EDIÇÃO 05
2021
REVISTA TRIANUAL DE VERYANT E isCOBOL

veryant NEWS

NESTA EDIÇÃO

DO ESCRITÓRIO DA VERYANT EM PIACENZA, ITÁLIA

Certifique-se de assistir ao vídeo de cenas da conferência do Distribuidor de 2021.

Mesmo se você não estivesse lá, você estaria interessado em ver o escritório de Piacenza e muitas das pessoas da Veryant que você pode reconhecer ou conversar regularmente.

O saguão do escritório tem várias características exclusivas, incluindo um carro pendurado na parede. Consulte a seção “Você já viu isto?” Na página 8.

POR FAVOR NOS SIGA NO

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

Assista a nossos vídeos de demonstração e inscreva-se em nosso canal no YouTube

2021 R2 chegou

Veryant tem o prazer de anunciar o lançamento mais recente do isCOBOL ™ Evolve, isCOBOL Evolve 2021 Versão 2.

A maior adição na versão 2021 R2 é a capacidade de incorporar ou encapsular um aplicativo COBOL em uma página da web sem modificar o COBOL! Ao permitir que seu isCOBOL converse com idiomas de página da web com as novas rotinas IWC e controle de GUI, tornamos mais fácil manter seu COBOL e trabalhar com seus desenvolvedores de página da web para fazer uma visão integrada de dois ambientes anteriormente separados.

- Você não precisa mais incluir a localização da pasta / isdef para que o compilador encontre os arquivos de definição de isCOBOL. Isso é feito automaticamente.
- As variáveis de ligação agora podem ser fixas OU de comprimento variável, o tempo de execução tratará do parâmetro conforme ele é passado sem um erro ou dados perdidos.
- No servidor de aplicativos, você pode recarregar qualquer programa encontrado nos caminhos code_prefix. Isso é útil quando você modifica um programa e deseja forçá-lo a ser carregado no lugar da classe antiga, descarregando a classe antiga da memória do servidor de aplicativos. No 2021R2, você pode descarregar TODAS as classes e forçar o servidor de aplicativos a iniciar com uma nova cópia de todas as classes.
- Aumentamos o desempenho da impressão significativamente, tanto ao realizar a impressão em um thread separado, quanto ao usar o manuseio de pacotes TCP aprimorado.
- 2021R2 também tem melhorias na codificação na classe HTTPClient EIS e no tratamento de Json nas classes HTTPClient e HTTPHandler.
- Para obter mais informações sobre as novas alterações do 2021R2, você pode acessar nosso website clicando aqui.

Instale o isCOBOL server usando Docker

Iniciando o servidor de aplicativos em um contêiner docker é fácil e eficiente. Aqui está um guia passo a passo do Engenheiro de Suporte Sênior, Valerio Biolchi

A fim de simplificar o processo de uso do isCOBOL Server em um contêiner docker, criamos um guia para mostrar como construir, instalar e iniciar um repositório Docker para executar um isCOBOL Application Server em uma máquina Linux de 64 bits.

Docker é uma maneira de fornecer soluções drop-in para entregar software em pacotes chamados contêineres, semelhantes ao que a plataforma como serviço (PAAS) ofereceria. Os contêineres são executados em um ambiente isolado e contêm seus próprios softwares, bibliotecas e arquivos de configuração. Os contêineres podem se comunicar entre si por meio de canais bem definidos. Como todos os contêineres compartilham os serviços de um único núcleo do sistema operacional, eles usam menos recursos do que as máquinas virtuais.

Pré-requisitos

Os pré-requisitos em um computador Linux de 64 bits (onde o docker é construído) são:

- O mecanismo Docker (https://docs.docker.com/engine/install/) instalado de um repositório
- Um ambiente licenciado isCOBOL SDK
- Seu aplicativo COBOL em execução no modo Thin Client

Arquivo de Instrução

Começaremos com um arquivo de instrução denominado "Dockerfile". Como isCOBOL precisa do Java para ser executado, o repositório isCOBOL será baseado na imagem OpenJDK. Todas as referências de comando docker aqui podem ser encontradas em https://docs.docker.com/engine/reference/builder.

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

Então, podemos definir as variáveis de ambiente a serem usadas durante a construção e a partir dos comandos invocados pela imagem docker com ENV.

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

O comando RUN cria a pasta principal isCOBOL dentro do contêiner do docker. RUN executa comandos em uma nova camada e cria uma nova imagem. Por exemplo, E.g., é frequentemente usado para instalar pacotes de software.

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

Agora, para iniciar nosso aplicativo Cobol através do servidor de aplicativos isCOBOL, precisamos copiar os arquivos apropriados para o contêiner do docker. COPY copia novos arquivos ou diretórios de <src> e os adiciona ao sistema de arquivos do contêiner no caminho <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
```

Por fim, definimos o comando que executará o servidor de aplicativos isCOBOL quando a imagem do docker for iniciada. O CMD define comandos e / ou parâmetros padrão, que podem ser substituídos na linha de comando quando o docker container é executado.

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

Aqui está o conteúdo completo do Dockerfile que criamos para ter um contêiner funcionando totalmente com o isCOBOL 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
```

A linha “CMD” acima usa o comando ‘run’ do isserver, que é novo em nossa próxima versão, 2022R1. Para usar a versão atual ou anterior, você pode iniciar o isserver com um arquivo de script como run.

sh abaixo:

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

Em seguida, substitua a linha CMD no Dockerfile acima por estas linhas para copiar e executar o script:

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

Construindo a imagem

Em seguida, podemos construir a imagem docker "iscobol" da pasta que contém o Dockerfile e sua licença. Observe que o nome do repositório (iscobol neste exemplo) deve ser minúsculo.

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

Agora podemos ver a imagem iscobol listando todas as imagens do docker, mostrando o nome do repositório e outras informações:

```
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

Crie o contêiner

Agora queremos criar um contêiner gravável chamado "iscobolserver" baseado na imagem "iscobol". Fazemos isso executando o comando especificado como a tag CMD no Dockerfile.

Às vezes, você pode precisar definir algumas regras de rede para permitir uma interação suave entre contêineres em aplicativos de vários contêineres ou tornar suas portas Docker acessíveis por serviços no mundo externo. Uma opção é usar o sinalizador -p ou o sinalizador -P na string de execução do Docker para publicar a porta padrão 10999 do servidor isCOBOL interno para 10999 externo. Aqui está um comando para executar o docker, especificando o nome do contêiner, combinando as portas internas e externas e especificando a imagem a ser usada para criar o contêiner:

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

Agora podemos verificar o status do nosso contêiner "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
```

Isso mostra que o isCOBOL Application Server está ouvindo na porta 10999 do computador onde o docker o está executando e todos os usuários do isCOBOL thin client podem usar esse serviço.

Parando o serviço

Podemos parar o contêiner e encerrar a execução do
Servidor de aplicativos isCOBOL com este comando:

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

Verificando o processo do docker novamente, podemos ver que seu status é "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
```

O que vem a seguir?

Leia nosso artigo no próximo boletim informativo, onde discutiremos como criar volumes Docker como um mecanismo para seus dados isCOBOL persistentes em um contêiner ou compartilhados entre contêineres.

Documentações em destaque

Uma história sobre Veryant's software

O Release Overview Book da versão no conjunto de documentação do Veryant inclui destaques dos novos recursos introduzidos pela versão atual, mas também inclui os destaques dos recursos introduzidos pelas versões anteriores.

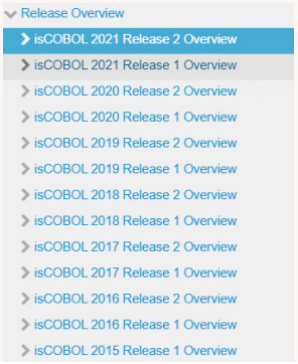
Você perdeu alguns lançamentos e quer saber o que mudou antes de atualizar para o atual?

Você deseja acompanhar como um produto específico evoluiu de uma versão para outra?

Você está interessado em saber quando um produto específico foi lançado?

O Release Overview Book da versão é o melhor lugar para esse tipo de leitura. A coleção de documentos de Visão geral da versão na documentação varia de 2015 até agora:

6 anos de história!



Impressão mais rápida em 2021R2

O desempenho dos trabalhos de impressão foi melhorado no 2021R2, tanto como resultado da implementação do modo assíncrono através das novas configurações:

iscobol.print.spooler_async=true|false (default: true) para definir o trabalho de impressão para ser executado de forma assíncrona

iscobol.print.pdf_async=true|false (default: false)para que o trabalho de impressão PDF seja executado de forma assíncrona.

E com melhor tratamento de pacotes TCP do servidor de aplicativos quando executado em ThinClient. Este gráfico mostra os ganhos de impressão usando a nova versão do 2021R2 em comparação com o 2021R1 anterior.

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 os tempos são em segundos. Detalhes de hardware da máquina cliente:
Windows 10 Pro i7-8550U CPU @ 1,80 GHz 16 GB
Detalhes de hardware da máquina servidor: macOS Big Sur Apple M1 16GB.

VOCÊ JÁ VIU ISSO?

NOVOS VIDEOS NO YOUTUBE

2021R1 New Features

WebClient 2021R1 Demo

DatabaseBridge (from the archives)

Veryant's Distributor's Meeting 2021

2021R2 New Features

Veryant's Load Balancer

ARTIGOS DA NOVA BASE DE CONHECIMENTO (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

Um salto gigante para a capacitação WEB

Temos notícias empolgantes sobre o novo recurso de combinar desenvolvimento HTML / JavaScript com seus aplicativos de desktop isCOBOL por meio do WebClient. Isso é útil

- para desenvolver páginas da web personalizadas com áreas onde aplicativos isCOBOL podem ser executados e interagir com a página da web de encapsulamento
- para desenvolver aplicativos de desktop isCOBOL com componentes da Web personalizados incorporados na tela do aplicativo.

Ao encapsular seu aplicativo em uma página da web, tudo no lado da web é feito com JavaScript usando uma página da web de contêiner. O lado do cliente usa as novas rotinas de biblioteca IWC $ do isCOBOL.

O isCOBOL WebClient agora permite o intercâmbio de mensagens JavaScript entre o aplicativo de desktop isCOBOL e a página da web subjacente. Seu código iniciará e parará o mecanismo de sistema de mensagens usando as novas rotinas de biblioteca IWC$INIT e IWC$STOP. O aplicativo isCOBOL envia uma mensagem para a página da web usando a rotina IWC$SEND. Quando a página da web envia uma mensagem para o aplicativo isCOBOL, ela é recebida usando a rotina IWC$GET.

Aqui está um exemplo de como você usaria essas 4 rotinas:

```
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.
```

Você pode encontrar um exemplo detalhado na pasta $ISCOBOL/samples/webclient/encapsulated/source, que usa pontos de entrada em IWC.cbl para responder às mensagens. Outra amostra está em $ISCOBOL/samples/issamples/s-routines/IWC.

Se, por outro lado, você deseja incluir um componente da web escrito em HTML / JavaScript na tela do aplicativo isCOBOL quando executado no WebClient, você deve usar o novo controle IWC-PANEL GUI. Aqui está um exemplo da descrição da seção da tela desse controle:

```
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 controle é fácil de usar para o programador COBOL e o programador JavaScript, porque não há cruzamento de linguagem - os programadores não precisam usar uma linguagem desconhecida. Se o programador COBOL quiser enviar uma mensagem do controle IWC-PANEL GUI, ele usará a instrução MODIFY e, se quiser receber uma mensagem, usará a instrução INQUIRE.

Um programa de amostra usando o IWC-PANEL está no $ISCOBOL/samples/issamples/s-gui/IWC-PANEL.cbl programa.

Para saber mais sobre esses recursos, você pode

olhe para os exemplos já mencionados

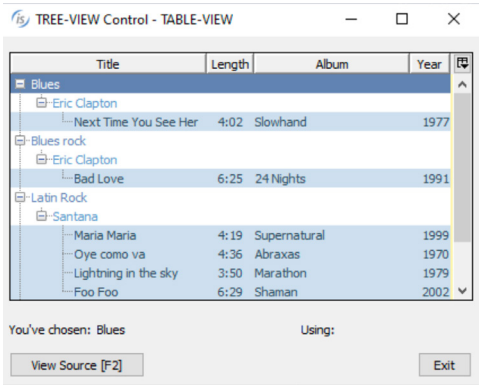
- leia a documentação aqui.
- Assista a um vídeo demonstrando WebClient encapsulado aqui.
- Assista a uma demonstração dos novos recursos no vídeo 2021R2 Novos recursos aqui.

Um novo estilo atualiza e expande seus controles de Tree-View

Quando esse estilo é definido, os itens podem ter vários valores em vez de um único valor.
Cada valor é exibido em uma coluna separada. As colunas são definidas pela propriedade Display-Columns.

Exemplo - Defina uma tabela Tree-View em 3 colunas:

```
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)
```



Como alterar a fonte padrão da impressora

Programas de impressão simples que produzem uma saída apenas de texto sem chamar qualquer rotina de biblioteca de impressão específica não definem uma fonte específica para o trabalho de impressão..

A fonte é deixada indefinida, portanto é tarefa da impressora escolher qual fonte deve ser usada.

Isso pode produzir uma saída diferente ao imprimir em impressoras diferentes, bem como uma saída indesejada quando a fonte escolhida não tem um pitch fixo.

Para evitar esse tipo de problema, você pode definir a propriedade de configuração iscobol.print.default_font

Com esta propriedade, você especifica a fonte que deve ser usada por padrão quando o programa não força nenhuma fonte por meio de rotinas de biblioteca de impressão.

Por exemplo, para ter Consolas tamanho 12 como padrão, defina:

iscobol.print.default_font= Consolas-12

veryant

ARQUIVOS TEMPORÁRIOS NA MEMÓRIA

Assuma o controle de seu desempenho de acesso a arquivos temporários, armazenando seus dados temporários na memória em vez de no disco.

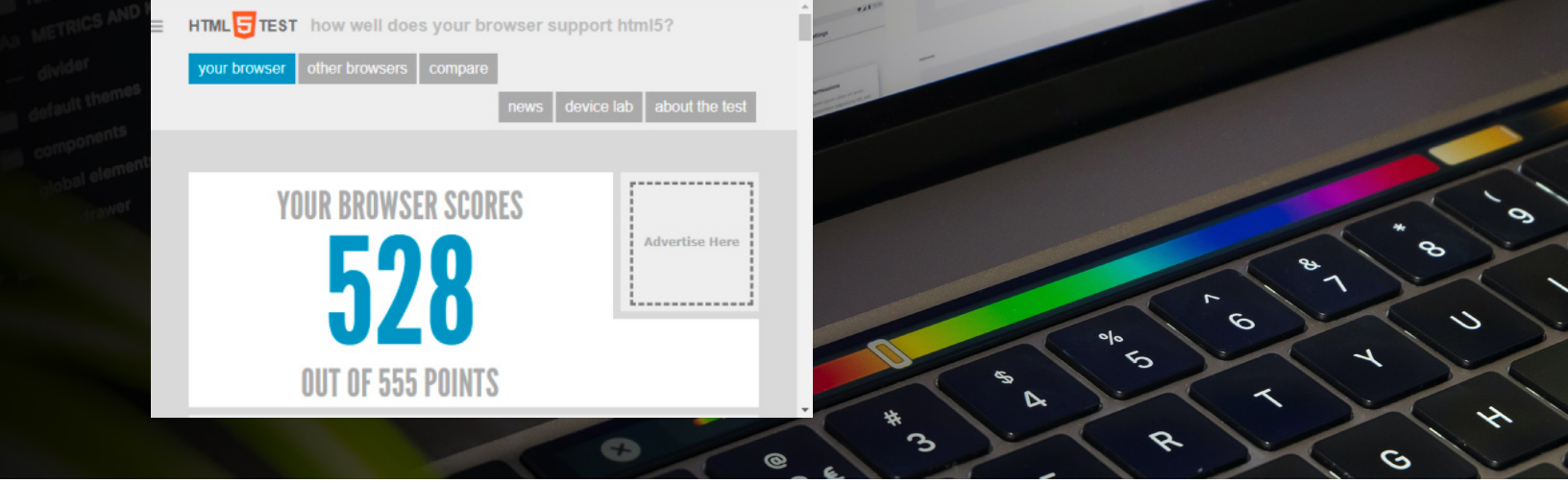
É uma prática regular em aplicativos COBOL usar arquivos sequenciais temporários. Em alguns casos, o acesso a esses arquivos pode afetar negativamente o desempenho dos aplicativos porque o tempo de execução precisa ir e voltar para acessar onde estão armazenados esses arquivos no disco.

Felizmente, existe um método para criar e acessar esses arquivos sequenciais temporários na memória, o que melhora o desempenho de seus programas. Esta solução é muito fácil de implementar: você só precisa definir o arquivo dentro do programa da seguinte maneira:

```
f k mr q J l r q mr q ╤ b ` q f l k K
c f i b J ` l k q o l i K
ë É ä É Å í = ã ó J Ñ á ä É =
═══~ ë ë á Ö å ═ í ç ═~ Ç Ç ê É ë ë =
═══════════ã É ã ç ê ó J ~ ê É ~
K K K K
t l o h f k d J p q l o ^ d b ╤ b ` q f l k K
T T = ã É ã ç ê ó J ~ ê É ~ =
═══é á Å ═ñ ═~ å ó ═ä É å Ö í Ü K
```

A variável "memory-area" conterá todos os registros inseridos no arquivo sequencial "my-file", tornando seu acesso mais rápido.

Para obter mais informações e um programa de amostra, consulte o artigo da base de conhecimento aqui.



NOVO NAVEGADOR WEB CHROMIUM

Adicionamos uma terceira implementação de navegador da web para acomodar a sintaxe complexa de Javascript e CSS

O controle do navegador da Web sempre deu a você a escolha de duas implementações diferentes: o componente DJBrowser baseado em SWT ou o componente JavaFX Webview. Você pode escolher qual implementação configurando o iscobol.gui. webbrowser. propriedade de configuração de classe.

No entanto, nenhuma dessas duas implementações é totalmente comparável a um navegador real, como Firefox, Chrome ou Edge. Essas duas implementações de navegador da web são boas para renderizar a maioria das páginas html, mas páginas com JavaScript ou sintaxe CSS moderna e complexa podem não funcionar corretamente.

Portanto, adicionamos uma terceira implementação de navegador da web que você pode ativar por meio do iscobol.gui.webbrowser.class Propriedade de configuração: JxBrowser.

Este é um navegador da web poderoso baseado em Chromium, capaz de renderizar todo o conteúdo html da mesma forma que os navegadores Chrome e Edge.

O JxBrowser pode ser baixado do site da TeamDev. Veja aqui na documentação isCOBOL para mais detalhes.

Por que começar a usar a sintaxe OCCURS DYNAMIC

Cada aplicativo COBOL tem muitos programas contendo definições com OCCURS nas seções Working Storage ou Linkage. Essas matrizes geralmente têm um tamanho fixo para acomodar o número máximo de referências possíveis na tabela. O resultado é que a memória consumida pelo array é fixa, mesmo quando o programa não precisa usar todas as ocorrências. Por exemplo, esta definição:

```
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).
```

irá alocar a memória para todas as 900 ocorrências na inicialização do programa, mesmo se o programa usar apenas algumas referências.
Para reduzir a memória consumida alocando apenas a memória realmente necessária, você pode alterar seu código para usar uma ocorrência dinâmica. Por exemplo:

```
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 obter mais informações e programas de amostra que demonstram como as matrizes dinâmicas funcionam, consulte o artigo KB aqui.

Evolution, without revolution

Nos Contate

Para clientes com suporte, envie-nos um e-mail para
support@veryant.com

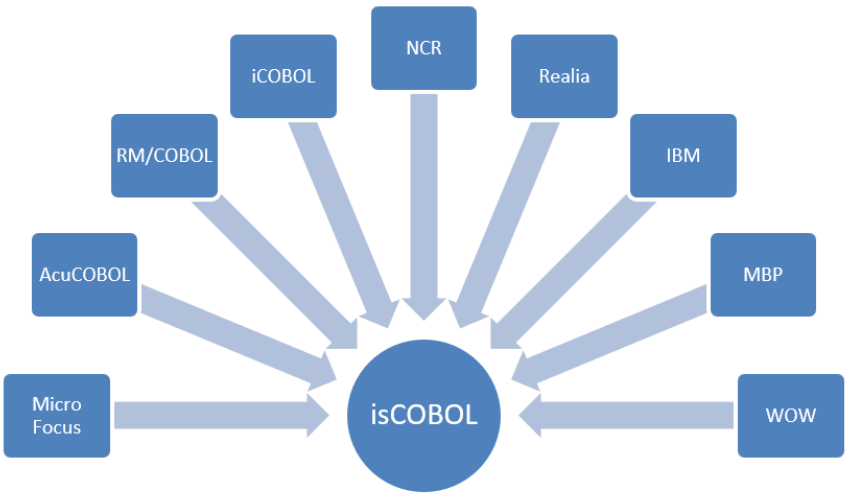
Se desejar que a Veryant entre em contato com você para agendar um briefing técnico do produto, envie-nos um e-mail para
info@veryant.com

Se desejar que a Veryant entre em contato com você para obter uma cotação especial ou os assistentes de vendas, envie-nos um e-mail para
sales@veryant.com

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

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

Como sempre, 2021R2 contém vários acréscimos de compatibilidade - à medida que continuamos a fazer seu processo de conversão o mais suave, rápido e indolor possível.



veryant.com

Follow **Veryant** on

veryant.com