isCOBOL™ Evolve

isCOBOL Evolve 2018 Release 2 Overview

Descripción de isCOBOL Evolve 2018 Release 2

Introducción

Veryant se complace en anunciar la última versión de isCOBOL ™ Evolve, isCOBOL Evolve 2018 R2.

isCOBOL Evolve proporciona un entorno completo para el desarrollo, la implementación, el mantenimiento y la modernización de las aplicaciones COBOL.

isCOBOL 2018 R2 incluye varias mejoras en los controles de la GUI, como búsqueda automática y filtro en el control de cuadrícula, y otras características nuevas y opciones de compatibilidad.

La productividad de los desarrolladores se ha mejorado al proporcionar una mejor navegación de códigos durante la edición en los programas IDE y de depuración, y se han implementado nuevas funciones de depuración. isCOBOL EIS se ha mejorado para permitir una integración más estrecha con programas externos.

También se ha mejorado el soporte para los mercados asiáticos, lo que permite una migración fácil a isCOBOL.

Los detalles sobre estas mejoras y actualizaciones se incluyen a continuación.

Mejoras en isCOBOL IDE

isCOBOL 2018 R2 IDE mejora el editor al mostrar información adicional útil a los desarrolladores de COBOL y al mejorar la usabilidad de las funciones existentes..

Funciones del editor

Al abrir un archivo fuente ahora mostrará información importante en la ventana "Properties", que se encuentra por defecto en la parte inferior derecha de la ventana IDE, como por ejemplo:

- una lista de copybooks utilizados

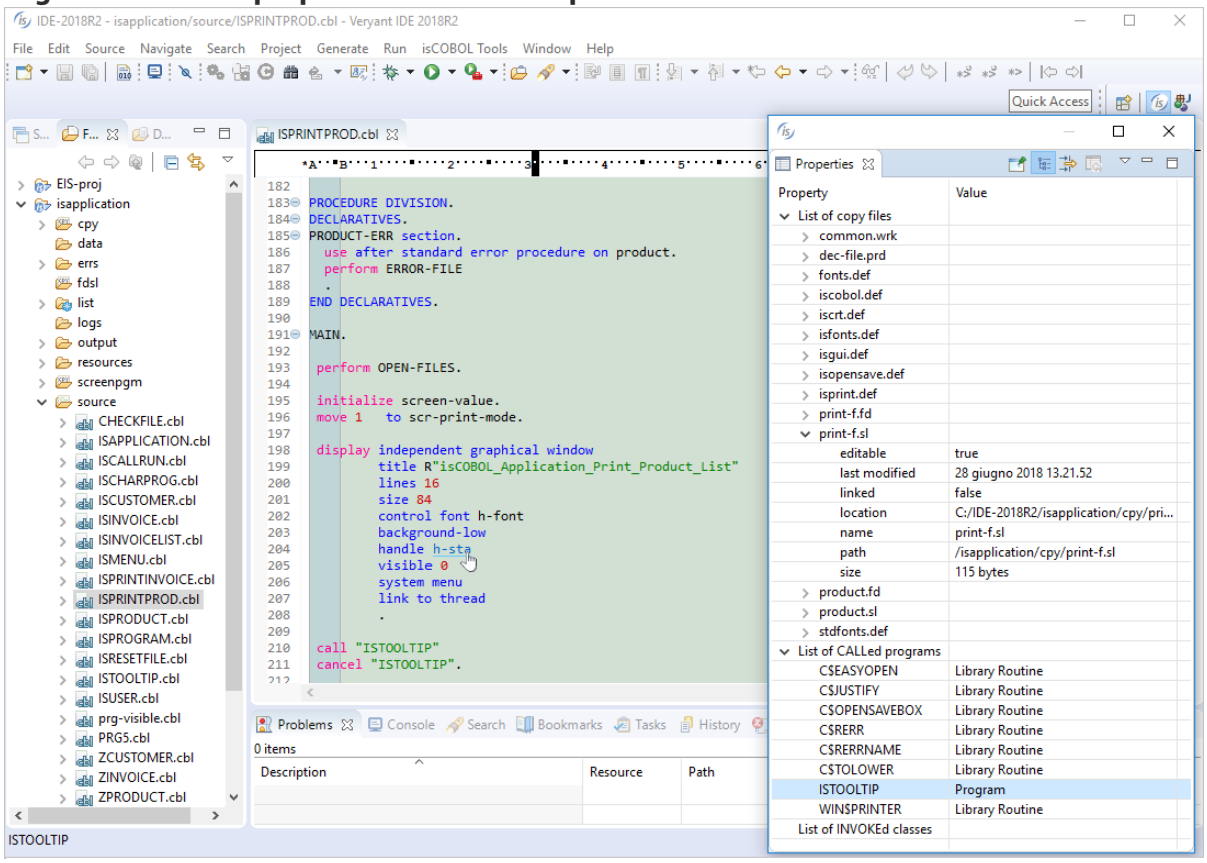
- una lista de programas llamados

- una lista de clases invocadas

Al hacer doble clic en una línea en la ventana de propiedades, el copybook o programa seleccionado se abrirá en el Editor si la fuente está disponible en el workspace actual. Por ejemplo, haciendo doble clic en la línea seleccionada que se muestra en la Figura 1, Ventana propiedades del IDE e Hipervínculo. abrirá el archivo de origen "ISTOOLTIP.cbl".

La característica que crea un "Hipervínculo" en una variable, párrafo o nombre de un control de pantalla en el editor se ha simplificado. Al hacer clic en el nombre cuando el mouse tiene el puntero con forma de "mano", el IDE colocará el editor en la declaración, ya sea que se encuentre en el mismo archivo fuente o en un copybook diferente.

Figura 1. Ventana propiedades del IDE e hipervínculo

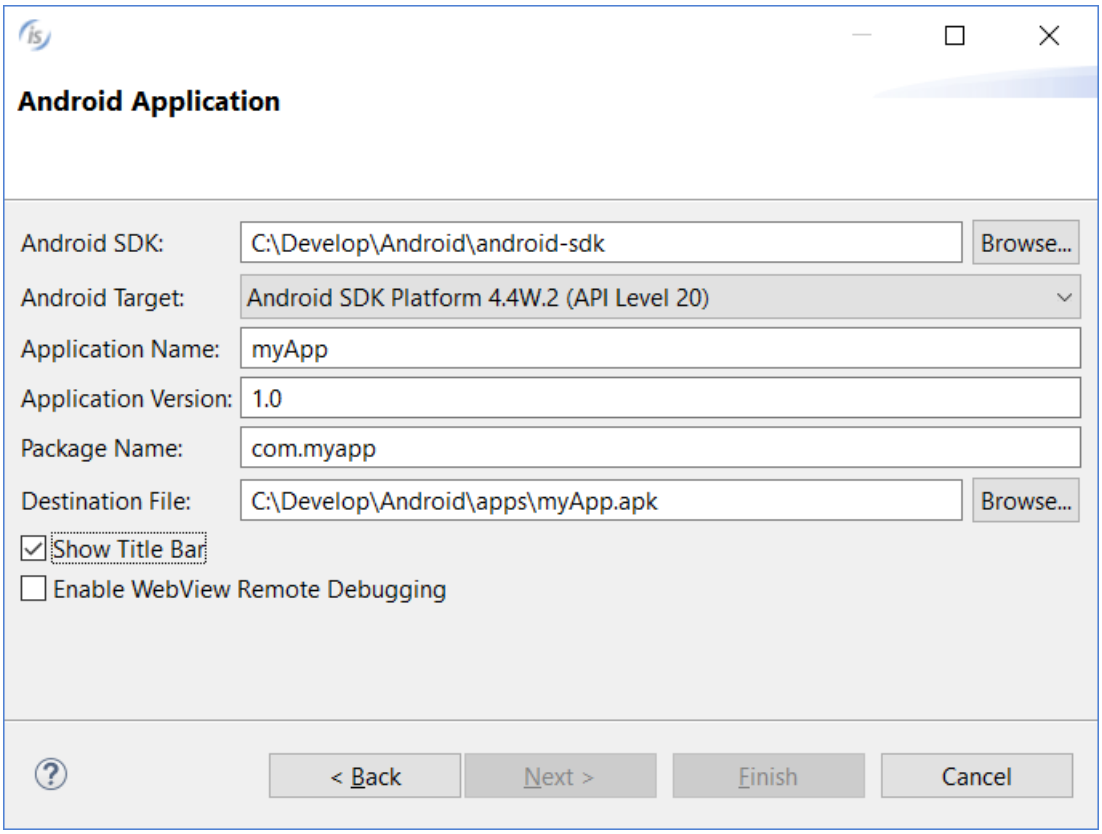


Ahora se puede compilar un programa incluso cuando el archivo actual del editor es un copybook, siempre que el copybook se haya abierto desde su archivo fuente principal, mientras que las versiones anteriores requerían que un archivo fuente del programa esté activo.

Exportar a Android

IsCOBOL IDE ahora le permite personalizar la barra de título de la aplicación al exportar el proyecto como una aplicación Android. Si marca la opción "Mostrar barra de título" como se muestra en la Figura 2, Barra de título de Android, se mostrará una Barra de Aplicaciones en la parte superior de la interfaz de usuario HTML de la aplicación exportada, mostrando el nombre de la aplicación.

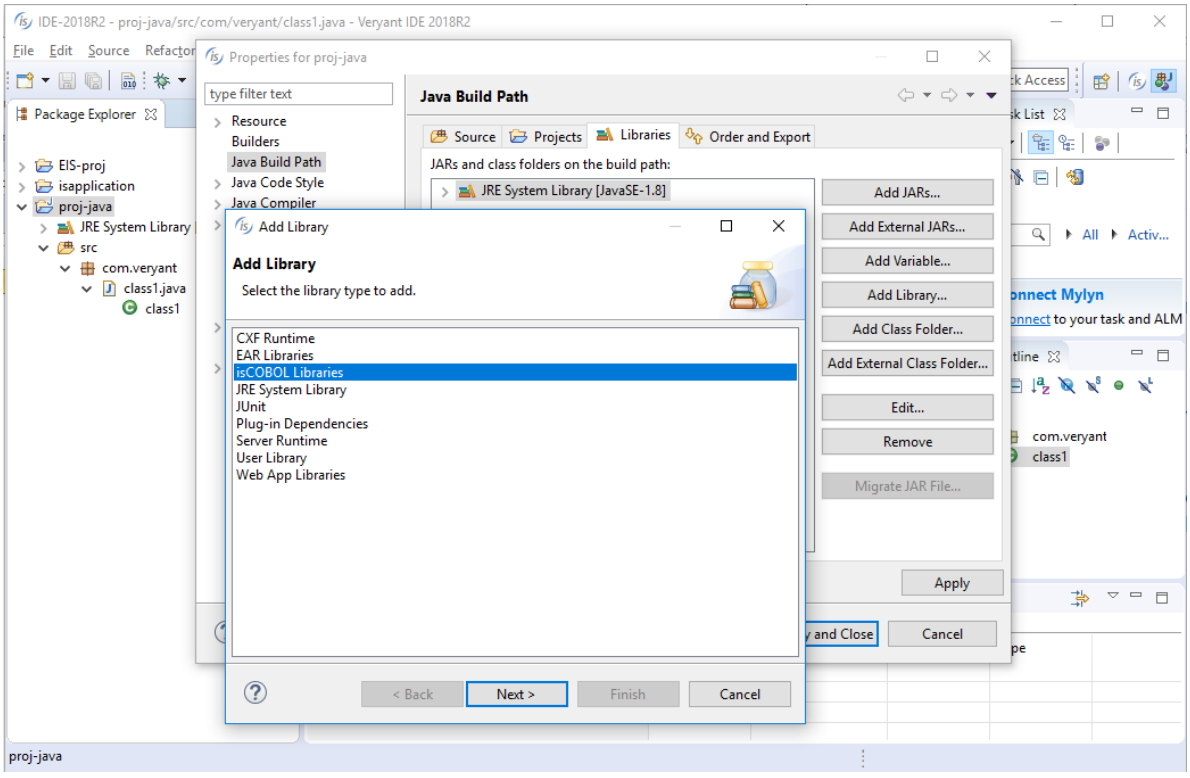
Figura 2. Barra de título de Android



Integración a Java más simple

Se agregó el elemento "isCOBOL libraries" al botón "Add Library" de la sección "Java Build Path" en la página de Java Properties, lo que permite la integración sencilla de bibliotecas de isCOBOL a una aplicación Java, como se muestra en la Figura 3, Agregar Bibliotecas de isCOBOL . Esto reemplaza la necesidad de agregar manualmente las librerías isCOBOL al ClassPath del proyecto Java, requerido en versiones anteriores.

Figure 3. Agregar Bibliotecas de isCOBOL



Nuevas características de interfaz de usuario

El control GRID se ha mejorado con funciones automáticas de búsqueda y filtro. Las secciones de pantalla ahora son dinámicas, lo que permite agregar y eliminar controles en tiempo de ejecución.

Otras mejoras menores están diseñadas para mejorar el manejo de la interfaz de usuario**.**

Mejoras al control GRID

La búsqueda en el Grid está habilitada de forma predeterminada y el usuario puede activarla en el tiempo de ejecución presionando Ctrl-F. Un estilo NO-SEARCH está disponible para desactivar la función cuando sea necesario.

Cuando la función de búsqueda está activada, se muestra un panel en la parte superior del Grid para permitir que el usuario busque datos dentro del Grid. En el panel, el usuario puede ingresar el texto de búsqueda o seleccionar uno previamente utilizado. Cuando se presiona el botón Buscar, las ocurrencias del texto de búsqueda se resaltan dentro del Grid. El botón Borrar permite al usuario restablecer el texto de búsqueda, y el botón 'X' permite al usuario cerrar el panel de búsqueda.

La propiedad HEADING-MENU-POPUP del Grid ahora también admite los valores GRHM-FINDON-RIGHT-CLICK y GRHM-FIND-ON-BUTTON para agregar el nuevo elemento Find en los menús de encabezado automático.

La Figura 4, El Grid antes de activar la función de búsqueda, muestra la cuadrícula mientras el usuario está navegando, mientras que la Figura 5, Búsqueda en cuadrícula muestra el panel de búsqueda que se muestra cuando el usuario presiona Ctrl-F mientras la cuadrícula tiene foco. Esto le permite al usuario buscar todas las ocurrencias del texto dado**.**

Figura 4. El Grid antes de activar la función de búsqueda

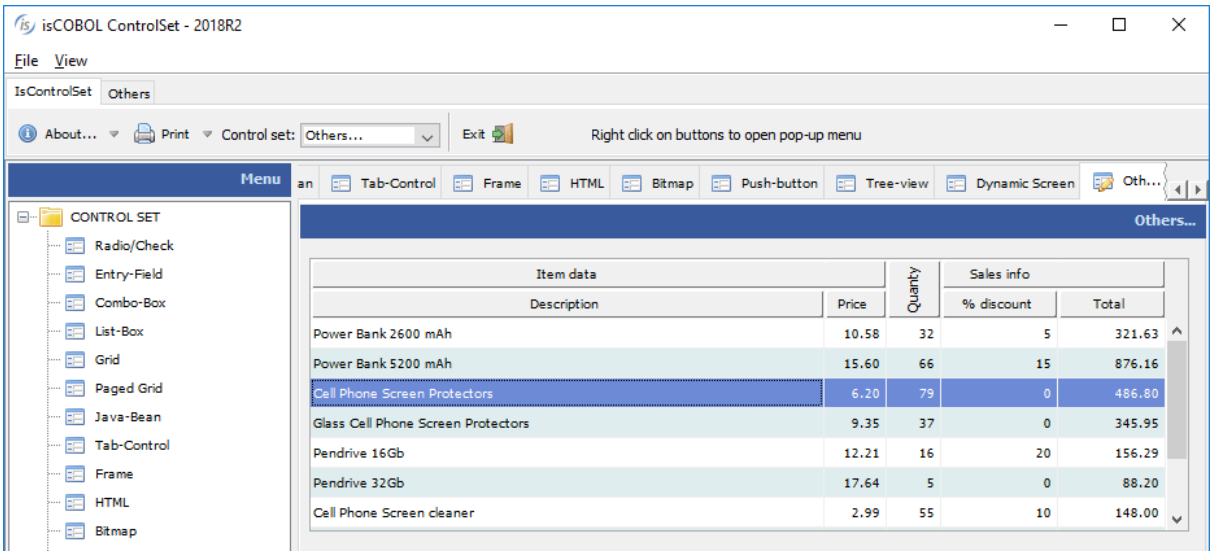
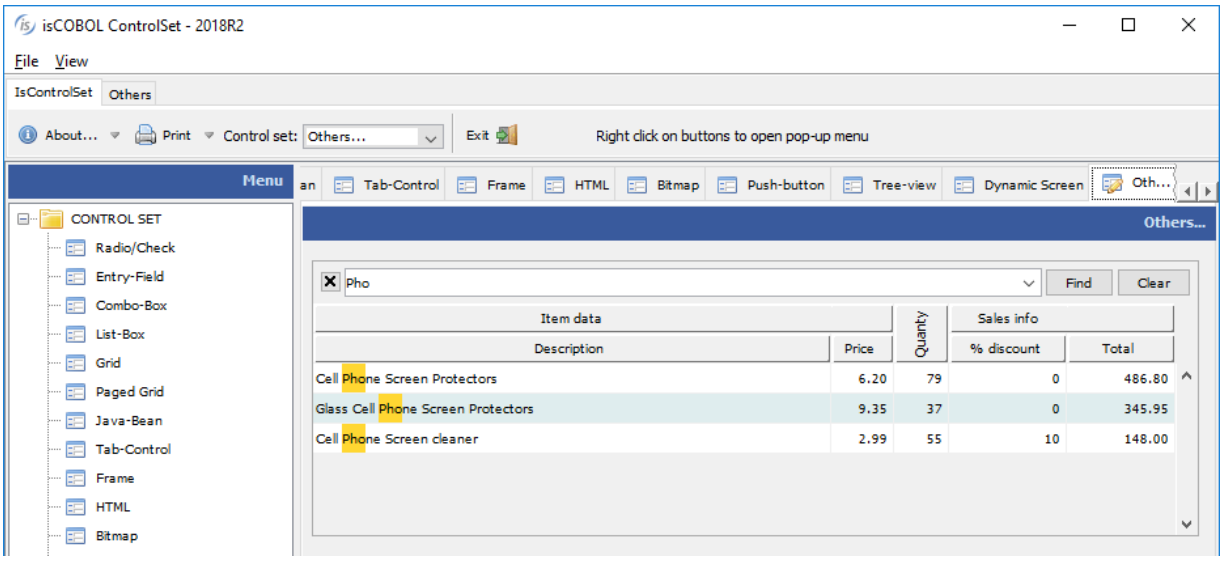


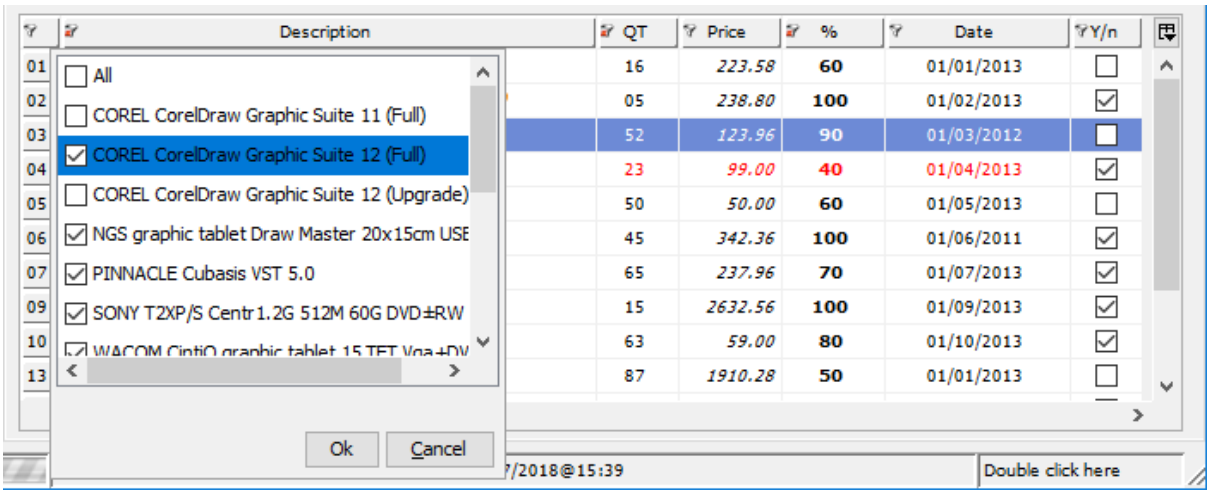
Figura 5. Grid con la función de búsqueda activada



Un nuevo estilo denominado FILTERABLE-COLUMNS ahora es soportado en el Grid, para permitir el filtrado de datos en función del contenido de la columna, como se muestra en la Figura 6, Filtro del Grid. Cuando se activa en una columna, una ventana emergente mostrará cada ocurrencia distinta de los valores de columna, lo que permite al usuario elegir uno o más valores para filtrar filas. Los filtros pueden agregarse a múltiples columnas y eliminarse según sea necesario. Se muestra un icono en el encabezado de la columna para mostrar un filtro activo.

```
05  h-grid, grid
    filterable-columns
    ...
```

Figura 6. Filtro del Grid



Pantallas dinámicas

Las Screen Sections ahora son dinámicas, permitiendo que los controles o la pantalla completa se agreguen o eliminen en tiempo de ejecución usando la sintaxis DISPLAY UPON SCREEN y DESTROY.

El objetivo UPON puede ser una screen section, usando la siguiente sintaxis

```
display screen1 upon screen2
```

o puede ser cualquiera de sus niveles de grupo, lo que permite la creación de controles secundarios, usando la sintaxis

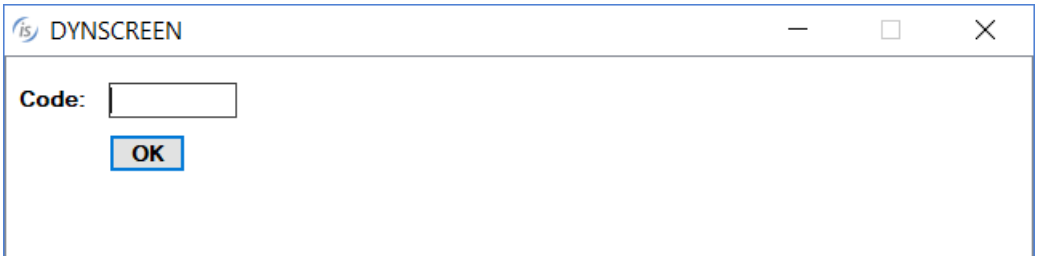
```
display screen1 upon screen2-group-2
```

El siguiente código muestra una pantalla con solo un botón, y lo rellena en tiempo de ejecución utilizando las instrucciones DISPLAY, antes de aceptar la entrada del usuario.

```
program-id. dynscreen.
working-storage section.
77 key-status special-names crt status pic 9(4).
   88 exit-pushed value 27.
77 w-code pic x any length.
screen section.
01 screen1.
   05 push-button ok-button
      line 4 col 9, size 6 cells.
procedure division.
main.
*display a window
     display standard graphical window.
*display the initial screen
     display screen1.
*add a label
     display label upon screen1
             line 2, col 2, size 6 cells
             title "Code:".
*add an entry-field
      display entry-field upon screen1
              line 2, col 9, size 10 cells
              value w-code.
*accept the screen
     perform until exit-pushed
        accept screen1
            on exception
               continue
        end-accept
     end-perform.
     destroy screen1.
```

El resultado de ejecutar el código anterior se muestra en la Figura 7, Pantalla dinámica

Figura 7. Pantalla dinámica



Otras mejoras

El control message-box ahora reproduce el correspondiente sonido de notificación basado en el tipo de mensaje, más fielmente al estándar Microsoft © Windows ©..

Mejoras al Framework (Runtime)

La nueva versión presenta una nueva ventana de vista previa de impresión no modal. Antes de la versión 2018R2, las ventanas de vista previa de impresión eran modales.

La nueva característica de auditoría permite que la auditoría del procesamiento de E/S tenga lugar en la aplicación, sin requerir cambios de código..

Vista Previa (Print Preview)

La función Print Preview de isCOBOL ahora se muestra en una ventana no modal que permite múltiples ventanas de vista previa simultáneas. El usuario puede cambiar entre ventanas de vista previa abiertas haciendo clic en ellas.

Función de Auditoría

La función de auditoría construido en la tecnología de activación (Trigger) de archivos de E/S, lo que permite a los desarrolladores de COBOL agregar el registro en todas las operaciones de E/S sin modificaciones del código fuente.

La herramienta de interfaz gráfica de usuario de isCOBOL 2018R2 proporciona una forma sencilla de configurar las características de auditoría, especificando usuarios, operaciones y archivos en los que se activa la auditoría. La auditoría se puede configurar para desencadenar en todos los usuarios que accedan a uno o más archivos, o en operaciones específicas como operaciones de eliminación ejecutadas por usuarios en un archivo específico.
La función de auditoría está escrita en COBOL y se proporciona como código fuente, para personalizarse fácilmente a cualquier entorno y caso de uso.

Mejoras de compatibilidad

Sintaxis adicional de ESQL soportada por Pro*COBOL ahora es compatible con isCOBOL, lo que simplifica la migración de Pro*COBOL a JDBC.

Nueva sintaxis, rutinas de biblioteca y propiedades de configuración son compatibles para mejorar la compatibilidad con otros dialectos COBOL.

Nueva sintaxis ESQL: .

- Soporte de la instrucción ESQL LOCK TABLE para definir un lock en una tabla o porción de una tabla.

  Ejemplo de código :

```
exec sql
     lock table customers
                in row exclusive mode
                nowait
end-exec.
```

- La cláusula FOR y el uso de arreglos de matriz (OCCURS) sin índice ahora son soportados en varias sentencias ESQL. El runtime repite una instrucción varias veces cuando se utiliza un elemento de datos OCCURS entre las variables del sistema principal. Por ejemplo, el siguiente código:

```
working-storage section.
77 ins-values pic x(10) occurs 4.
procedure division.
    move "aaa" to ins-values(1).
    move "bbb" to ins-values(2).
    move "ccc" to ins-values(3).
    move "ddd" to ins-values(4).
    exec sql
         for 3
         insert into tbl1 (column1) values (:ins-values)
    end-exec.
```

  hará que el runtime repita la instrucción INSERT 3 veces, utilizando ins-values (1), ins-values (2) e ins-values (3), respectivamente. Sin la cláusula FOR, la instrucción se ejecutará 4 veces, una para cada elemento en ins-values.

- La cláusula RETURNING ahora es soportada, lo que permite la recuperación de valores actualizados después de que se haya ejecutado una instrucción, por ejemplo :

```
exec sql
      insert into emptbl
       (empno,
        ename,
        deptno
        )
      values
       ('12', 'John Doe', 'dep1')
        returning empno, ename, deptno into
        :new_emp_number, :new_emp_name, :new_dept
end-exec.
```

- La cláusula INTO de las instrucciones SELECT y FETCH ESQL ahora puede vincularse a un elemento de datos OCCURS, por ejemplo :

```
working-storage section.
77 col-val pic x(10) occurs 3.
procedure division.
    exec sql
        select column1 into :col-val from tbl1
    end-exec.
```

  La instrucción anterior leerá 5 registros de la tabla tbl1 y almacenará los valores del campo column1 en col-val (1), col-val (2), col-val (3).

Compatibilidad mejorada con otros COBOLes

isCOBOL ofrece ahora una mayor compatibilidad con otros dialectos COBOL.

- Las propiedades en los programas CLASS-ID ahora son soportadas. El programa CLASS-ID establece la propiedad como lo haría con una variable estándar, por ejemplo:

```
identification division.
Class-id. myClass.
identification division.
factory.
working-storage section.
01 myProp pic 9(10) comp property.
procedure division.
identification division.
method-id. myMethod
procedure division.
main.
    move 2 to myProp.
```

  Los programas heredados y los programas CLASS-ID que hacen referencia a la clase anterior también pueden hacer referencia a la propiedad en el párrafo REPOSITORY y usarla como una variable COBOL estándar, por ejemplo:

```
configuration section.
repository.
    class myClass as "myClass"
    property myProp.
procedure division.
main.
    display myProp of myClass.
```

- La sintaxis ACTIVE-CLASS ahora es soportada, lo que le permite identificar una instancia específica de la clase actual o una de sus subclases .

  Ejemplo:

```
method-id. getInstance as "getInstance".
working-storage section.
77 cls-instance object reference active-class.
procedure division returning cls-instance.
main.
    invoke self "new" giving cls-instance.
```

- Nuevas directivas de Micro Focus® ahora son soportadas :

```
$SET INDD"<filename>"
$SET OUTDD"<filename [recsize] [filetype]>"
```

  Para permitirle asignar la entrada del sistema (SYSIN) y la salida del sistema (SYSOUT) a archivos en disco. Cada programa puede usar diferentes archivos de disco. Esto se explica en detalle más adelante en este documento.

Nueva rutina de bibioteca

Se implementó una nueva rutina de biblioteca para proporcionar una mejor compatibilidad con RM/COBOL®:
C$WRU devuelve el nombre del programa de llamada.

El siguiente fragmento de código muestra el uso de la rutina de la biblioteca C$WRU

```
01 WHO-CALLED-ME.
     05 THE-CALLING-PROGRAM PIC X(30)  VALUE SPACES.
     05 THE-CALLING-LINE    PIC S9(6)  BINARY.
     05 THE-LINE-NUM        PIC S9(02) BINARY.
PROCEDURE DIVISION.
     CALL 'C$WRU' USING THE-CALLING-PROGRAM
                        THE-CALLING-LINE
                        THE-LINE-NUM.
```

Nueva propiedad de configuración

Una nueva propiedad de configuración, `iscobol.memory.alpha_edited=true,` se ha implementado para administrar la cláusula VALUE de elementos editados alfanuméricos en compatibilidad con Micro Focus®, AcuCOBOL-GT® y RM/COBOL®.

Mejoras del compilador

isCOBOL isCOBOL Evolve 2018 R2 implementa nuevas opciones del compilador y nueva sintaxis para mejor soporte a las aplicaciones COBOL que usan DBCS (Cadenas de caracteres de doble byte) como japonés, chino y coreano sin codificación Unicode.
El compilador ahora puede compilar automáticamente las clases utilizadas por el programa que se compila.

Nuevas opciones del compilador

-ccbas para contar bytes en lugar de caracteres para código fuente fijo (también conocido como ANSI)
Esta opción ayuda al compilar programas fuente que contienen texto DBCS. De forma predeterminada, el compilador isCOBOL cuenta los caracteres para determinar las posiciones de texto de las distintas áreas del formato fuente COBOL fijo (ANSI), mientras que los compiladores asiáticos cuentan los bytes. Los archivos de origen escritos para COBOL asiáticos pueden no compilarse limpiamente sin la opción -ccbas.

-cndbcs para utilizar el DBCS en lugar de Unicode en PIC N sin la cláusula USAGE NATIONAL.

Un elemento de datos como :

```
77 n-item pic n(10).
```

almacena datos en UTF-16 Big Endian de forma predeterminada. Al compilar el programa con -cndbcs, los datos se almacenarán utilizando la codificación actual. Si se necesitan elementos de datos Unicode y DBCS en la misma aplicación, se pueden declarar de la siguiente manera:

```
77 dbcs-item    pic n(10).
77 unicode-item pic n(10) USAGE NATIONAL.
```

Compilación automática de clases

Cuando compila programas COBOL (PROGRAM-ID) o clases COBOL (CLASS-ID) que hacen referencia a clases COBOL adicionales (CLASS-ID), el compilador ahora compila automáticamente el código fuente al que se hace referencia.

Por ejemplo, teniendo en cuenta los siguientes tres archivos de código fuente:

Prog.cbl:

```
identification division.
program-id. Prog.
environment division.
configuration section.
repository.
    class Class2 as "Class2".
procedure division.
main.
    Class2:>method1().
    goback.
```

Class1.cbl:

```
identification division.
class-id. Class1 as "Class1".
identification division.
factory.
procedure division.
identification division.
method-id. method1 as "method1".
procedure division.
main.
* do something here
end method.
end factory.
```

Class2.cbl:

```
identification division.
class-id. Class2 as "Class2" inherits Class1.
environment division.
configuration section.
repository.
    class Class1 as "Class1".
identification division.
factory.
procedure division.
*add some methods here
end factory.
```

Al compilar Prog.cbl, los archivos fuente Class1.cbl y Class2.cbl se compilan automáticamente, tanto al compilar desde el IDE como desde la línea de comando .

Mapeo de SYSIN y SYSOUT

Se han agregado dos nuevas propiedades al compilador para permitirle asignar la entrada estándar del sistema (SYSIN) y la salida estándar (SYSOUT) a archivos del disco.

```
iscobol.compiler.indd=<filename>
iscobol.compiler.outdd=<filename [recsize] [filetype]>
```

Por ejemplo, consideremos usar las siguientes propiedades de configuración del compilador :

```
iscobol.compiler.indd=input.txt
iscobol.compiler.outdd=output.txt
```

Si un programa se compila con esas propiedades, leerá una línea del archivo input.txt en lugar de aceptar la entrada del usuario cuando realice ACCEPT dest-item FROM SYSIN. Cuando el programa ejecuta un DISPLAY ... UPON SYSOUT, escribirá una línea en el archivo output.txt en lugar de imprimir en la consola.

Para permitir que los programas usen archivos de entrada y salida específicos del programa, estas dos propiedades también se pueden usar como directivas del compilador dentro del código fuente. Por ejemplo:

```
$SET INDD "input-prog1.txt"
$SET OUTDD "output-prog1.txt"
program-id. prog1.
```

El Debugger de isCOBOL

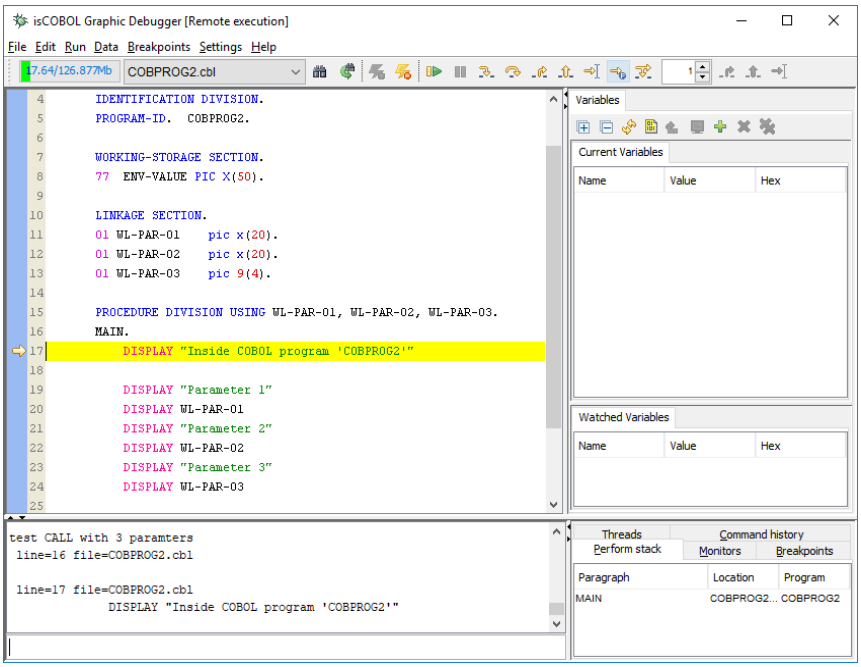
El depurador isCOBOL se ha mejorado mucho, con nuevas funciones destinadas a aumentar la productividad del desarrollador, como un nuevo comando de depurador, navegación de código más sencilla y más.

Nuevo commando del debugger

El depurador isCOBOL tienen ahora la nueva función "Run to next program", utilizando el comando PROG. Cuando se utiliza el comando PROG, el depurador continúa la ejecución del programa actual, deteniéndose al comienzo del siguiente programa COBOL. Esto es especialmente útil cuando el programa que llama no es un programa COBOL, como un programa Java o C, lo que permite que el depurador se detenga en el primer programa COBOL llamado.

Por ejemplo, al escribir PROG en la ventana de comandos, presionar el botón en la barra de herramientas que se muestra en la Figura 8 o elegir la opción del menú, el comando PROG del depurador, se ejecutará COBPROG2 y el Depurador se detendrá cuando se ejecute COBPROG3, que es el próximo programa llamado por la fuente principal de Java.

Figur8 8. Comando PROG del Depurador (Debugger)

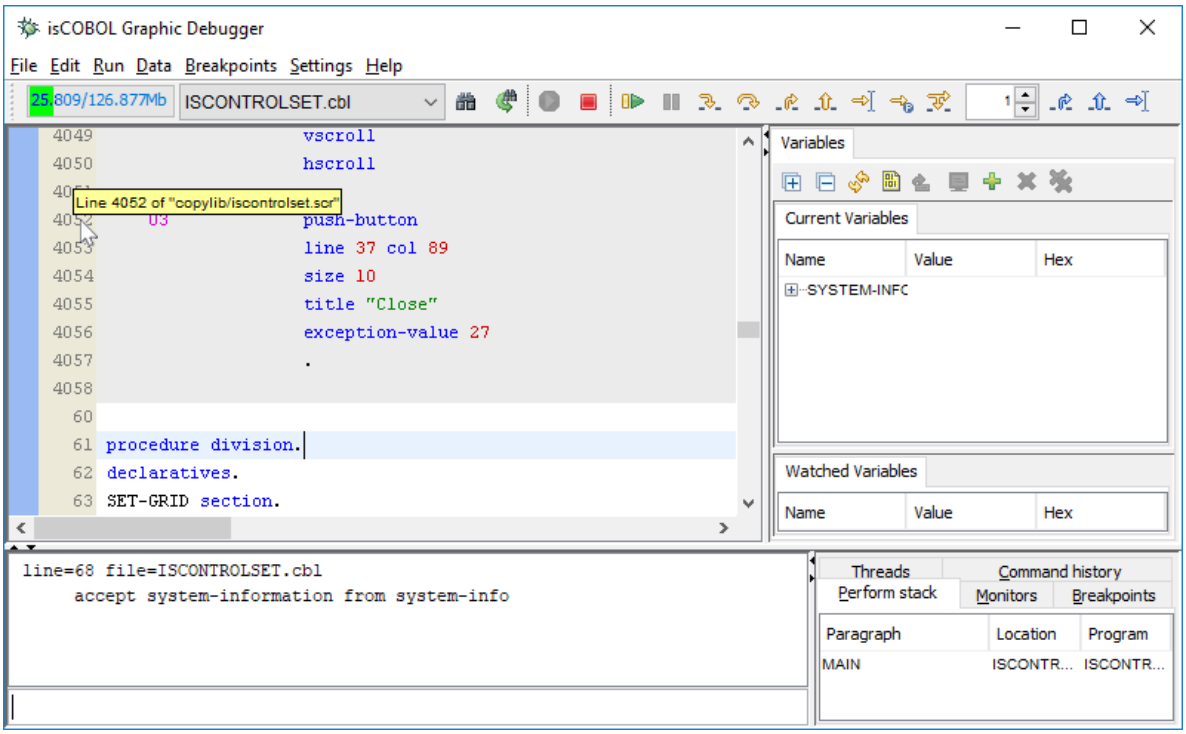


Información de la línea del programa

En programas grandes con muchos archivos de cuadernos, a veces es difícil determinar qué archivo contiene el código fuente que se está depurando. Con el nuevo depurador, aparecerá una sugerencia al pasar el puntero del mouse sobre la columna del número de línea, ubicado en el panel izquierdo de la ventana del depurador, mostrando la ruta del archivo y el número de línea de la línea resaltada.

La Figura 9, Sugerencia del depurador en la columna de línea, muestra la característica en acción.

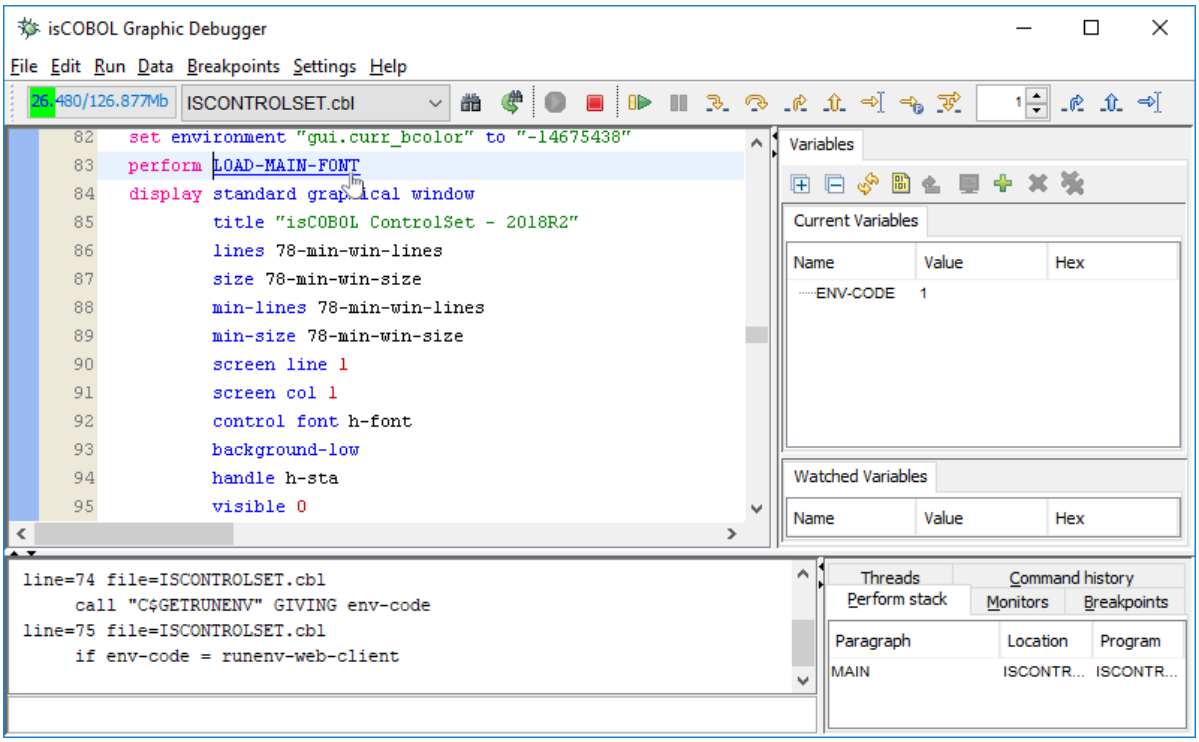
Figura 9. Sugerencia del depurador



Hipervínculo

Para simplificar el saltar a un párrafo dado o definición de variable, se ha introducido la característica de hipervínculo. Al colocar el puntero del mouse sobre un nombre de párrafo o un nombre de variable, el nombre se subraya y el cursor del mouse cambiará al punto de forma de la mano, como se muestra en la Figura 10, hipervínculo del depurador. Al hacer clic con el botón izquierdo sobre el nombre, el depurador mostrará la definición.

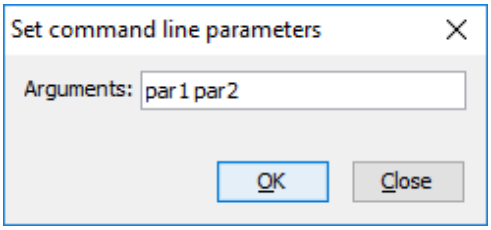
Figura 10. Hipervínculo del depurador



Parámetros en la línea de comando

Los valores de ' Set command line parameters' ahora se guardan en el archivo de sesión del depurador (.isd), lo que facilita la depuración de un programa con parámetros de CHAINING en diferentes sesiones de depuración, como se muestra en la Figura 11, Parámetros de la línea de comandos del Depurador.

Figure 11. Parámetros de la línea de comandos del Depurador



isCOBOL EIS

Las funciones isCOBOL EIS se han mejorado en varias áreas, como la clase HTTPHandler, las rutinas WD2 y el utilitario Stream2Wrk.

Mejoras de HTTPHandler

isCOBOL EIS ahora permite el acceso a los objetos clave de Servlet en ambos, en los los servlets de COBOL y las aplicaciones Web Direct 2.0.

- La clase HTTPHandler proporciona los siguientes nuevos métodos:

```
HTTPHandler:>getRequest()

HTTPHandler:>getResponse()

HTTPHandler:>getSession()
```

Estos métodos devuelven la instancia de HTTP Request, Response y Session respectivamente. Por ejemplo, estos objetos se pueden usar para recuperar la dirección IP del usuario final en su Servlet, usando el siguiente código:

```
repository.
    class HTTPHandler as "com.iscobol.rts.HTTPHandler"
    class HTTPRequest as "javax.servlet.ServletRequest"
    .
working-storage section.
77 servlet-request object reference HTTPRequest.
77 client-ip pic x any length.
linkage section.
77 http-handler object reference HTTPHandler.
procedure division using http-handler.
    set servlet-request to http-handler:>getRequest()
                        as HTTPRequest.
    set client-ip to servlet-request:>getRemoteAddr().
```

Mejoras a WD2

La rutina de Web Direct 2.0 WD2$SESSION se ha mejorado con nuevas propiedades que devuelven información sobre el contexto del servlet y la sesión HTTP del servlet que gestiona la aplicación COBOL. Estas son:

```
iscobol.wd2.servletcontext.name
iscobol.wd2.servletcontext.realpath
iscobol.wd2.servletcontext.path
iscobol.wd2.servletcontext.serverinfo
iscobol.wd2.servletcontext.majorversion
iscobol.wd2.servletcontext.minorversion
iscobol.wd2.httpsession.id
iscobol.wd2.httpsession.creationtime
```

Por ejemplo, para recuperar la ruta real de una aplicación desplegada en un contenedor de servlets, se puede usar el siguiente código :

```
working-storage section.
copy "iscobol.def".
77 real-path pic x any length.
procedure division.
    call "wd2$session" using wd2-get-session-value
                             "iscobol.wd2.servletcontext.realpath"
                             real-path.
```

Además, la representación de controles en WD2 se ha actualizado para mejorar la visualización de la Screen Section .

Mejoras a Stream2Wrk

El utilitario Stream2Wrk proporciona una nueva opción para especificar el nombre del elemento raíz al procesar flujos JSON que no tienen ningún elemento raíz.
Por ejemplo, el siguiente JSON:

```
{
"name":"John",
"age":30,
"cars":[ "Ford", "BMW", "Fiat" ]
}
```

generaría la siguiente estructura en working-storage :

```
01 json2wrk identified by ''.
    03 name identified by 'name'.
        05 name-data pic x any length.
    03 age identified by 'age'.
        05 age-data pic x any length.
    03 cars identified by 'cars' occurs dynamic capacity cars-count.
        05 cars-data pic x any length.
```

Para cambiar, por ejemplo, el nombre "json2wrk" generado automáticamente creado por Stream2Wrk a "allcars", se puede usar la siguiente línea de comando:

```
$ stream2wrk json yourfile.json -r allcars
```

El resultado sería :

```
01 allcars identified by ''.
   03 name identified by 'name'.
      05 name-data pic x any length.
   03 age identified by 'age'.
      05 age-data pic x any length.
   03 cars identified by 'cars' occurs dynamic capacity cars-count.
      05 cars-data pic x any length.
```