2021

NÚMERO 05

PUBLICACIÓN TRIANUAL DE VERYANT E ISCOBOL

veryant NEWS

OFICINA DE VERYANT EN PIACENZA, ITALIAY

Asegúrese de ver el video de escenas de la conferencia de distribuidores de 2021. Incluso si no estuvo allí, le interesará ver la oficina de Piacenza y muchas de las personas de Veryant a las que puede reconocer o con las que puede hablar con regularidad. El vestíbulo de la oficina tiene varias características únicas, como un coche colgado de la pared. Consulte la sección "¿Ha visto esto?" En la página 8

ÚNASE A NOSOTROS EN

Twitter, LinkedIn, o Facebook para estar al día con las novedades de Veryant

Vea nuestros videos de demostración y suscríbase a nuestro canal de YouTube

ESTE NÚMERO

Ya está aquí 2021 R2

Veryant se complace en anunciar la última versión de isCOBOL™ Evolve, isCOBOL Evolve 2021 Release 2.

¡La mayor adición en 2021 R2 es la capacidad de incrustar o encapsular una aplicación COBOL dentro de una página web sin modificar el COBOL! Al permitir que su isCOBOL hable con los lenguajes de las páginas web con las nuevas rutinas y control de IWC, hemos facilitado el mantenimiento de su COBOL y el trabajo con los desarrolladores de su página web para crear una vista integrada de dos entornos previamente separados.

- Ya no necesita incluir la ubicación de la carpeta/isdef para que el compilador encuentre los archivos de definición de isCOBOL. Se hace de forma automática
- Las variables de vinculación ahora pueden ser de longitud fija O variable, el runtime manejará el parámetro a medida que se pasa sin errores ni pérdida de datos.
- En el servidor de aplicaciones, puede volver a cargar cualquier programa que se encuentre en las rutas code_prefix. Esto es útil para que cuando modifique un programa, pueda forzar que se cargue en lugar de la clase anterior descargando la clase anterior de la memoria del servidor de aplicaciones. En 2021R2 puede descargar TODAS las clases y forzar al servidor de aplicaciones a comenzar con una copia nueva de todas las clases.
- Hemos aumentado significativamente el rendimiento de la impresión, tanto al realizar la impresión en un subproceso separado como al utilizar un manejo mejorado de paquetes TCP.
- 2021R2 también tiene mejoras en la codificación en la clase HTTPClient EIS y en el manejo de Json en las clases HTTPClient y HTTPHandler

Para obtener más información sobre los nuevos cambios de 2021R2, puede visitar nuestro sitio web haciendo click aquí.

Instalando el isCOBOL Server en un Docker

Arrancar el Application Server en un contenedor docker no solo es fácil sino eficiente. Descúbralo en esta guía paso a paso preparada por Valerio Biolchi, Ingeniero Senior de Soporte.

Para simplificar el uso del isCOBOL Server en un contenedor docker, hemos creado esta guía que muestra como construir, instalar y arrancar un repositorio Docker para la ejecución del isCOBOL Application Server en una máquina Linux de 64bit.

La tecnología Docker es un mecanismo para obtener una soluciones de uso inmediato y así poder proveer software en paquetes llamados contenedores, es similar a lo que ofrecería la tecnología de Plataforma-como-servicio (PAAS por sus siglas en Inglés de Platform-as-a-service). Los contenedores corren en un ambiente autocontenido e incluyen su propio software, librerías y archivos de configuración. Además, estos contenedores se pueden comunicar entre ellos mediante canales bien definidos. Dado que todos los contenedores comparten los servicios de un único kernel de sistema operativo, terminan usando mucho menos recursos que las máquinas virtuales.

Prerrequisitos

Los prerrequisitos en un computador Linux de 64bit (en donde se creará el Docker) son :

- Instalar el motor Docker (https://docs.docker.com/engine/install/) desde un repositorio
- Un ambiente SDK de isCOBOL con licencia
- YSu aplicación COBOL que corra en modo Thin Client

Archivo de Instrucciones

Comenzaremos con un archivo de instrucciones llamado “Dockerfile”. Dado que isCOBOL necesita correr sobre Java, el repositorio de isCOBOL estará basado en la imagen del OpenJDK. Todos los commandos de docker referidos aquí pueden ser encontrados en https://docs.docker.com/engine/reference/builder.

```
# Dockerfile
FROM. openjdk:11
MAINTAINER  Veryant
```

Luego usando el comando ENV, podemos definir variables de ambiente que se usarán durante la construcción del Docker y por los comandos invocados por la imagen del Docker.

```
ENV ISCOBOL=/var/isCOBOL2021R2
ENV ISCOBOL_CLASSPATH=${ISCOBOL}:${ISCOBOL}/iscontrolset
```

Con el comando RUN creamos la carpeta principal de isCOBOL dentro del contenedor docker. Este comando, ejecuta comandos en una nueva capa y crea una nueva imagen, Ejemplo: se usa habitualmente para instalar paquetes de software.

```
RUN mkdir ${ISCOBOL}
```

Ahora, para arrancar nuestra aplicación COBOL a través del isCOBOL Application Server, necesitamos copiar los archivos necesarios al contenedor Docker. El comando COPY copia nuevos archivos o carpetas desde <src> y los agrega al sistema de archivos del contenedor en la ruta <dest>::

```
COPY yourlicense.properties ${ISCOBOL}/iscobol.properties
COPY isCOBOL_SDK2021R2/lib ${ISCOBOL}/lib
COPY isCOBOL_SDK2021R2/bin/isserver ${ISCOBOL}/bin/isserver
COPY isCOBOL_SDK2021R2/sample/iscontrolset ${ISCOBOL}/iscontrolset
```

Finalmente, definimos el comando que ejecutará el isCOBOL Application Server cuando se arranque la imagen Docker. El comando CMD define comandos y/o parámetros por defecto, que pueden ser reemplazados desde la línea de comandos cuando el contenedor Docker arranque.

```
CMD ${ISCOBOL}/bin/isserver run
```

Este es el contenido completo del Dockerfile que hemos creado para obtener un contenedor que trabaje totalmente con el isCOBOL Application Server:

```
# Dockerfile
FROM  openjdk:11
MAINTAINER  Veryant
ENV ISCOBOL=/var/isCOBOL2022R1
ENV ISCOBOL_CLASSPATH=${ISCOBOL}:${ISCOBOL}/iscontrolset
RUN mkdir ${ISCOBOL}
COPY yourlicense.properties ${ISCOBOL}/iscobol.properties
COPY isCOBOL_SDK2022R1/lib ${ISCOBOL}/lib
COPY isCOBOL_SDK2022R1/bin/isserver ${ISCOBOL}/bin/isserver
COPY isCOBOL_SDK2022R1/sample/iscontrolset ${ISCOBOL}/iscontrolset
CMD ${ISCOBOL}/bin/isserver run
```

El “CMD” de la línea anterior usa la opción ‘run’ del isserver, la cual es una nueva opción en nuestra nueva versión 2022R1. Para usar la versión actual o alguna previa, se puede arrancar el isserver con un archivo de comandos (script file) run.sh como el siguiente

```
#!/bin/sh
java -cp .:${ISCOBOL}/lib/*:${ISCOBOL}/iscontrolset com.iscobol.as.AppServerImpl
wait $!
```

Luego, reemplazar la línea CMD del Dockerfile de arriba con las líneas siguientes para ejecutar el script:

```
COPY run.sh ${ISCOBOL}/bin/run.sh
CMD ${ISCOBOL}/bin/run.sh
```

Construir la imagen

Ahora podemos crear la imagen docker de "iscobol" desde la carpeta que contenga el Dockerfile y su licencia. Notar que el nombre del repositorio (iscobol en este ejemplo) debe estar todo en minúsculas.

```
root@ubuntu:/home/myDocker/myApp# ls
Dockerfile isCOBOL_SDK2021R2 yourlicense.properties
root@ubuntu:/home/myDocker/myApp# docker build -t iscobol .
```

Luego podemos ver la imagen iscobol al listar todas las imágenes de docker, lo cual muestra su nombre de repositorio y otra información adicional:

```
root@ubuntu:/home/myDocker/myApp# docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
iscobol	latest	8e66cb700e21	About a minute ago	706MB
postgres	latest	317a302c7480	4 weeks ago	374MB
mysql	5.6	f3b364958c23	6 weeks ago	303MB
ibmcom/db2	latest	ced2e4b31b7a	8 weeks ago	2.97GB
mcr.microsoft.com/mssql/server	2019-latest	80bdc8efc889	8 weeks ago	1.55GB
store/oracle/database-enterprise	2019-latest	12a359cd0528	4 years ago	3.44GB

Crear el Contenedor

A continuación queremos crear un contenedor modificable llamado "iscobolserver", basado en la imagen "iscobol". Hacemos esto al correr el commando especificado con la etiqueta CMD del Dockerfile.
A veces, se puede necesitar definir algunas reglas de red para facilitar la interacción entre contenedores en aplicaciones multi-contenedor o hacer que sus puertos Docker sean accesibles al mundo exterior. Una manera es usar la bandera -p o -P en el comando de ejecución del Docker para conectar el puerto interno por defecto del isCOBOL Server, el 10999, a un puerto externo 10999. A continuación un comando para ejecutar el docker, especificando el nombre del contenedor, indicando la forma de emparejar el puerto interno con el externo, y especificando la imagen a usar para crear el contenedor Docker:

```
docker run --name iscobolserver -p 10999:10999 -d iscobol
```

Ahora podemos verificar el estado del contenedor “iscobolserver”.

```
root@ubuntu:/home/myDocker/myApp# docker ps -a
```

```
CONTAINER ID   IMAGE     COMMAND                  CREATED         STATUS
6e8c9e89ffb3   iscobol   “/bin/sh -c ‘${ISCOB…”   3 seconds ago   Up 2 seconds

                                PORTS                                                 NAMES
                                0.0.0.0:10999->10999/tcp, :::10999->10999/tcp         iscobolserver
```

Allí muestra que el isCOBOL Application Server está escuchando en el puerto 10999 de la computadora en donde está corriendo el docker y que todos los clientes delgados (thin-client) de isCOBOL pueden usar ese servicio.

Detener el servicio

Podemos detener el contenedor y terminar la ejecución del isCOBOL Application Server con este comando:

```
root@ubuntu:/home/myDocker/myApp# docker stop iscobolserver
```

Al verificar de nuevo el proceso docker podemos ver que su estado es “Exited”:

```
root@ubuntu:/home/myDocker/myApp# docker ps -a
```

```
CONTAINER ID   IMAGE     COMMAND                  CREATED         STATUS
6e8c9e89ffb3   iscobol   “/bin/sh -c ‘${ISCOB…”   28 minutes ago  Exited (137) About a minute ago

                                                                  PORTS          NAMES
                                                                                 iscobolserver
```

Qué sigue?

Espere nuestro artículo de seguimiento en el próximo boletín en donde discutiremos como crear Volúmenes Docker como mecanismo persistente de datos de isCOBOL, en un contenedor o compartido entre varios contenedores.

Destacados en la documentación

Una historia del software de Veryant

El Release Overview book ien la documentación de Veryant incluye aspectos destacados de las nuevas funciones introducidas por la versión actual, pero también incluye aspectos destacados de las funciones introducidas en versiones anteriores.

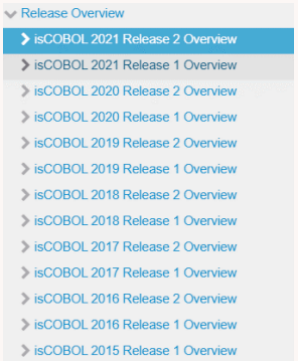
¿Te perdiste un par de versiones y quieres saber qué cambió antes de actualizar a la actual?

¿Quieres hacer un seguimiento de la evolución de un producto específico de una versión a otra?

¿Estás interesado en saber cuándo se presentó un producto específico?

E Release Overview book es el mejor lugar para este tipo de lectura. La colección de documentos de descripción general de la versión en los rangos de documentación

6 años de historia!



Impresión más rápida en 2021R2

El rendimiento de los trabajos de impresión se ha mejorado en 2021R2, como resultado de la implementación del modo asíncrono a través de las nuevas configuraciones:

iscobol.print.spooler_async=true|false (predeterminado: true) para configurar el trabajo de impresión para que se ejecute de forma asincrónica
iscobol.print.pdf_async=true|false (predeterminado: false) para que el trabajo de impresión en PDF se ejecute de forma asincrónica.

y con un mejor manejo de paquetes TCP del servidor de aplicaciones cuando se ejecuta en ThinClient. Este gráfico muestra las ganancias de la impresión con la nueva versión 2021R2 en comparación con la anterior 2021R1.

Standalone tests	2021R1	2021R2
character print job with 200 pages on -P SPOOLER	18,27	6,35
graphical print job with 20 pages on -P SPOOLER	11,37	4,38
ThinClient tests	**2021R1**	**2021R2**
character print job with 200 pages on -P SPOOLER	16,93	1,75
graphical print job with 20 pages on -P SPOOLER	69,80	14,75
character print job with 200 pages on -P PDF	17,53	13,70
graphical print job with 20 pages on -P PDF	64,86	22,05
character print job with 200 pages on -P PREVIEW	6,79	2,04
graphical print job with 20 pages on -P PREVIEW	59,10	14,25

Todos los tiempos están en segundos. Detalles de hardware de la máquina cliente:
Windows 10 Pro i7-8550U CPU @ 1.80GHz 16GB
Detalles del hardware de la máquina del servidor: macOS Big Sur Apple M1 16GB.

HAS VISTO ESTO?

NUEVOS VIDEOS EN YOUTUBE

2021R1 New Features

WebClient 2021R1 Demo

DatabaseBridge (from the archives)

Veryant's Distributor's Meeting 2021

2021R2 New Features

Veryant's Load Balancer

NUEVOS ARTÍCULOS EN LA BASE DE CONOCIMIENTO (KB):

Can I define one or more data items based on the definition of another one?

Modernizing your COBOL application by using isCOBOL Compiler code injection

Why start using the OCCURS DYNAMIC? How to save memory by replacing your fixed arrays

Un gran paso adelante para la habilitación WEB

Tenemos noticias interesantes sobre la nueva capacidad de combinar el desarrollo de HTML / JavaScript con sus aplicaciones de escritorio isCOBOL a través de WebClient. Esto es útil para desarrollar páginas web personalizadas con áreas donde las aplicaciones isCOBOL pueden ejecutarse e interactuar con la página web encapsulada y desarrollar aplicaciones de escritorio isCOBOL con componentes web personalizados integrados en la pantalla de la aplicación.

Al encapsular su aplicación en una página web, todo en el lado web se hace con JavaScript utilizando una página web contenedora. El lado del cliente usa las nuevas rutinas de la biblioteca IWC$ de isCOBOL.

IsCOBOL WebClient ahora permite el intercambio de mensajes JavaScript entre la aplicación de escritorio isCOBOL y la página web subyacente. Su código iniciará y detendrá el motor de mensajería utilizando las nuevas rutinas de biblioteca IWC$ INIT e IWC$ STOP. La aplicación isCOBOL envía un mensaje a la página web utilizando la rutina IWC$ SEND. Cuando la página web envía un mensaje a la aplicación isCOBOL, se recibe mediante la rutina IWC$ GET.

Aquí tienes un ejemplo de cómo usarías esas 4 rutinas:

```
78  78-iwc-crt-status           value 1001.
77  data-to-send                pic x any length.
01  iwc-struct.
    03 iwc-action                pic x any length.
    03 iwc-data                  pic x any length.
    03 iwc-bytes                 pic x any length.

ACTIVATE.
    call "IWC$INIT" using 78-iwc-crt-status
SEND-TO-HTML.
    initialize iwc-struct.
    move "ComSample"  to iwc-action
    move data-to-send to iwc-data
    call "IWC$SEND"   using iwc-struct
READ-DATA-FROM-HTML.
    initialize iwc-action
    call "IWC$GET" using iwc-struct
    if iwc-action = "EXECUTE_PGM"
       call IWC-DATA
    end-if.
DEACTIVATE.
    call "IWC$STOP" giving return-code.
```

Puede encontrar un ejemplo detallado en el folder $ISCOBOL/samples/webclient/encapsulated/source , que utiliza puntos de entrada en IWC.cbl para responder a los mensajes. Otro ejemplo está en $ISCOBOL/samples/issamples/s-routines/IWC.

Si, por otro lado, desea incluir un componente web escrito en HTML / JavaScript en la pantalla de su aplicación isCOBOL cuando se ejecuta en el WebClient, debe utilizar el nuevo control GUI IWC-PANEL. A continuación, se muestra un ejemplo de la descripción de la sección de pantalla de ese control:

```
03 f-map iwc-panel
         js-name                "f-map"
         line  5                column 2
         size  68 cells         lines  15 cells
         value                  fmap-struct
         event procedure        FMAP-PROC.
```

Este control es fácil de usar tanto para el programador COBOL como para el programador JavaScript, porque no hay un cruce de lenguajes : los programadores no tienen que usar un lenguaje desconocido. Si el programador COBOL desea enviar un mensaje desde el control GUI IWC-PANEL, usaría la instrucción MODIFY, y si desea recibir un mensaje, usaría la instrucción INQUIRE.

Un programa de muestra que utiliza IWC-PANEL está en el programa

$ISCOBOL/samples/issamples/s-gui/IWC-PANEL.cbl.

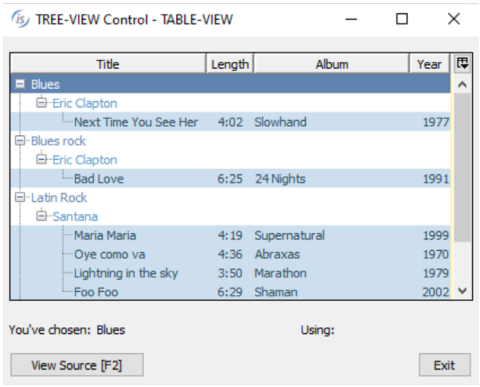
Para obtener más información sobre estas capacidades, puede:

- mirar los ejemplos ya mencionados
- leer la documentación aquí.
- Vea un video de demostración de WebClient encapsulado aquí.
- Vea una demostración de las nuevas funciones en el video de Nuevas funciones de 2021R2 aquí.

Actualizaciones de un nuevo estilo

Cuando se establece este estilo, los elementos pueden tener varios valores en lugar de un solo valor.
Cada valor se muestra en una columna separada. Las columnas se definen mediante la propiedad Display-Columns.
Ejemplo: definir una vista de tabla de árbol con 3 columnas:

```
screen section.
  ...
  03 screen-1-tv-1 Tree-View
     line 2.7
     column 3.4
     size 20.8 cells
     lines 29.1 cells
     height-in-cells
     color 144
     id 2
     table-view
     display-columns
          (1, 10, 15)
```



Cómo cambiar el font predeterminado de la impresora

Los programas de impresión simples que producen una salida de solo texto sin llamar a ninguna rutina de biblioteca de impresión específica no establecen un font específico para el trabajo de impresión.

El Font se deja sin definir, por lo que es deber de la impresora elegir qué font se debe utilizar.

Esto puede producir una salida diferente cuando se imprime en diferentes impresoras, así como una salida no deseada cuando el font elegido tien un ancho fijo.

Para evitar este tipo de problemas, puede establecer la propiedad de configuración *iscobol.print.default_font*

Con esta propiedad, especifica la fuente que debe usarse de forma predeterminada cuando el programa no fuerza ninguna fuente a través de las rutinas de la biblioteca de impresión.

Por ejemplo, para tener Consolas tamaño 12 por defecto, configure:

iscobol.print.default_font= Consolas-12

veryant

ARCHIVOS TEMPORALES EN MEMORIA

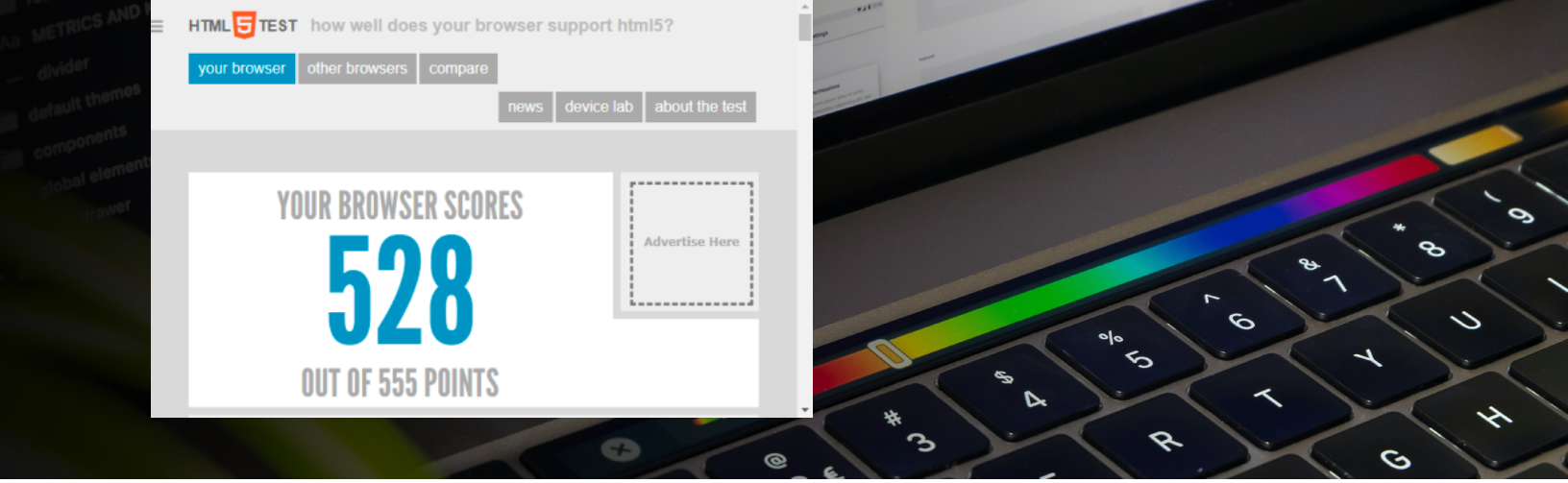
Tome el control de su rendimiento de acceso a archivos temporales almacenando sus datos temporales en la memoria en lugar de en el disco.

Es una práctica habitual en las aplicaciones COBOL utilizar archivos secuenciales temporales. En algunos casos, acceder a estos archivos podría afectar negativamente el rendimiento de las aplicaciones porque elruntime necesita ir y venir para acceder a estos archivos donde están almacenados en el disco.

Afortunadamente, existe un método para crear y acceder a esos archivos secuenciales temporales en la memoria, lo que mejora el rendimiento de sus programas. Esta solución es muy fácil de implementar: solo necesita definir el archivo dentro del programa de la siguiente manera:

```
INPUT-OUTPUT SECTION.
FILE-CONTROL.
select my-file
       assign to address
            memory-area
....
WORKING-STORAGE SECTION.
77 memory-area
       pic x any length.
```

La variable “área de memoria” contendrá todos los registros insertados en el archivo secuencial “mi-archivo”, lo que agilizará el acceso.
Para obtener más información y un programa de ejemplo, consulte el artículo de KB aquí.



NUEVO WEB-BROWSER CHROMIUM

Agregamos una tercera implementación de navegador web para adaptarse a la sintaxis compleja de JavaScript y CSS.

El control del navegador web siempre le ha dado la opción de dos implementaciones diferentes: el componente **DJBrowser** basado en SWT o el componente **JavaFX Webview**. Puede elegir qué implementación estableciendo la propiedad de configuración iscobol.gui.webbrowser.class.
Sin embargo, ninguna de estas dos implementaciones es completamente comparable a un navegador web real como Firefox, Chrome o Edge. Estas dos implementaciones de navegador web son buenas para representar la mayoría de las páginas html, pero es posible que las páginas con sintaxis mas moderna y compleja de JavaScript o CSS no funcionen correctamente.

Por lo tanto, agregamos una tercera implementación de navegador web que se puede activar a través de la propiedad de configuración *iscobol.gui.webbrowser.class*: **JxBrowser**.
Este es un poderoso navegador web basado en Chromium, capaz de renderizar todo el contenido html de la misma manera que los navegadores Chrome y Edge
TEl JxBrowser se puede descargar del sitio web de TeamDev. Ver aqui ien la documentación de isCOBOL para obtener más detalles.

Por qué empezar a utilizar la sintaxis de OCCURS

Cada aplicación COBOL tiene muchos programas que contienen definiciones con OCCURS en las secciones Working Storage o Linkage. Estas matrices suelen tener un tamaño fijo para acomodar el número máximo de referencias posibles en la tabla. El resultado es que la memoria consumida por la matriz es fija, incluso cuando el programa no necesita usar todas las ocurrencias. Por ejemplo, esta definición:

```
78  max-element     value 900.
01  w-group.
   03 w-element     occurs max-element.
        05 w-cod    pic 9(3).
        05 w-desc   pic x(20).
        05 w-note   pic x(100).
```

asignará la memoria para las 900 ocurrencias en el inicio del programa, incluso si el programa utilizará solo unas pocas referencias.
Para reducir la memoria consumida al asignar solo la memoria realmente necesaria, puede cambiar su código para usar una dinámica. Por ejemplo:

```
01  w-group.
   03 w-element     occurs dynamic capacity max-element.
        05 w-cod    pic 9(3).
        05 w-desc   pic x(20).
        05 w-note   pic x(100).
```

Para mayor información y programas de ejemplo que demuestren cómo funcionan las matrices dinámicas, consulte the KB article here.

Evolution, without revolution

Contáctenos

Para clientes con soporte, envíenos un correo electrónico a **support@veryant.com**

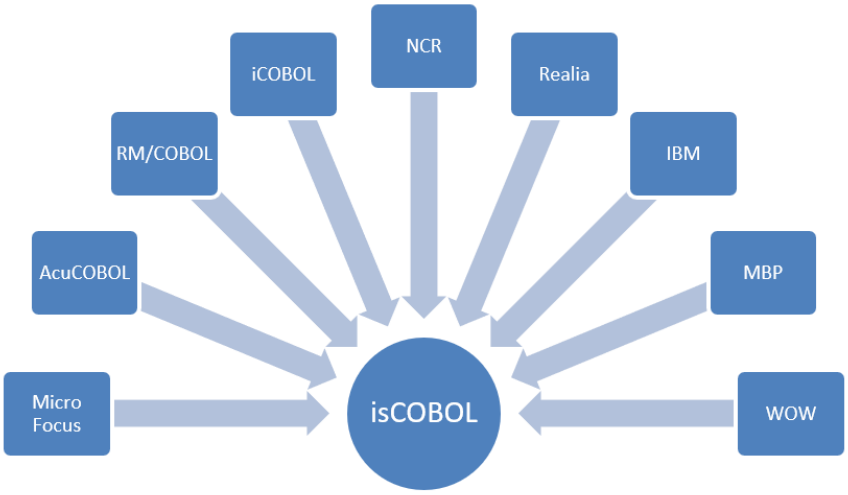
Si desea que Veryant se comunique con usted para programar una sesión informativa técnica sobre el producto, envíenos un correo electrónico **info@veryant.com**

Si desea que Veryant se comunique con usted para obtener una cotización especial o asistentes de ventas, envíenos un correo electrónico a **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

Como siempre, 2021R1 contiene múltiples adiciones de compatibilidad, a medida que continuamos haciendo que su proceso de conversión sea lo más fluido, rápido y sencillo posible.



veryant.com

Follow **Veryant** on

veryant.com