



BIANNUAL JOURNAL OF VERYANT AND isCOBOL

veryant

NEWS

THIS ISSUE

1. 2022 R2 **3.** Mouse events | What are compiler directives | documentation highlights **4.** Docker Containers – part three **11.** Should we still use COBOL? Where do isCOBOL wrappers look for Java? **12.** New KB articles and videos.

PLEASE JOIN US ON

LinkedIn or Facebook to
up-to-date with



Watch our demonstration
videos and Subscribe to
our YouTube Channel



2022 R2 is here

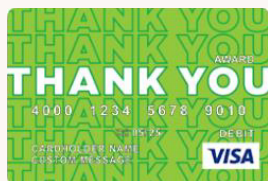
isCOBOL 2022 Release 2 was released September 12, 2022.



TELL US:

We want to spread the word
about how wonderful Veryant
is. And we think you can help
us.

If you know of another
company – ISV or End User
– running applications in
COBOL, send us their contact
information, and we'll send you
a \$50.00 pre-paid Visa Credit
Card. Just like that!



**sCOBOL 2022 Release 2 was released
September 12, 2022. It includes:**

- **Clearer GUI screens with high DPI-aware controls and windows**
- **A new compiler capable of beating all of our competitors' batch processing time and instructions-per-second levels**
- **A compiler pre-processor feature so you can change your COBOL code on-the-fly – as it's being compiled.**

2022 R2 is here

Being high-DPI aware is an important feature if you've written your GUI screens in COBOL and you plan to run your applications on different size screens. The old method of assuming all screens have the same number of dots, or pixels, per inch (DPI), and then stretching them out if the DPI is larger is OK when everything's run on monitors about the same size.

But when you add smartphone and tablet screens, as well as those huge-dpi laptops coming out now, your screens may have started to look fuzzy and dated. We've solved this in 2022R2 with windows and GUI controls that adjust to the DPI correctly, making your screens as sharp and up-to-date as you originally programmed them. You don't need to do anything like recompile, or add a variable to your properties file – simply upgrade your runtime to the new version and run it!

We've been working on a completely new compiler using newer technology so your applications will run faster with less memory. And this version is our first release of that compiler. You can use it to make your batch, non-ui, programs fly!

Remove, add, or change your code using our new line-by-line compiler preprocessor feature. Write a program to do something like move lines written to the console to a log file, add comment lines at the beginning of every program to print your copyright

information, or whatever change you want to make to your programs without actually changing the code. Write a program to make the change, and the compiler will read your program and do what you need before it creates the Java code and compiles to a Java class.

With the usual bevy of compatibility additions, GUI control enhancements, and new configuration options, 2022R2 is sure to have something for everyone.

TELL US:

If you aren't a Veryant customer, and are interested in switching to isCOBOL, now's the best time to do it.

We have special deals for ISVs and Corporate End Users through the end of the year to make it easy to afford, and the compatibility to make it easy to move your code to isCOBOL.

Learn more about this deal [here](#).



Mouse Events

Did you know you can manage mouse events on all controls?

Some controls return an event to the event-procedure paragraph when it's important to intercept mouse events like the "click" or "double click" on an icon or cell. For example the events MSG-GOTO-CELL-MOUSE on grids, MSG-BITMAP-CLICKED and MSG-BITMAP-DBLCLICK on grids and entry-fields with bitmaps.

You can also receive events on other controls by activating the NOTIFY-MOUSE style, so that these events are returned to the event-procedure: MSG-MOUSE-ENTER, MSG-MOUSE-EXIT, MSG-MOUSE-CLICKED, MSG-MOUSE-DBLCLICK.

A typical use of these events are on controls like bitmaps, labels, and frames. These controls don't fire any events by default, so setting the NOTIFY-MOUSE style gives developers finer control on the user interface design.

For example, here is a code snippet to manage MSG-MOUSE-CLICKED event on a label:

```
...
03 l1 label line 2 col 2 size 10
    title "Click me"
    notify-mouse event label-events
...
label-events.
    if event-type = msg-mouse-clicked
        perform do-action
    end-if.
```

What are Compiler Directives?

Compiler directives are a special syntax written in the source code to declare something that is particular for this source. By putting the source-specific compiler options in the source code, you can compile all your programs with one set of standard options defined in the SDK script or IDE project level.

Here are some examples:

- If only a few of your programs need to use long lines, you can set the format for just those programs by adding this directive to the top of those programs that need it:
`>> SOURCE FORMAT FREE`
- For Fixed format (aka ANSI) programs that need to write long lines, a good alternative to Free format is to use this directive (equivalent to `-sl` option):
`>> IMP MARGIN-R IS AFTER END OF RECORD`
- If only one program needs the `-big` compiler option to have a clean compilation while avoiding the java compiler error "too many constants", you can set:
`>> IMP OPTION "-big"`

Another benefit to using compiler directives is to write a source that needs to include or exclude some part of code depending on a specific condition. More information about this is explained in this KB article:

<https://support.veryant.com/support/phpkb/question.php?ID=27>

DOCUMENTATION HIGHLIGHTS

isCOBOL supports embedded SQL without the need of precompiling the sources. The embedded SQL code is compiled along with other COBOL statements in one step. The resulting COBOL program will connect to databases with JDBC technology.

The isCOBOL approach has some advantages if compared with a precompiler approach:

- Simpler compilation process, as the compilation is done in one step
- Simpler deployment, as you need only to change the JDBC driver library to access different databases, keeping the same compiled class
- Thread safety of the JDBC driver that lets you run the programs in application server environments like Tomcat

How can I move my ESQL programs from a precompiler environment like Oracle

Pro*COBOL or DB2 Preprocessor to JDBC? Is there any issue I should be aware of? The answer to these questions can be found in the following Transitioning Guides:

[Transitioning from Pro*COBOL](#)

[Transitioning from the IBM DB2 Preprocessor](#)



The third part of a four-part series on isCOBOL and Docker Containers, this article is a step-by-step guide to running more than one container. From Senior Support Engineer, Valerio Biolchi

Modern apps consist of different components that need to communicate with each other.

Communication between the isCOBOL Server and c-treeRTG Running in Different Docker Containers

Prerequisites

The prerequisites on a Linux 64bit computer (where the docker is built) are :

- The Docker engine (<https://docs.docker.com/engine/install/>) installed from a repository
- A licensed isCOBOL SDK environment
- An isCOBOL Server running in Docker having ISAPPLICATION sample setup and running. See instructions in the first article of the series, [here](#).

In the real world, beyond the realm of the simple hello-world tutorial, running just one container isn't enough for most apps. A modern application typically consists of multiple components – such as a database, a web server, and some microservices. So, if you want to run all of your components in containers, how can the applications talk to each other?

How do containers communicate with each other, if they're supposed to be isolated?

Let's start looking at a simple communication between Docker containers, when they are running on the same host (sometimes called single-host networking).

How do containers communicate?

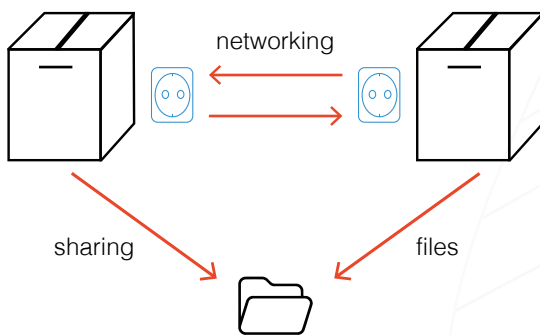
First, a quick overview! Although containers have a level of isolation from the environment around them, they often need to communicate with each other, and the outside world.

You've gone through the other articles and you've run your first Docker containers. But now you're struggling to understand how to run more than one container at the same time. If Docker containers are isolated, then how the heck do they communicate with each other?

Communication between the isCOBOL Server and c-treeRTG Running in Different Docker Containers

Two containers can talk to each other in one of two ways, usually:

- Communicating through networking: Containers are designed to be isolated. But they can send and receive requests to other applications using networking.
- Sharing files on disk: Some applications communicate by reading and writing files. These kinds of applications can communicate by writing their files into a [volume](#), which can also be shared with other containers.



Building your (Virtual) Network

A Docker network lets your containers communicate with each other

If you are running more than one container, you can let your containers communicate with each other by attaching them to the same network.

In a network, a container has an IP address, and optionally a hostname.

You can create different types of networks depending on what you would like to do. We'll cover the easiest options:

- The default bridge network, which allows simple container-to-container communication by IP address and is created by default.
- A user-defined bridge network, which you create yourself, that allows your containers to communicate with each other by using their container name as a hostname.

Building a Bridge Network

The simplest network in Docker is the bridge network. It's also Docker's default networking driver. A [bridge network](#) gives you simple communication between containers on the same host.

When Docker starts up, it will create a default network called... bridge. 🤔

It should start automatically, without any configuration required by you. From that point forwards, all containers are added to the bridge network, unless you say otherwise. In a bridge network, each container is assigned its own IP address. So containers can communicate with each other by IP.

Typing `docker network` should show the bridge network in the list

```
root@ubuntu:/home/valerio/myDocker/myApp4# docker network ls
NETWORK ID      NAME      DRIVER  SCOPE
6e432c2bd969    bridge   bridge  local
a11a782681ea    host     host    local
8eb1f36b9925    none     null    local
```

Communication between the isCOBOL Server and c-treeRTG Running in Different Docker Containers

Build and start the c-treeRTG Server container

As explained into the first article, we start with the “Dockerfile” instruction file. These are the minimum directories we need to copy in order to properly start the c-treeRTG Server:

```
COPY c-treeRTG_v3.0.2/config ${CTREE}/config
COPY c-treeRTG_v3.0.2/data ${CTREE}/data
COPY c-treeRTG_v3.0.2/server ${CTREE}/server
COPY c-treeRTG_v3.0.2/tranlogs ${CTREE}/tranlogs
```

Here are the full contents of the Dockerfile we’ve created to have a container fully working with the c-treeRTG Server:

```
# Dockerfile
FROM openjdk:11
MAINTAINER Veryant
ENV CTREE=/var/c-treeRTG3.0.2
RUN mkdir ${CTREE}
COPY c-treeRTG_v3.0.2/config ${CTREE}/config
COPY c-treeRTG_v3.0.2/data ${CTREE}/data
COPY c-treeRTG_v3.0.2/server ${CTREE}/server
COPY c-treeRTG_v3.0.2/tranlogs ${CTREE}/tranlogs
WORKDIR “${CTREE}/server”
CMD ./ctreesql run
```

Build the c-treeRTG image

Now we can build the ctree docker image from the folder containing the Dockerfile. The repository name (ctree in this example) must be lowercase.

```
root@ubuntu:/home/valerio/myDocker/myApp4# ls
c-treeRTG_v3.0.2 Dockerfile
root@ubuntu:/home/valerio/myDocker/myApp4# docker build -t ctree
```

Starting the c-treeRTG container as normal adds it to the bridge network:

```
docker run --name ctreeserver -d -v /isapplication/data:/isapplication/data -p 5597:5597 ctree
```

Communication between the isCOBOL Server and c-treeRTG Running in Different Docker Containers

`/isapplication/data:/isapplication/data` mounts the host directory `/isapplication/data` to the `/isapplication/data` in the container. To find the IP addresses of a container, look at the output of the `docker inspect` command:

```
root@ubuntu:/home/valerio/myDocker/myApp4# docker inspect ctreeserver | grep IPAddress
    "SecondaryIPAddresses": null,
    "IPAddress": "172.17.0.2",
    "IPAddress": "172.17.0.2",
```

Now we can check that the TCP/IP communication running an ‘iscobolserver’ container will allow the execution of the IO_INDEXED sample benchmark. The Docker file used to build the iscobol image to run the IO_INDEXED will contain:

```
FROM openjdk:11
MAINTAINER Veryant
ENV ISCOBOL=/var/isCOBOL2022R2
ENV ISCOBOL_CLASSPATH=${ISCOBOL}:${ISCOBOL}/io-performance
RUN mkdir ${ISCOBOL}
COPY iscobol.properties ${ISCOBOL}/iscobol.properties
COPY isCOBOL_SDK2022R2/lib ${ISCOBOL}/lib
COPY isCOBOL_SDK2022R2/native/lib ${ISCOBOL}/native/lib
COPY isCOBOL_SDK2022R2/bin/isserver ${ISCOBOL}/bin/isserver
COPY isCOBOL_SDK2022R2/sample/io-performance ${ISCOBOL}/io-performance
WORKDIR "${ISCOBOL}/io-performance"
CMD ${ISCOBOL}/bin/isserver run
```

The iscobolserver is started with this command:

```
docker run --name iscobolserver -e ISSERVER_OPTS='-c /isapplication/data/runtime.properties' -dit -v /isapplication/data:/isapplication/data -p 10999:10999 iscobol
```

`-e <env_variable>` is used to set an environment variable for the ‘isserver’ daemon started by container. The `/isapplication/data/runtime.properties` contains the proper configuration properties to work and communicate properly with the c-treeRTG Server:

```
iscobol.file.index=ctreej
iscobol.file.index.server=FAIRCOMS@172.17.0.2
```

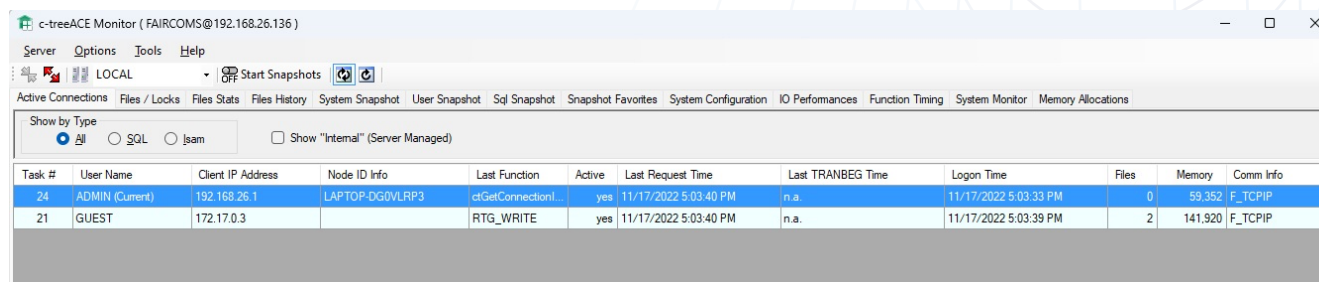
Communication between the isCOBOL Server and c-treeRTG Running in Different Docker Containers

Finally running the IO_INDEXED thru an isCOBOL Client we see the execution working with TCP/IP:

```
Command Prompt
E:\Veryant\isCOBOL_Evolve2022R2\sdk>iscclient -hostname 192.168.26.136 IO_INDEXED

INDEXED FILES
NUM-TIMES: 10000
WRITE:      2.88
READ:       2.13
REWRITE:    1.80
DELETE:     1.96
Total Time: 8.77
```

During the execution of the isCOBOL Client we can also see in the c-treeACE Monitor utility what clients are attached to the c-treeRTG Server:



The screenshot shows the c-treeACE Monitor utility interface. The title bar indicates it is connected to FAIRCOMS@192.168.26.136. The 'Active Connections' tab is selected, displaying a table of active clients.

Task #	User Name	Client IP Address	Node ID Info	Last Function	Active	Last Request Time	Last TRANBEG Time	Logon Time	Files	Memory	Comm Info
24	ADMIN (Current)	192.168.26.1	LAPTOP-DG0VLRP3	ctGetConnectionI...	yes	11/17/2022 5:03:40 PM	n.a.	11/17/2022 5:03:33 PM	0	59,352	F_TCP/IP
21	GUEST	172.17.0.3		RTG_WRITE	yes	11/17/2022 5:03:40 PM	n.a.	11/17/2022 5:03:39 PM	2	141,920	F_TCP/IP

The c-tree client connected with the IP Address equal to 172.17.0.3 corresponds to the address assigned to the 'iscobolserver' container by the Docker bridge network:

```
root@ubuntu:/home/valerio/myDocker/myApp3# docker inspect iscobolserver |grep IPAddress
    "SecondaryIPAddresses": null,
    "IPAddress": "172.17.0.3",
    "IPAddress": "172.17.0.3",
```

The second option, the user-defined bridge, lets you have a bit more control.

Communication between the isCOBOL Server and c-treeRTG Running in Different Docker Containers

Get more control, with a user-defined bridge

To let Docker containers communicate with each other by name, you can create a user-defined bridge. In a **user-defined bridge** network, you can be more explicit about who joins the network, and you get an added bonus:

...containers can be addressed by their name or alias.



Create a user-defined bridge network

Create your own custom bridge network first using the docker network create command. Under the hood, Docker sets up the relevant networking tables on your operating system.

```
docker create network veryant
```

Connect the ctreeserver container to your user-defined bridge

Start the ctreeserver container and connect it to the bridge using the `--net` option.

```
docker run --net veryant --name ctreeserver -d ...
```

Communication between the isCOBOL Server and c-treeRTG Running in Different Docker Containers

Connect the iscobolserver container to your user-defined bridge

Start the iscobolserver container using ctreeserver as the hostname. When two containers are joined to the same user-defined bridge network, one container is able to address another by using its name (as the hostname)

```
docker run --net veryant --name iscobolserver -e ISSERVER_OPTS='-c /isapplication/data/runtime.properties ...
```

Connect through shared memory

The next Docker article will show how to setup the communication between Docker containers thru the shared memory. Stay tuned!

When computer programmers were trained on the job, the majority of computer programmers were women like Kathleen Booth. Then computers got small and affordable enough to give to kids as presents, enabling them to learn to code in childhood. And parents

almost always gave these presents to boys. When it came time to choose a career, men had the computer experience that women didn't have, so men became the majority of programmers. Celebrating Kathleen Booth!



THEREGISTER.COM

RIP: Kathleen Booth, the inventor of assembly language

Builder and programmer of the ARC and SEC turned 100 this year

Read more about it here: [RIP: Kathleen Booth, the inventor of assembly language • The Register](#)

COBOL

Should we still use COBOL?

This was the question posed and answered by Tom Ross in [his blog on IBM's community](#). He says COBOL was first declared dead in 1969 when PL/1 was introduced, and then again several times after that. But because COBOL's strengths haven't been well replicated in other languages, COBOL continues to dominate banks, insurance companies, large-scale manufacturing and government applications.

COBOL excels in number crunching and customer data processing, with its expert handling of character strings, variable-length records, and conversion from character to numeric data.

COBOL is better at self-documenting with an English-like syntax that's highly readable. It's wordy, but Ross counts that as an advantage, making it difficult to write mysterious (and therefore less maintainable) code.

COBOL is easy to learn without any special training, so there's no need to switch to another language just because your programmers don't know COBOL. Ross suggests staying with the language you have – whether it's Java or COBOL.

That puts isCOBOL in a unique position, with our close Java and COBOL connection. You don't have to throw away your COBOL programs; with Veryant you can transform them and integrate them with new and modern languages and interfaces.

Where do isCOBOL wrappers look for Java?

The compiler, `iscc.exe`, uses the value of `ISCOBOL_JDK_ROOT`, and every other wrapper uses the `ISCOBOL_JRE_ROOT` variable.

In Windows these variables are set in two files in Windows: “`pref_jre.cfg`” in the `.install4j` folder and `isshell.bat` (called by `cobcmd.bat`) in the `bin` folder.

Where do isCOBOL wrappers look for Java?

They are read in that order, with the last setting taking precedence. Both are generated during installation.

In Linux, Unix, and MacOSX these variables are set in a file named default_java.conf in the bin folder. This file is generated during installation.

If ISCOBOL_JDK_ROOT or ISCOBOL_JRE_ROOT is not set, the wrappers will look for the default Java installation. In Windows, this is returned by the “where java” command. In Linux, Unix, and Mac OSX the default Java is returned by the “which java” command.



HAVE YOU SEEN THIS?



NEW YOUTUBE VIDEOS

[All About Configuration Files](#)

[Configuring the isCOBOL Application Server](#)

[Handling Multiple Monitors in your program](#)

[Using ISMIGRATE to convert your files easily](#)

[New Features in 2022R2](#)

[Veryant End of Year Campaign](#)

[Introducing Veryant and isCOBOL Product lines](#)

[Code Analysis Reports \(CAR\) for Prospects](#)

NEW KNOWLEDGE BASE (KB) ARTICLES:

[Guide to update the isCOBOL version in your SDK](#)

[How to manage a large COBOL OCCURS on a Database with EasyDB](#)

[Guide to updating the isCOBOL software version in production environment](#)

[Working with remote projects](#)

[Guide to updating isCOBOL runtime libraries in Tomcat WebApplications](#)

[How to Access files on a different server](#)

[How to create, write, and read to files without a program](#)

[How to adjust program screens to different screen resolutions](#)



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 assistants email us at sales@veryant.com

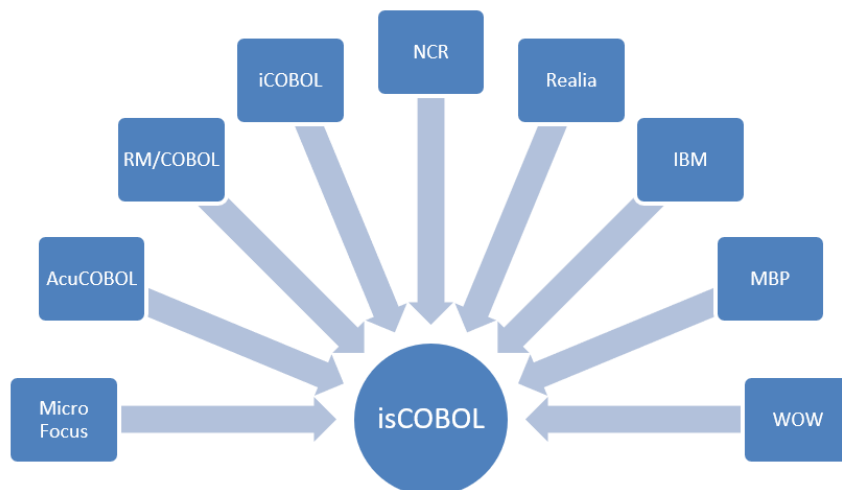
Corporate Headquarters

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, 2022R2 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

©2022 Veryant - All Rights Reserved