j4b562mHM9U62kZ
-    “F3yMyZhf8AaL3Cum4vGPnFcHc1H9V9dH720v8OURMTCsZKhOrMfO46t8EoTC
-    “EWIJfY-NskA-L9ulzem8lwk5gcac662fam8Jz1HUJGFH4LTIVJbozsWXccIx
-    “kbIm3FrZVWKJAmAE71S_6Id4Nw5Jq5S5gEmi-DBVSIRntieYogYvkSNBmfRV
-    “NiMhR5ixZrNkFOSOQH-RiRGgzTz1VpRT8EwKxlgHmrtiriA7DRtoQJy7jIN0
-    “oGfpYf9aI1pyKMeEAMD3Hi9UXrb_VP0MgO8FP2GbWH4vTGh3MtsX49xcAX-i
-    “B7LIUu5O8Ia0Wt8TUSXqe_Q-A” to token.

     perform check-token.
     goback.
 check-token.
     move
     “-----BEGIN CERTIFICATE----- MIIDHDCCAgSgAwIBAgIIITyq+eI2xDsw
-    “DQYJKoZIhvcNAQEFBQAwMTEvMC0GA1UE AxMmc2VjdXJldG9rZW4uc3lzdGV
-    “tLmdzZXJ2aWNlYWNjb3VudC5jb20wHhcNMTkw NzA3MjEyMDU1WhcNMTkwNz
-    “I0MDkzNTU1WjAxMS8wLQYDVQQDEyZzZWN1cmV0b2tl bi5zeXN0ZW0uZ3Nlc
-    “nZpY2VhY2NvdW50LmNvbTCCASIwDQYJKoZIhvcNAQEBBQAD ggEPADCCAQoC
-    “ggEBALmpZwoVX2b2/nwphVDh+Y3+F5d6EZuk3rgDoP2Vu/Lrqapx ovdxAt0
-    “atmuU+YiyU6wGHBycoMXu0nWKRdArn8VdoDa8vm3RxX2WFwe8u9yIo1Ju Pq
-    “dGOaiIdgbsNSxXofmkepEDoBNdRqdfRfxZ19H+mbOZ6gjb79gHU0n58cb1vP
-    “5C r02RYjiC/Fy7jZ+D0AqODE/e1kU76+SyeAISAKFQnzd62Zg9rIZ0zeZ7F
-    “J5U4BJ5 S/JsoLe+gUtC7ZTZHd9wNc5Ga3+KE7scYEsO18lk6UlxF+VN0R3M
-    “Z/ObJ/q3Q9dR pDQU3OB19xvPrBC5kypiXFPqabg+mHfZUxaSWzECAwEAAaM
-    “4MDYwDAYDVR0TAQH/ BAIwADAOBgNVHQ8BAf8EBAMCB4AwFgYDVR0lAQH/BA
-    “wwCgYIKwYBBQUHAwIwDQYJ KoZIhvcNAQEFBQADggEBAGiVv+cdrAuZrNV5w
-    “VPFaCvUSiGn+LGq/OC6hZWiljt/ AEn6ypoZfkRJEwTv+IrZHKcEBSR7Dees
-    “n0sj+tigL1g37w3I9I1WNEaip/snCCTe BtBm4gmkX9cDv4Ga/rhObGOZAhN
-    “NhEFWY8ofQFuohOxoaZe/Z5fvoDZW6eFKc84a q/tNDaN2Xglyiadccgux9/
-    “7OH2oH+AwoniPlIHmQAaccUsNqOjg/X4WpaIKfBJ2i 9zhevne1IUm0RMxPH
-    “oHSR8iqllu9phpeRX7Po1tRfg3YUY2fyaC4Wad9FBJAGBx7 9YSEIyd0K82o
-    “3h7Y3ITtIGOmiA8NDI9tkNOTVpBTXds= -----END CERTIFICATE-----”
     to certificate.
     Try
        set cert    to X509CertUtils:>parse(certificate)
        set  pubKey to cert:>getPublicKey()
        set o-publicKey to pubKey as RSAPublicKey
        set o-algorithm to Algorithm:>RSA256(o-publicKey
                                                        as RSAKey)
        set verifier    to JWT:>require(o-algorithm)
                              :>withIssuer
             (“https://securetoken.google.com/collection-partner”)
                              :>build() |build the verifier instance
        set o-jwt to verifier:>verify(token)
        | access jwt fields if needed to verify token data
        display “Token verification successful”
     catch exception
        |Invalid signature/claims
        display “Token verification failed: “
                 exception-object:>getLocalizedMessage()

     end-try.
```

How it works

The program’s primary purpose is to verify that a given JWT is genuine and has not been tampered with.

1. The program’s repository imports necessary classes for handling RSA keys, certificates, and JWT operations. Those classes are included in the following .jar files: jackson-core-2.9.9.jar, jackson-annotations-2.9.9.jar, commons-codec-1.9.jar, nimbus-jose-jwt-6.0.jar, java-jwt-3.8.1.jar, jackson-databind-2.9.9.2.jar, json-smart-2.2.1.jar
2. The main paragraph sets a JWT string in the token item for it to be used by the next paragraph. In checkToken, a hardcoded X.509 certificate in PEM format is parsed to obtain an X509Certificate object.
3. The public key is extracted from the certificate and checked if it is an RSAPublicKey, and then an Algorithm instance is created using that key.
4. A JWTVerifier instance is constructed, specifying the issuer (https://securetoken.google.com/collection-partner), and then the verifier:>verify(token) method is called to verify the JWT. A confirmation or an error message is displayed depending on the success or failure of this.

This program provides a mechanism for ensuring that a JWT can be trusted. By verifying the token’s signature and other critical attributes, it helps to prevent unauthorized access and protect sensitive data.

Evolution, without revolution

Contact Us

For supported customer email us at support@veryant.com

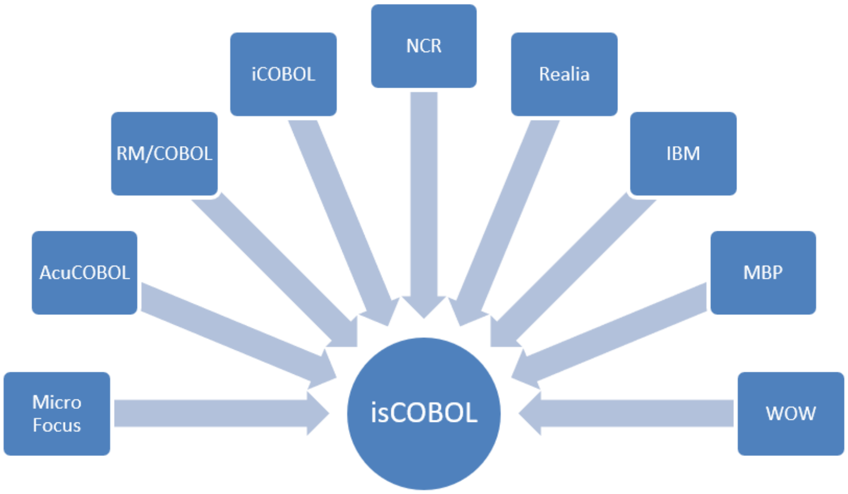
If you would like Veryant to contact you to schedule a technical product briefing, email us at info@veryant.com

If you would like Veryant to contact you for special quote or sales assistance 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, the newest isCOBOL Evolve release contains multiple compatibility additions – as we continue to make your conversion process as smooth, quick, and pain-free as possible.



veryant.com

Follow Veryant on

veryant.com