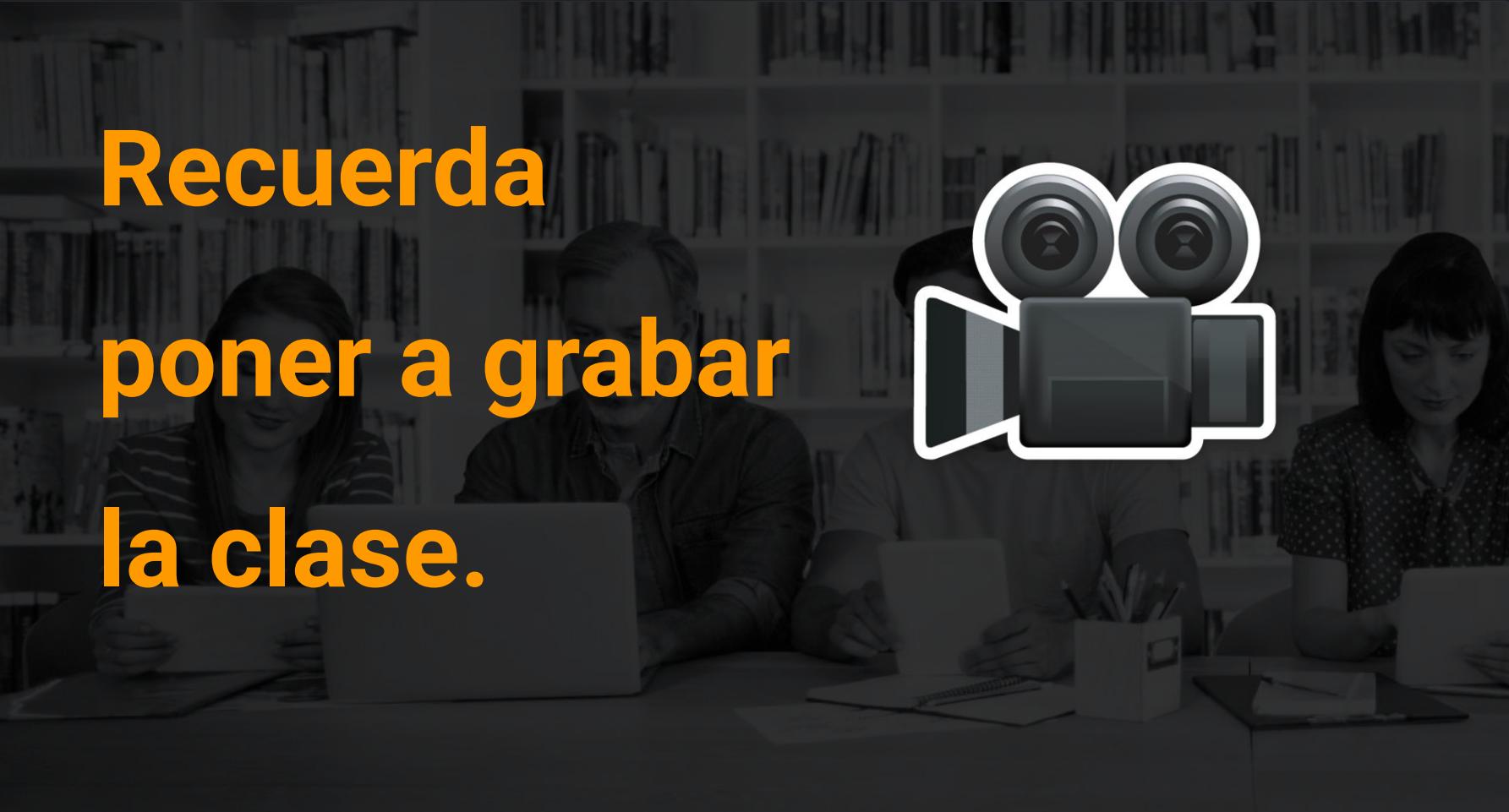


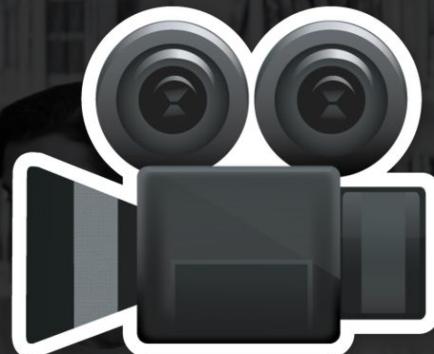
The background is a dark, slightly blurred photograph of an office environment. Several people are visible, each sitting at a desk with multiple computer monitors. The scene suggests a busy, technical workspace.

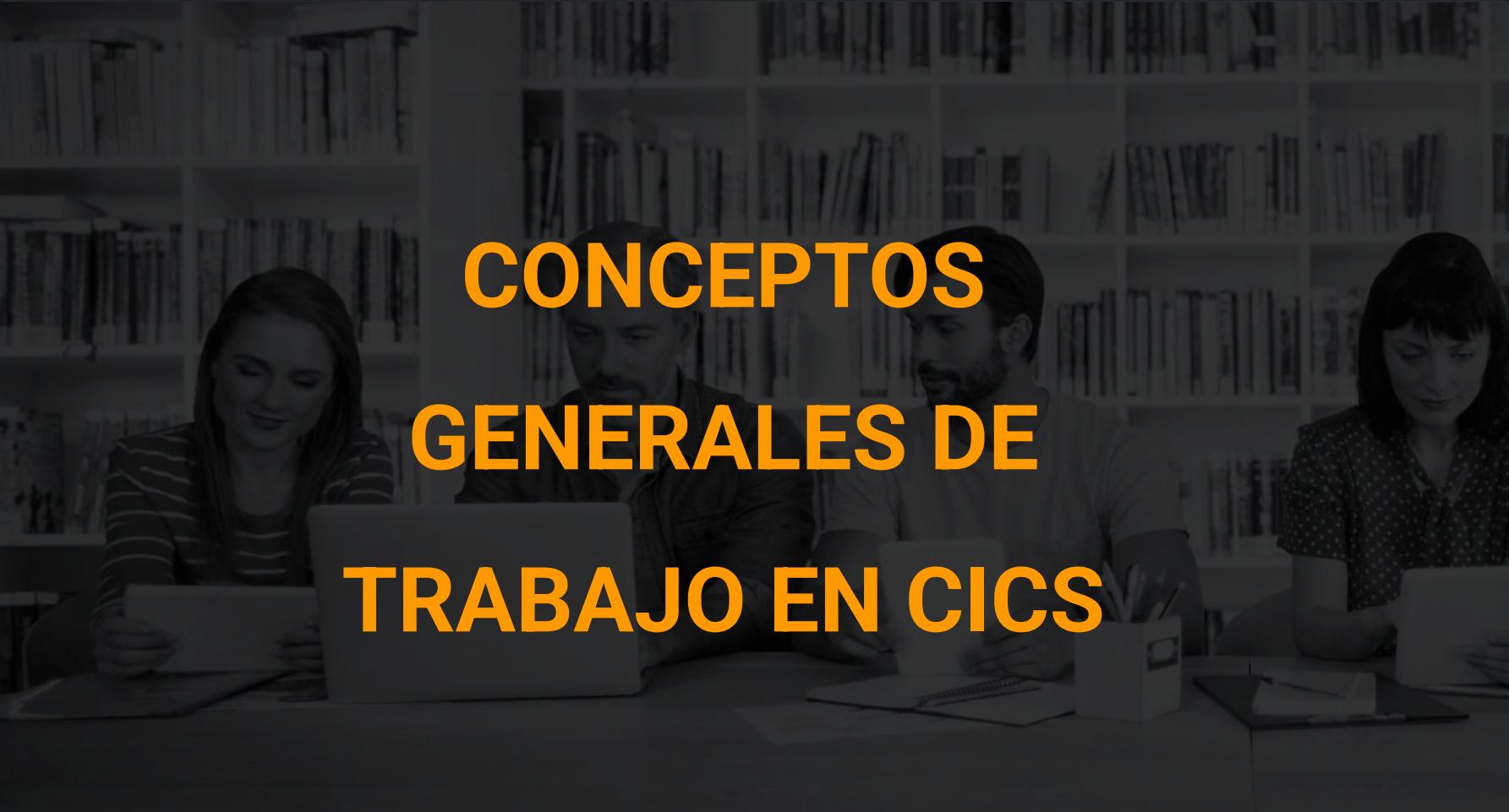
Curso

PROGRAMA COBOL CICS



Recuerda
poner a grabar
la clase.



A black and white photograph showing four people in a library or study room. On the left, a woman in a striped shirt looks down at her laptop. In the center, a man with a beard is focused on his screen. To his right, another man with a beard is looking towards the camera. On the far right, a woman with short hair is also looking at her laptop. Bookshelves filled with books are visible in the background.

CONCEPTOS GENERALES DE TRABAJO EN CICS



... Mapa Conceptual

01

Exec Interfase
Block

02

Testeo
Programas

03

Manejo de Mapas

04

Tratamiento
de archivos



... **Agenda**

- 1 EIB – Exec Interfase Block**
- 2 Testeo de programas**
- 3 Condición HANDLE / RESP**
- 4 Manejo de mapas**
- 5 Tratamiento de archivos**





01

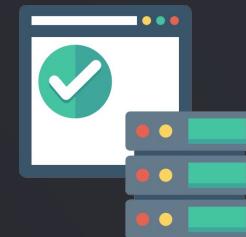
EIB – EXEC INTERFASE BLOCK

...

Al momento de la traducción de los comandos de CICS por el Translator, se incorpora en la LINKAGE SECTION una estructura de datos que el CICS utilizará para pasarnos información a nuestro programa.

Esta estructura es denominada como DFHEIBLK y se corresponde con la EIB (EXEC INTERFASE BLOCK) del CICS, donde este almacena toda la información necesaria sobre la tarea actualmente en ejecución, la terminal, la hora, el usuario y, sobre todo, el resultado de cada comando CICS que nuestra aplicación invocó.

Nuestro programa puede acceder a todos los campos de la EIB, pero se recomienda no cambiar el contenido de ninguno.





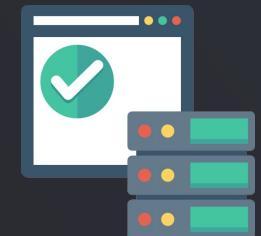
01

EIB – EXEC INTERFASE BLOCK

...

Los campos de la EIB más utilizados son:

- **EIBAID** Contiene el identificador de atención del último comando RECEIVE ejecutado (ENTER, PF1-PF24, CLEAR, ATTN).
- **EIBCALEN** Contiene la longitud del área de comunicación DFHCOMMAREA que ha sido colocada en la LINKAGE SECTION de nuestro programa.
- **EIBCPOSN** brinda la posición que ocupaba el cursor al momento del último RECEIVE ejecutado.
- **EIBDATE** Fecha en que fue arrancada la tarea.
- **EIBF** Código de función que indica cuál comando de CICS fue el más recientemente utilizado por nuestro programa.
- **EIBRCODE** Código de condición del último comando CICS ejecutado.
- **EIBRSRCE** Contiene el nombre del recurso accedido más recientemente por un comando CICS (como ser el nombre de una cola TS o TD, un archivo, etc.)
- **EIBTASKN** número que el CICS asignó a la tarea actualmente en ejecución.
- **EIBTIME** Hora de arranque de la tarea.
- **EIBTRMID** Código de terminal donde arrancó la tarea.
- **EIBTRNID** Código identificador de la transacción actual.





... Mapa Conceptual

01

Exec Interfase
Block

02

Testeo

03

MANEJO de
MAPAS

04

Tratamiento
de archivos





02

TESTEO DE RESULTADO DE EJECUCIÓN

...
Cada vez que ejecutamos un comando CICS, este nos avisa sobre si la operación del comando fue satisfactoria, o si el recurso sobre el que estamos actuando (un archivo, una terminal, etc.) está disponible, etc.

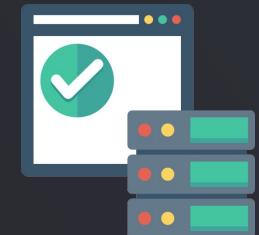
Por supuesto que nuestro programa deberá tener incorporado el código pertinente para llevar a cabo esos chequeos.

Entonces, cada vez que ejecutemos un comando CICS deberemos asegurarnos sobre el éxito de su cometido.

Si bien el propio CICS tiene mecanismos automáticos que ante errores provoca un vuelco de memoria, es necesario que en nuestro programa podamos prever acciones ante condiciones de excepción en el proceso que no son necesariamente errores.

Existen dos alternativas para la evaluación del resultado de nuestro comando CICS:

- 1) La utilización del **HANDLE CONDITION**
- 2) La utilización del **RESP**
- 3) La utilización combinada del **HANDLE CONDITION** y del **RESP**





... Mapa Conceptual

01

Exec Interfase
Block

02

Testeo

03

MANEJO de
MAPAS

04

Tratamiento
de archivos





03

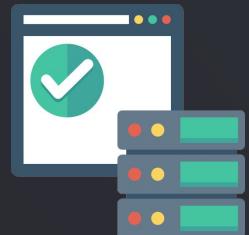
HANDLE CONDITION

...

```
EXEC CICS HANDLE CONDITION  
    condición [ (nombre-párrafo) ]  
    .....  
    condición [ (nombre-párrafo) ]  
END-EXEC.
```

Condition palabra clave que identifica la condición de excepción

Nombre-parrafo Nombre de párrafo o de sección de cobol donde se derivará el control del programa por cumplirse la condición de excepción asociada en la condición. Esta bifurcación es equivalente a un **GO TO** y no a un **PERFORM**.



HANDLE CONDITION



OBSERVACIONES sobre el HANDLE CONDITION

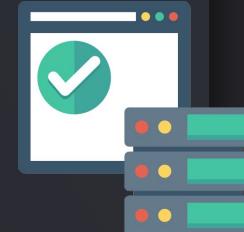
Se pueden especificar hasta 16 condiciones en un solo comando.

Si especificamos la condición sin nombre de párrafo asociado, se desactiva cualquier HANDLE CONDITION previa que atendía esa condición. En este caso el CICS toma el control.

Si se incorpora una condición que es común a varios comandos, debe entenderse que es extensiva la bifurcación por excepción de la condición sin importar el comando que la pueda activar.

En caso de LINK, no se hacen extensivos los HANDLE CONDITION al subprograma; este deberá establecer sus propios HANDLE CONDITION. Una vez retornado el control a la aplicación principal, se reanudan los HANDLE CONDITION que quedaron interrumpidos.

Si se especifica más de una vez el comando, las condiciones y sus bifurcaciones se van acumulando. En caso efectuar un PERFORM, se puede determinar una HANDLE CONDITION diferente dentro de esa rutina con el comando EXEC CICS PUSH HANDLE que suspende todas las HANDLE CONDITION, HANDLE AID y HANDLE ABEND actuales. Se setean entonces los HANDLE CONDITION propios para esa rutina y a su finalización se ejecuta el comando EXEC CICS POP HANDLE que restaura todos los HANDLE al estado anterior al PUSH.





03

HANDLE CONDITION

...

OBSERVACIONES sobre el HANDLE CONDITION

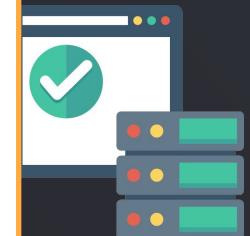
Para especificar una rutina de manejo de errores no contemplados explícitamente por los HANDLE CONDITION que se encuentran en nuestro programa, es conveniente utilizar al inicio del mismo la siguiente instrucción:

```
EXEC CICS HANDLE CONDITION  
      ERROR (error-cics)  
END-EXEC.
```

Entonces, si ocurre un error inesperado en la rutina error-cics, nuestro programa entraría en LOOP, salvo que al inicio de nuestra rutina error-cics incluyamos la siguiente instrucción.
Error-cics.

```
EXEC CICS HANDLE CONDITION  
      ERROR  
END-EXEC.  
.....
```

Fin-error-cics.





03

RESP

...

A todos los comandos de CICS les podemos agregar la opción **RESP** (PIC S9(8) COMP). Esta opción cumple la misma finalidad que el NOHANDLE, pero acto seguido podemos chequear su resultado con la función **DFHRESP**.

```
EXEC CICS  
  READQ TS QUEUE (COM-KEY)  
    INTO (COM-COMMAREA)  
    LENGTH (COM-LEN)  
    ITEM (WS-ITEM)  
    RESP (WS-RESP)  
END-EXEC.
```

```
EVALUATE WS-RESP  
  WHEN DFHRESP(NORMAL)  
    CONTINUE  
  WHEN DFHRESP(QIDERR)  
    CONTINUE  
  WHEN OTHER  
    PERFORM CICS-ERROR-AUT  
    THRU FIN-CICS-ERROR-AUT  
END-EVALUATE.
```





03

HANDLE AID

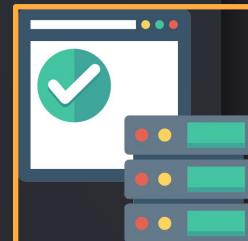
...

```
EXEC CICS HANDLE AID  
  AID-NAME [ (nombre-parrafo) ]  
  .....  
  ANYKEY [ (nombre-parrafo) ]  
END-EXEC.
```

Este comando funciona del mismo modo que el HANDLE CONDITION pero atendiendo a teclas pulsadas por el usuario operador. Las condiciones de este comando se activan siempre y cuando se haya ejecutado un comando RECEIVE o RECEIVE MAP.

AID-NAME Identifica palabras claves como ENTER, PF1, PF24, CLEAR, etc.

ANYKEY Se aplica a cualquier tecla pulsada, siempre que no se haya indicado en forma específica.
Tampoco aplica a la tecla ENTER.





03

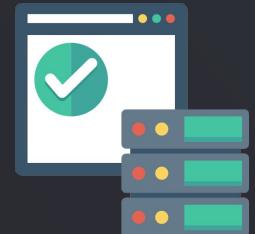
HANDLE AID vs EIBAID

```
PERFORM RECEIVE-MAPA  
THRU FIN-RECEIVE-MAPA.  
  
EXEC CICS HANDLE AID  
    ENTER (PFKEY-ENTER)  
    PF2 (PFKEY-PF2)  
    PF6 (PFKEY-PF6)  
    ANYKEY (PFKEY-INVALIDA)  
  
END-EXEC.
```

EMITE UN
GETMAIN
IMPLÍCITO

```
PERFORM RECEIVE-MAPA  
THRU FIN-RECEIVE-MAPA.  
  
EVALUATE EIBAID  
    WHEN DFHENTER  
        PERFORM PFKEY-ENTER  
        THRU FIN-PFKEY-ENTER  
  
    WHEN DFHPF2  
        PERFORM PFKEY-PF2  
        THRU FIN-PFKEY-PF2  
  
    WHEN DFHPF6  
        PERFORM PFKEY-PF6  
        THRU FIN-PFKEY-PF6  
  
    WHEN OTHER  
        PERFORM PFKEY-INVALIDA  
        THRU FIN-PFKEY-INVALIDA  
  
END-EVALUATE.
```

CON EIBAID SE
LOGRA MEJOR
PERFORMANCE





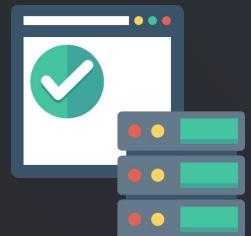
03

ACCESO A DATOS

...

COMANDOS PARA ACCESO A DATOS

- Obtención de fecha
- Comandos para manejo de Mapas
- Acceso a Archivos VSAM
- Acceso a TS Temporary Storage
- Acceso a TD Transient Data





03

ASK TIME

...

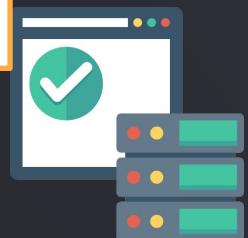
El comando **ASKTIME** provee fecha y hora y los actualiza en los campos **EIBDATE** y **EIBTIME** de la **EIB**.

EXEC CICS ASKTIME

[ABSTIME (data-area)]

END-EXEC.

Data-area Debe definirse como una doble palabra binaria en la cual se almacena el tiempo, en milisegundos, transcurridos desde las 00.00 hs del 1ro de Enero de 1900.





03

FORMATTIME

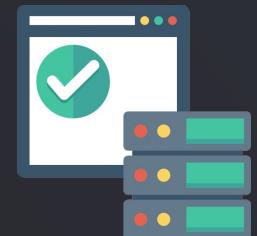
El comando **FORMATTIME** permite transformar a un formato dado la fecha obtenida por el comando **ASKTIME**.

EXEC CICS FORMATTIME

```
    ABSTIME (data-value)
    [ YYDDD(data-area) ] [ YYMMDD(data-area) ] [ YYDDMM(data-area) ]
    [ DDMMYY(data-area) ] [ MMDDYY(data-area) ] [ DATE(data-area) ]
    [ DATEFORM(data-area) ] [ DATESEP(data-value) ] [ DAYCOUNT(data-area) ]
    [ DAYOFWEEK(data-area) ] [ DAYOFMONTH(data-area) ]
    [ MONTHOFYEAR(data-area) ] [ YEAR(data-area) ]
    [ TIME(data-area) [ TIMESEP(data-value) ] ]
```

END-EXEC.

- **ABSTIME data-value** especifica el nombre de la data-area definida en el comando ASKTIME. Este valor de tiempo absoluto podrá ser formateado de acuerdo a las opciones que se indiquen.
- **YYDDD Data-area** especifica un campo de 5 caracteres donde se retorna la fecha en el formato indicado.
- **YYMMDD YYDDMM DDMMYY MMDDYY Data-area** especifica un campo de 8 caracteres donde se retorna la fecha en el formato indicado.



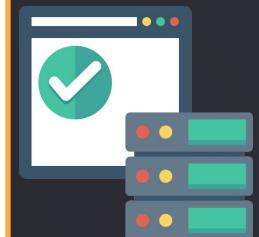


03

FORMATTIME

...

- **DATE data-area** especifica un campo de 8 caracteres en donde se retorna la fecha en el formato default seteado en la Instalación.
- **DATESEP Data-value** especifica el carácter que será insertado como separador entre los componentes de una fecha. Si se omite data-value, asume '/' como separador.
 - **Omitiendo DATESEP** : 311200
 - **DATESEP sin data-value** : 31/12/00
 - **DATESEP con data-value '-'** : 31-12-00
- **DATEFORM Data-area** especifica un campo de 6 caracteres donde se retorna la fecha en el formato default de la Instalación.
- **DAYCOUNT Data-area** especifica una palabra binaria donde retorna el número de días transcurridos desde el 1ro de Enero de 1900.
- **DAYOFWEEK Data-area** especifica una palabra binaria donde retorna el número de día relativo de la semana comenzando el domingo con valor cero, lunes valor 1 etc.
- **DAYOFMONTH Data-area** especifica una palabra binaria donde retorna el número de día relativo del mes.
- **MONTHOFYEAR Data-area** especifica una palabra binaria donde retorna el número de mes relativo al año.



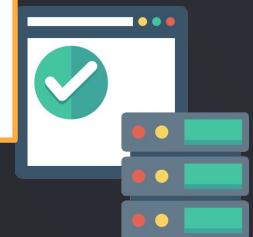


03

FORMATTIME

...

- **YEAR data-area** especifica una palabra binaria donde retorna el número completo del año (por ejemplo 2010).
- **TIME Data-area** especifica un campo de 8 caracteres donde retorna la hora en formato hhmmss.
- **TIMESEP Data-value** especifica un carácter que será insertado como separador entre hhmmss. Si se omite data-value, se asume ‘:’ como separador. Si se omite por completo esta opción, no inserta ningún carácter como separador.
 - **omitiendo TIMESEP** : 223059
 - **TIMESEP sin data-value** : 22:30:59
 - **TIMESEP con data-value ‘/’** : 22/20/59





03

ASKTIME y FORMATTIME

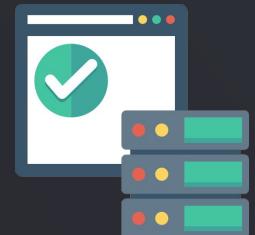
...

```
EXEC CICS  
  ASKTIME ABSTIME (WS-ABSTIME)  
END-EXEC.  
  
EXEC CICS FORMATTIME  
  ABSTIME (WS-ABSTIME)  
  YYMMDD (WS-AAMMDD)  
  TIME (WS-TIME) TIMESEP (':')  
  YEAR (WS-YEAR4)  
END-EXEC.  
  
MOVE WS-YEAR4 TO WS-FECHA-MAPA-AAAA.  
MOVE WS-AAMMDD-MM TO WS-FECHA-MAPA-MM.  
MOVE WS-AAMMDD-DD TO WS-FECHA-MAPA-DD.  
MOVE '/' TO WS-FECHA-MAPA-B1  
      WS-FECHA-MAPA-B2.
```

PROCEDURE

```
01 WS-TIME PIC X(08)VALUE SPACES.  
01 WS-HORA-SQL PIC X(08)VALUE SPACES.  
01 WS-HORA-MAPA PIC X(08)VALUE SPACES.  
01 WS-ABSTIME PIC S9(16)COMP VALUE +0.  
01 WS-YEAR4 PIC S9(09)COMP VALUE +0.  
01 WS-RESP PIC S9(08)COMP VALUE +0.  
01 WS-RESP2 PIC S9(08)COMP VALUE +0.  
01 WS-FECHA-MAPA.  
  10 WS-FECHA-MAPA-DD PIC 9(02).  
  10 WS-FECHA-MAPA-B1 PIC X(01).  
  10 WS-FECHA-MAPA-MM PIC 9(02).  
  10 WS-FECHA-MAPA-B2 PIC X(01).  
  10 WS-FECHA-MAPA-AAAA PIC 9(04).
```

WORKING





03

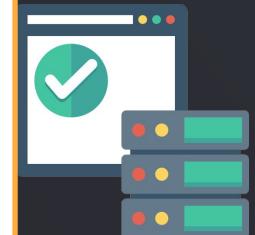
MANEJO DE MAPAS – SEND MAP

...

El comando **SEND MAP** permite enviar un mapa con datos a la terminal.

```
EXEC CICS SEND MAP ('name')
  [ MAPSET ('name') ]
  [ { DATAONLY : MAPONLY } ]
  [ FROM (data-area) ]
  [ LENGTH (data-value) ]
  [ CURSOR [ (data-value) ]
  [ { ERASE : ERASEUP } ]
  [ FREEKB ] [ FRSET ] [ ALARM ] [ PRINT ] [ FORMFEED ]
END-EXEC.
```

- **MAP** especifica el nombre del **MAPA** a ser enviado a la terminal (1 a 7 caracteres). Deberá coincidir con el nombre definido en la macro **BMS DFHMDI**.
- **MAPSET** especifica el nombre del **MAPSET**, conjunto de mapas al que pertenece el mapa a ser enviado a la terminal (1 a 7 caracteres). Deberá coincidir con el nombre definido en la macro **BMS DFHMSD**. Si es omitido, se asume el nombre especificado en **MAP**.
- **DATAONLY** especifica que solo los datos del programa de aplicación deben enviarse a la terminal. Todo dato o atributo del mapa físico es ignorado. No debe ser especificada la opción **ERASE**. Debe haber sido enviado con anterioridad un mapa válido a la terminal.



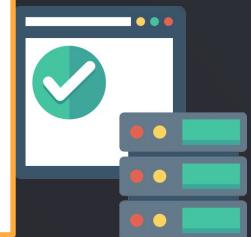


03

MANEJO DE MAPAS – SEND MAP

...

- **MAPONLY** especifica que solo el mapa físico será enviado a la terminal. La opción **FROM** no debe ser especificada.
Omitiendo **DATAONLY** y **MAPONLY**, los datos serán enviados a la terminal utilizando datos del mapa físico y del mapa simbólico.
- **FROM** especifica el área de datos del mapa simbólico a ser formateado dentro del mapa físico.
- **LENGTH** Indica la longitud de los datos a ser formateados; debe definirse como un campo binario de media palabra. No se requiere este parámetro si se especificó la opción **MAPONLY**.
- **CURSOR** Especifica la posición en la pantalla donde deberá ser situado el cursor. Se utiliza conjuntamente con el seteo del atributo de longitud del campo indicado en -1 (campo terminado en L). No debe ser especificado si se especificó la opción **MAPONLY**.
- **ERASE** Borra la pantalla antes de ser enviado el **MAPA**.
- **ERASEUP** Borra todos los campos no protegidos de la pantalla, permitiendo que permanezcan los campos definidos como **ASKIP** y **PROT**.
- **FREEKB** Indica que debe ser desbloqueado el teclado después de ser enviado el **MAPA**.
- **FRSET** Especifica que los **MDT** de todos los campos deben ser seteados en **OFF**.
- **ALARM** Especifica que la alarma de la terminal debe ser activada al mostrar el mapa.





03

MANEJO DE MAPAS – SEND MAP

...

- **PRINT** Inicia la operación de impresión cuando se envía el mapa. Si se omite este parámetro, los datos se almacenan en el buffer de la impresora pero no son impresos.
- **FORMFEED** especifica que una nueva página es requerida para la impresión del mapa enviado.

IMPORTANTE:

Antes de ser enviado un mapa, el área del mapa simbólico deberá ser inicializado con **LOW-VALUES**.





03

MANEJO DE MAPAS – RECEIVE MAP

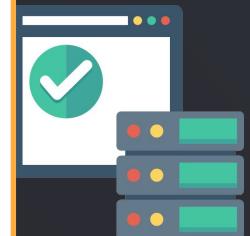
- **FROM** esta opción es usada cuando el mapa ya ha sido recibido en un área de almacenamiento, pero los datos no fueron formateados como mapa. Este comando realmente los formatea.
- **LENGTH** especifica La longitud del dato a ser formateado; debe definirse como un campo binario de media palabra **PIC S9(4) COMP**.
- **TERMINAL** especifica que los datos ingresados deben leerse desde la terminal que originó la transacción.
- **ASSIS** especifica que no debe efectuarse la conversión de letras minúsculas a mayúsculas.

1- El RECEIVE MAP recibe los datos no formateados que han sido leídos por el Programa de Control de Terminal (**TCP**) del **CICS**.

2- Formatea estos datos de acuerdo a la información obtenida del mapa físico correspondiente.

3- Inicializa el área receptora del programa con **LOW-VALUES**. Esta es la Work-area que está formateada usando el copy del mapa simbólico.

4- Coloca los datos de cada campo modificado en el correspondiente campo receptor dentro de esta área.





03

MANEJO DE MAPAS – RECEIVE MAP

...

El comando **RECEIVE MAP** permite recibir un mapa con datos desde la terminal.

EXEC CICS

RECEIVE MAP (WS-MAP)

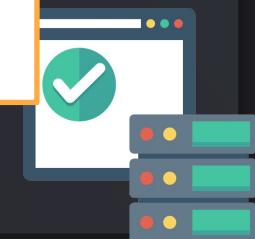
MAPSET (WS-MAPSET)

INTO (MAP0299I)

RESP(WS-RESP)

END-EXEC

.





... Mapa Conceptual

01

Exec Interfase
Block

02

Testeo

03

MANEJO de
MAPAS

04

Tratamiento
de archivos





03

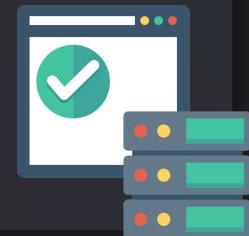
MANEJO DE ARCHIVOS - READ

...

El comando READ permite la lectura de un registro de un archivo VSAM.

```
EXEC CICS READ DATASET ('name')
  [ UPDATE ]
  RIDFLD (data-area)
  [ KEYLENGTH (data-value) : [ GENERIC ] ]
  [ { RBA : RRN } ]
  [ { SET (pointer-ref) : INTO (data-area) } ]
  [ LENGTH (data-area) ]
  [ { GTEQ : EQUAL } ]
END-EXEC.
```

- **DATASET** especifica el nombre del archivo a leer (que coincide con el declarado en la tabla **FCT**). Debe tener 8 caracteres como máximo y ajustado a la izquierda.
- **UPDATE** esta opción le indica al **FCP** que el registro accedido será posteriormente actualizado.
- **RIDFLD** identifica el nombre simbólico del área que contiene la clave del registro a leer. El formato y el contenido de **RIDFLD** varían en función del método de acceso que utilicemos.





03

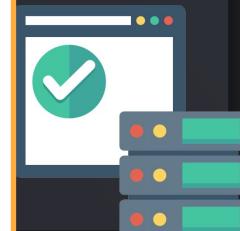
MANEJO DE ARCHIVOS - READ

- **RIDFLD** Para los archivos VSAM KSDS puede especificarse una clave COMPLETA o GENÉRICA. Cuando se usa una clave genérica, deben también especificarse las opciones KEYLENGTH y GENERIC. Estas opciones se describen más abajo. Luego de completado el READ, RIDFLD contendrá la clave completa del registro leído.

Para los archivos VSAM KSDS accedidos mediante RBA, RIDFLD contiene la dirección binaria del byte relativo de 4 bytes.

Para los archivos VSAM RRDS, RIDFLD contiene el número binario del registro relativo de 4 bytes.

- **GENERIC** identifica a la clave usada para buscar un registro en un archivo **VSAM KSDS** como una **CLAVE PARCIAL**, cuya longitud se especifica en **KEYLENGTH**.
- **KEYLENGTH** es un campo binario de dos bytes que identifica la longitud de la clave especificada en **RIDFLD**. Esta opción debe incluirse si se utiliza una clave genérica, ya que indica la longitud de la clave genérica.
- **SET** especifica el nombre de una celda **BLL**.
- **INTO** identifica el área de trabajo en la que ha de colocarse un registro después de haber sido leído. Si se estuviera trabajando con registros de longitud variable, la descripción del registro ha de ser lo suficientemente grande como para contener el registro de longitud máxima.





03

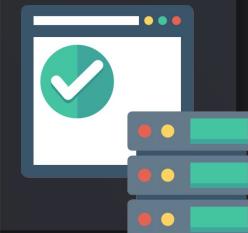
MANEJO DE ARCHIVOS - READ

- **RIDFLD** Para los archivos VSAM KSDS puede especificarse una clave COMPLETA o GENÉRICA. Cuando se usa una clave genérica, deben también especificarse las opciones KEYLENGTH y GENERIC. Estas opciones se describen más abajo. Luego de completado el READ, RIDFLD contendrá la clave completa del registro leído.

Para los archivos VSAM KSDS accedidos mediante R.B.A., RIDFLD contiene la dirección binaria del byte relativo de 4 bytes.

Para los archivos VSAM RRDS, RIDFLD contiene el número binario del registro relativo de 4 bytes.

- **GENERIC** identifica a la clave usada para buscar un registro en un archivo **VSAM KSDS** como una **CLAVE PARCIAL**, cuya longitud se especifica en **KEYLENGTH**.
- **KEYLENGTH** es un campo binario de dos bytes que identifica la longitud de la clave especificada en **RIDFLD**. Esta opción debe incluirse si se utiliza una clave genérica, ya que indica la longitud de la clave genérica.
- **SET** especifica el nombre de una celda **BLL**.
- **INTO** identifica el área de trabajo en la que ha de colocarse un registro después de haber sido leído. Si se estuviera trabajando con registros de longitud variable, la descripción del registro ha de ser lo suficientemente grande como para contener el registro de longitud máxima.



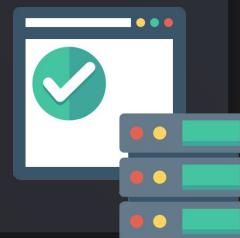


03

...

MANEJO DE ARCHIVOS - READ

- **LENGTH** especifica una media palabra binaria que da la longitud del área de trabajo. Define la longitud del registro más largo que aceptará el programa. La longitud del registro leído se coloca en el campo binario de dos bytes definido por **LENGTH**. Si un registro fuera más largo que lo especificado, se lo trunca y coloca en el área de trabajo, activándose la condición **LENGERR**. La longitud **REAL** del registro se coloca en **LENGTH**.
- **GTEQ** significa que una clave de registro **MAYOR QUE**, o **IGUAL** a la clave especificada en el comando **READ** satisface la búsqueda.
- **EQUAL** significa que una clave de registro **IDÉNTICA** a la clave especificada en el comando **READ** satisface la búsqueda.





03

MANEJO DE ARCHIVOS - READ

...

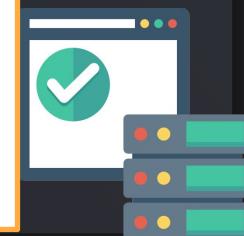
CONDICIONES DE EXCEPCIÓN EN COMANDO READ

Todas las operaciones de Entrada/Salida de datos en archivos pueden provocar condiciones de excepción. Estas condiciones pueden ser especificadas en un comando HANDLE que preceda a un comando READ, WRITE, etc. Luego, de producirse la condición de excepción, el control se transferirá a la sentencia de programa especificada en las opciones de la sentencia HANDLE. Por ejemplo:

```
EXEC CICS HANDLE CONDITION  
    LENGERR (1000-ERR-LONG)  
    NOTFND (1100-SIN-CLIENTE)  
    ERROR   (1200-ERROR-CICS)  
END-EXEC.
```

Los parámetros **1000-ERR-LONG** , **1100-SIN-CLIENTE** y **1200-ERROR-CICS** son nombres de párrafos o secciones del programa de aplicación a los cuales se transferirá el control de producirse cualquiera de las condiciones mencionadas.

La siguiente lista agrupa condiciones de excepción de control de archivos de acuerdo con el tipo de información que generan.





03

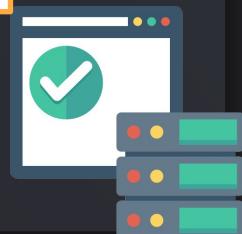
MANEJO DE ARCHIVOS - READ

...

CONDICIONES DE EXCEPCIÓN – INFORMACIÓN AL PROGRAMA

- ENDFILE** se ha alcanzado el fin del archivo
- NOTFND** el registro con la clave especificada no está en el archivo
- DUPREC** se intenta agregar un registro a un archivo con la misma clave de uno que ya existe.

Estas condiciones indican el estado de la operación. El procesamiento puede continuar después de tomar la acción adecuada.





03

...

MANEJO DE ARCHIVOS - READ

CONDICIONES DE EXCEPCIÓN – ERRORES AL PROGRAMA

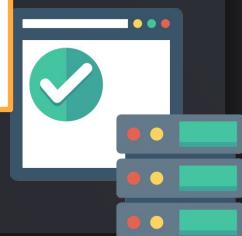
INVREQ pedido inválido (INVALID REQUEST)

LENGERR error de longitud para un registro leído o grabado

DSIDERR el nombre del archivo no está en la FCT

ILLOGIC solamente para VSAM - error de VSAM no cubierto por otras categorías de respuesta CICS.

Estas condiciones indican usualmente un error del programa de aplicación. Estas condiciones no debieran producirse una vez que el programa esté en operación.





03

MANEJO DE ARCHIVOS - READ

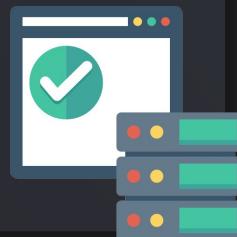
...

CONDICIONES DE EXCEPCIÓN – ERRORES AL PROGRAMA

IOERR error de Entrada/Salida no cubierto por otra condición de excepción CICS.

NOTOPEN el archivo requerido está cerrado.

Estas condiciones indican que el archivo que se está procesando no está disponible.





03

MANEJO DE ARCHIVOS - WRITE

...

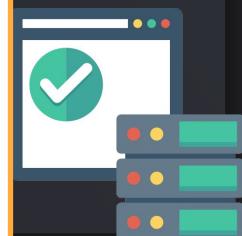
El comando **WRITE** permite la grabación de un nuevo registro en un archivo **VSAM**.

EXEC CICS WRITE DATASET ('name')

```
    RIDFLD (data-area)
    FROM   (data-area)
    [ LENGTH (data-value) ]
    [ { RBA : RRN } ]
    [ MASSINSERT ]
```

END-EXEC.

- **DATASET** especifica el nombre del archivo a grabar (que coincide con el declarado en la tabla **FCT**). Debe tener 8 caracteres como máximo y ajustado a la izquierda.
- **FROM** identifica el área de trabajo en la que se encuentra el registro a grabar.
- **LENGTH** debe usarse únicamente si el archivo contiene registros de longitud variable.
- **MASSINSERT** se usa cuando se deben agregar a un archivo **VSAM KSDS** múltiples registros en secuencia ascendente. Después que los registros han sido grabados, debería usarse el comando **UNLOCK** para desafectar los intervalos de control que son bloqueados durante la ejecución de este comando.





03

MANEJO DE ARCHIVOS - UNLOCK

...

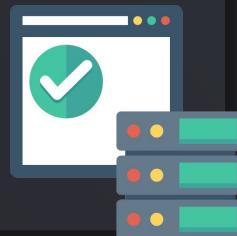
El comando **UNLOCK** permite la liberación de los registros afectados de un archivo que fue accedido con la opción **UPDATE** en un **READ** para que otras transacciones puedan tener acceso a esos registros.

EXEC CICS UNLOCK

DATASET ('name')

END-EXEC.

- **DATASET** especifica el nombre del archivo a liberar (que coincide con el declarado en la tabla FCT). Debe tener 8 caracteres como máximo y ajustado a la izquierda.





03

MANEJO DE ARCHIVOS - REWRITE

...

El comando **REWRITE** permite la actualización de un registro existente en un archivo **VSAM**.

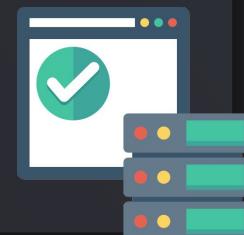
EXEC CICS REWRITE DATASET ('name')

```
    FROM   (data-area)
    [ LENGTH (data-value) ]
```

END-EXEC.

- **DATASET** especifica el nombre del archivo a regrabar (que coincide con el declarado en la tabla **FCT**). Debe tener 8 caracteres como máximo y ajustado a la izquierda.
- **FROM** identifica el área de trabajo en la que se encuentra el registro a regrabar.
- **LENGTH** debe usarse únicamente si el archivo contiene registros de longitud variable.

Para actualizar un registro existente en un archivo **VSAM**, primero se debe leer con un comando **READ** con la opción **UPDATE**. Luego recién se puede emitir este comando para actualizar el registro previamente leído.





03

MANEJO DE ARCHIVOS - DELETE

...

El comando **DELETE** permite la eliminación de un registro de un archivo **VSAM**.

EXEC CICS DELETE DATASET ('name')

```
[ RIDFLD (data-area) ]  
[ KEYLENGTH (data-value) ]  
[ GENERIC ]  
[ NUMREC (data-area) ]  
[ { RBA : RRN } ]
```

END-EXEC.

- **NUMREC** especifica un campo binario de dos bytes donde se indica el número de registros que se desean eliminar. Esta opción debe usarse con **GENERIC**.

Sí el registro que debe eliminarse ha sido leído previamente con el comando **READ** con opción **UPDATE**, solamente la entrada **DATASET** debe ser incluida.

Sí el registro que debe eliminarse no ha sido leído previamente, deberán incluirse las entradas **DATASET Y RIDFLD**.

Como puede verse, existen ciertos problemas potenciales asociados a la eliminación de registros sin haberlos leído previamente. Si no se está absolutamente seguro de lo que se está haciendo, podrían eliminarse en forma accidental registros erróneamente.





03

MANEJO DE ARCHIVOS - STARTBR

...

El comando **STARTBR** permite posicionar a la próxima operación de lectura en la clave que cumpla con determinada condición de búsqueda (Equivale a una sentencia **START** de **COBOL** sobre archivos **VSAM**).

EXEC CICS STARTBR DATASET ('name')

```
[ RIDFLD (data-area) ]  
[ KEYLENGTH (data-value) [ GENERIC ] ]  
[ { GTEQ : EQUAL } ]  
[ { RBA : RRN } ]
```

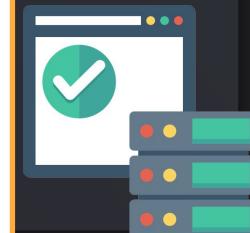
END-EXEC.

Este comando no efectúa lectura de registros y se complementa con los comandos **READNEXT**, **READPREV**, **RESETBR** y **ENDBR**.

Para posicionarnos al comienzo del archivo inicializar el **RIDFLD** con **LOW-VALUES** o especificar **GENERIC** con **KEYLENGTH(0)** para luego leer secuencialmente los registros según su clave con **READNEXT** en forma ascendente.

Para posicionarnos al final del archivo inicializar el **RIDFLD** con **HIGH-VALUES** sin especificar las opciones **GENERIC** y **KEYLENGTH** y leer secuencialmente los registros según su clave con **READPREV** en forma descendente.

Con la opción **GTEQ**, si no existe la clave en el archivo, la posición se establece en el siguiente registro. Pero con la opción **EQUAL** esta circunstancia arroja **NOTFND**.





03

MANEJO DE ARCHIVOS - READNEXT

...

El comando **READNEXT** permite la lectura del siguiente registro de un archivo **VSAM** que se encuentra bajo el efecto del comando **STARTBR**.

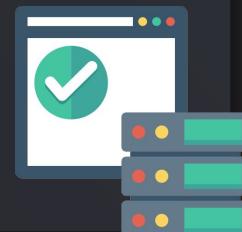
EXEC CICS READNEXT DATASET ('name')

```
RIDFLD (data-area)
[ KEYLENGTH (data-value) [ GENERIC ] ]
[ {RBA : RRN } ]
[ { SET (pointer-ref) : INTO (data-area) } ]
[ LENGTH (data-area) ]
```

END-EXEC.

- **RIDFLD** identifica el nombre simbólico del área que contiene la clave del registro a leer, debiendo ser la misma que la utilizada en el comando **STARTBR**. El contenido de **RIDFLD** es actualizado por el CICS en forma automática con el valor del nuevo registro leído.

Se puede utilizar una técnica conocida como '**SKIP-SEQUENTIAL-PROCESSING**', que consiste en modificar el valor del **RIDFLD** previo a ejecutar el siguiente **READNEXT** a fin de saltar la lectura de los registros existentes entre el valor de la última clave accedida y el nuevo valor del **RIDFLD**.





03

MANEJO DE ARCHIVOS -READPREV

...

El comando **READPREV** permite la lectura del registro anterior de un archivo **VSAM** que se encuentra bajo el efecto del comando **STARTBR**.

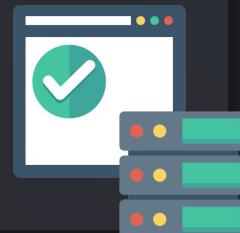
EXEC CICS READPREV DATASET ('name')

```
RIDFLD (data-area)
[ KEYLENGTH (data-value) [ GENERIC ] ]
[ { RBA : RRN } ]
[ { SET (pointer-ref) : INTO (data-area) } ]
[ LENGTH (data-area) ]
```

END-EXEC.

El registro identificado en el **STARTBR** debe existir para que la operación secuencial continúe. Si el registro requerido por el primer comando **READPREV** no está en el archivo, se activa la condición **NOTFND**.

No utilizar **GENERIC** en el comando **STARTBR** para iniciar una búsqueda hacia atrás;, caso contrario se activa la condición **INVREQ**.





03

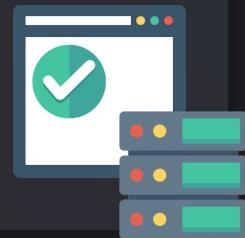
MANEJO DE ARCHIVOS - ENDBR

...

El comando **ENDBR** permite terminar una búsqueda liberando los recursos afectados.

```
EXEC CICS ENDBR  
  DATASET ('name')  
END-EXEC.
```

Este comando fallará si se ejecuta cuando el comando **STARTBR** no fue ejecutado exitosamente.





03

MANEJO DE ARCHIVOS - RESETBR

...

El comando **RESETBR** permite terminar una búsqueda y comenzar otra sobre el mismo archivo inmediatamente. Equivale a ejecutar un comando **ENDBR** y seguidamente el comando **STARTBR**.

EXEC CICS RESETBR DATASET ('name')

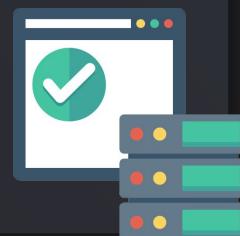
RIDFLD (data-area)

[KEYLENGTH (data-value) [GENERIC]]

[{ RBA : RRN }]

[{ GTEQ : EQUAL }]

END-EXEC.



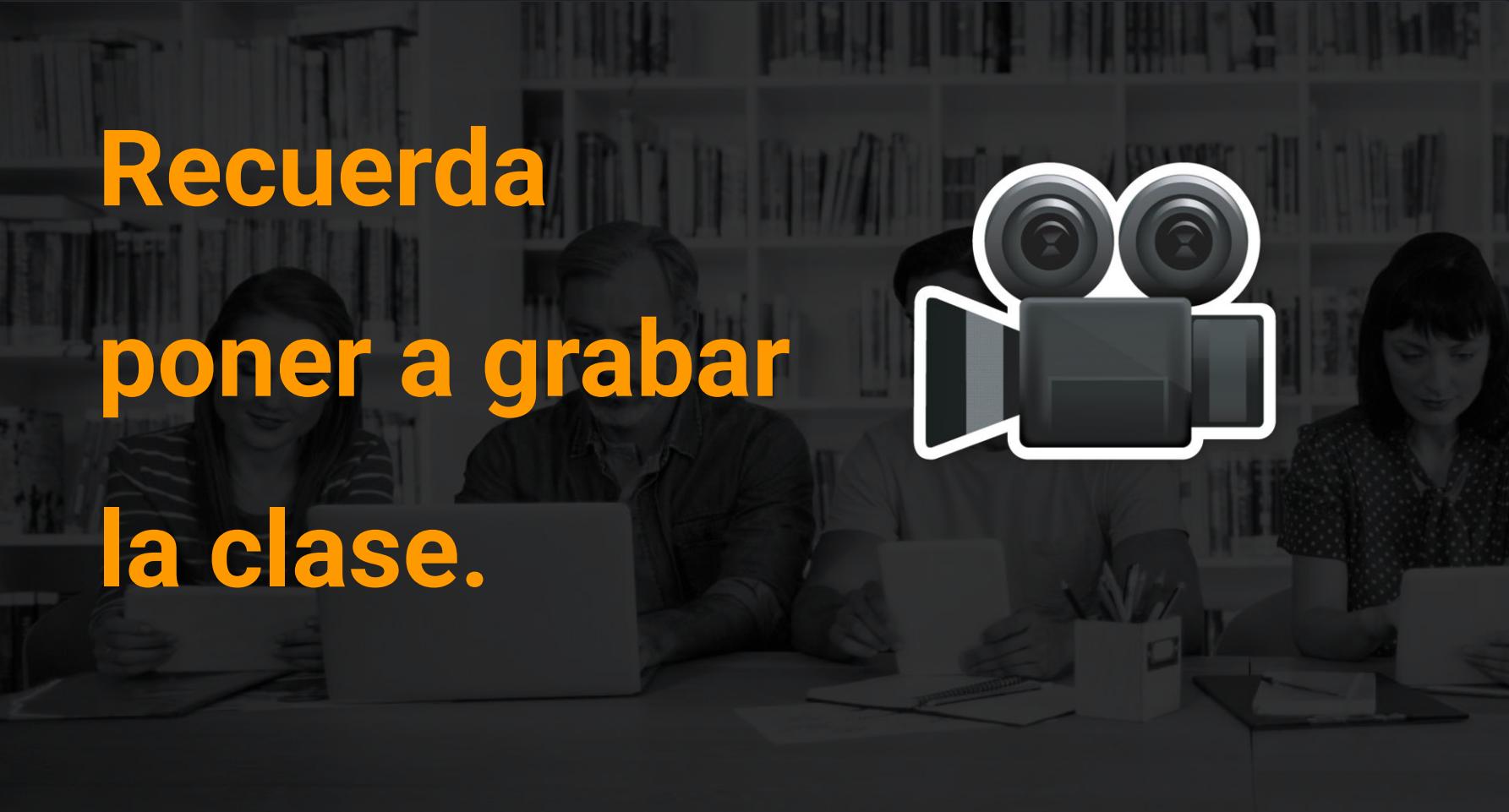
**¡Muchas
gracias!**



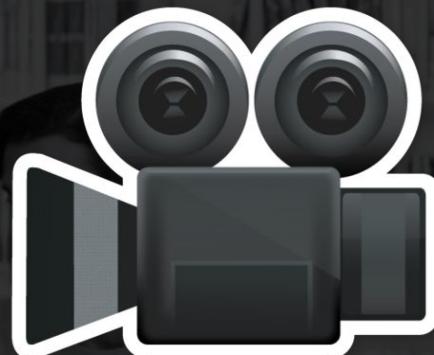


Curso

PROGRAMADOR COBOL – Introducción a CICS



Recuerda
poner a grabar
la clase.





INTRODUCCIÓN A CÓDIGO COBOL CICS



... Mapa Conceptual

01
CONCEPTOS Y
FACILIDADES

02
Preparación
de un
programa

03
Generación de
mapas - BMS

04
Control de
programas



... **Agenda**

1

Conceptos y facilidades COBOL CICS

2

Preparación de un programa COBOL CICS

3

Basic Mapping Support – Generación de mapas

4

Códigos de control de programa





01

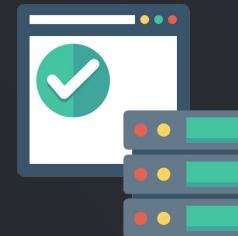
... INTRODUCCIÓN AL CICS - TEMARIO

CONCEPTOS Y FACILIDADES

- Estructura del CICS, apreciación global de los componentes y Tablas de recursos
- Concepto de Programación conversacional y Pseudo Conversacional
- El ciclo de un programa Bajo CICS

PREPARACIÓN DE UN PROGRAMA COBOL

- Estructura de Comandos de CICS dentro de un programa
- Codificación del programa fuente
- Proceso de Compilación
- Alta del Programa en el CICS
- Alta de la Transacción en el CICS
- NEW COPY





01

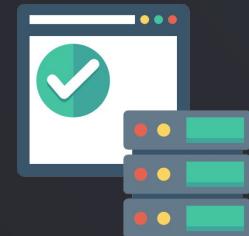
... INTRODUCCIÓN AL CICS - TEMARIO

GENERACIÓN DE MAPAS - BMS

- Codificación de Fuente BMS
- Proceso de Compilación
- Alta del MAPA en el CICS
- NEW COPY

CONTROL DE PROGRAMAS

- COMMAREA
- LINK
- XCTL
- RETURN
- START/RETRIEVE
- EIB - EXEC INTERFACE BLOCK





01

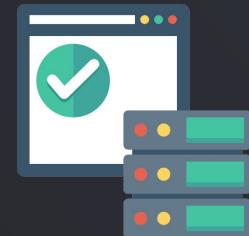
... INTRODUCCIÓN AL CICS - TEMARIO

MANEJO DE ERRORES DEL PROGRAMA

- Testeo de Condiciones de Excepción en la ejecución de Comandos CICS
- HANDLE CONDITION
- RESP
- HANDLE AID

COMANDOS PARA ACCESO A DATOS

- Obtención de fecha
- Comandos para manejo de Mapas
- Acceso a Archivos VSAM
- Acceso a TS Temporary Storage
- Acceso a TD Transient Data





01

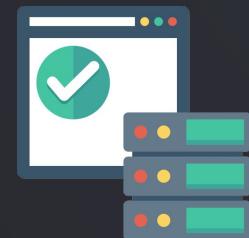
... INTRODUCCIÓN AL CICS - TEMARIO

UNIDAD LÓGICA DE TRABAJO

- CONCEPTOS
- SYNCPOINT
- SYNCPOINT ROLLBACK
- ABEND

DEMOSTRACIÓN DE CEDF

DEMOSTRACIÓN DE CEMT CECI CEDA



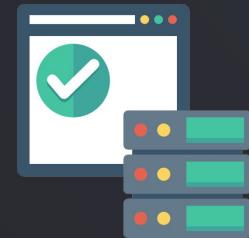


01

...

CONCEPTOS Y FACILIDADES

- Estructura del CICS, apreciación global de los componentes y Tablas de recursos
 - Concepto de Programación conversacional y Pseudo Conversacional
 - El ciclo de un programa Bajo CICS





01

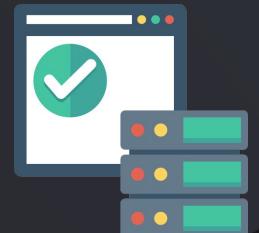
...

ESTRUCTURA DEL CICS

El CICS es una monitor de comunicaciones bajo el que se pueden desarrollar 'TRANSACCIONES' ON-Line en diversos Equipos (Mainframes, PC's, etc.) y en diversas plataformas (MVS, VSE, RS/6000, Etc.)

Los componentes básicos para implementar una aplicación de gestión son:

- **Programas:** Desarrollados bajo los lenguajes de programación COBOL, ASSEMBLER, PL/I,RPG, que una vez compilados para CICS se podrán asociar a una Transacción CICS
- **Mapas:** Son las pantallas que permiten la interacción de la aplicación con el usuario. Por medio de estas se le permite el ingreso de datos o efectuar la visualización de un resultado
- **Transacciones:** Son las unidades lógicas de procesamiento. Los nombres de las transacciones son de 4 caracteres y son únicas en cada sesión de CICS. Hay una relación unívoca
- **Archivos:** Las estructuras de archivos que son soportadas por el CICS son VSAM (KSDS, ESDS, RRDS) y BDSM.
- **Bases de Datos:** Existen diversos tipos de Bases de Datos soportadas, IMS/DB (Jerárquica) y DB2 (Relacional).
- **Terminales:** Son cualquier dispositivo que pueda conectarse al CICS por medio de un método de acceso de telecomunicaciones o protocolos. Podemos incluir terminales (teclados y pantallas), impresoras, y computadoras en general.





01

...

ESTRUCTURA DEL CICS

Para relacionar todos los componentes básicos de nuestra aplicación, el CICS usa tablas internas, y que, a su vez, también las utiliza para mantener el control de todos sus recursos y actividades.

Algunas de las tablas que el CICS utiliza son las siguientes:

- **FCT (File Control Table):** Todos los archivos que deban ser accedidos por nuestras aplicaciones deberán estar declarados con una entrada en esta tabla. La relación consiste en el nombre del DATASET para el CICS, que consta de 8(ocho) caracteres, y el nombre real o label que este posee en el disco.

Otros atributos, como la longitud de la clave, la longitud del registro, si es fijo o variable y las acciones que se pueden efectuar sobre el archivo (lectura, grabación, actualización y borrado) también están contenidas en esta tabla, como así también un status sobre si el archivo está abierto para el CICS y disponible.

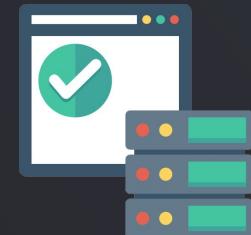
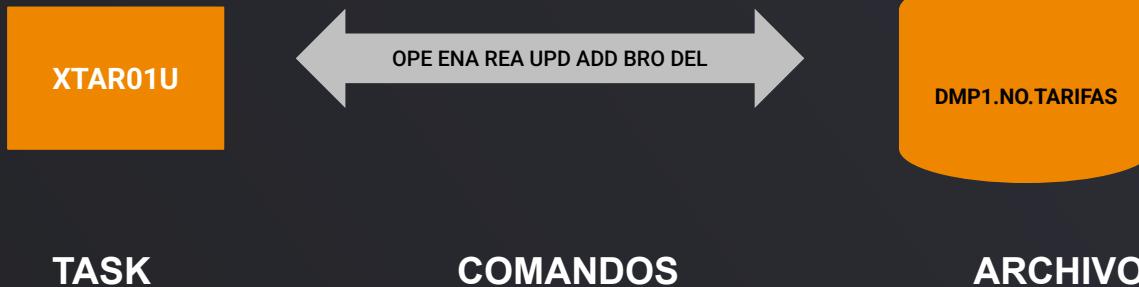




01

...

ESTRUCTURA DEL CICS



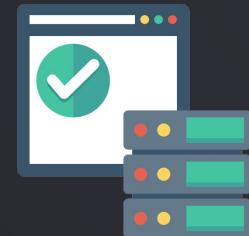


01

...

ESTRUCTURA DEL CICS

-TCT (Terminal Control Table): Por cada terminal asociada al CICS existe una entrada (TCT) que describe el tipo de dispositivo y su dirección. Contiene también un pointer al Buffer que fue asignado para esa terminal (TIOA Terminal i/o Área). En la tabla TCTTE, el CICS establece una relación entre la tarea asociada a cada terminal, lo que posibilita que, ante una respuesta del usuario sobre una terminal, se dispare la transacción correcta. Esto se lleva a cabo por un pointer a la TCA (Task Control Área) y puede leerse desde una aplicación consultando al EIB (Executive Interface Block), que se verá más adelante.

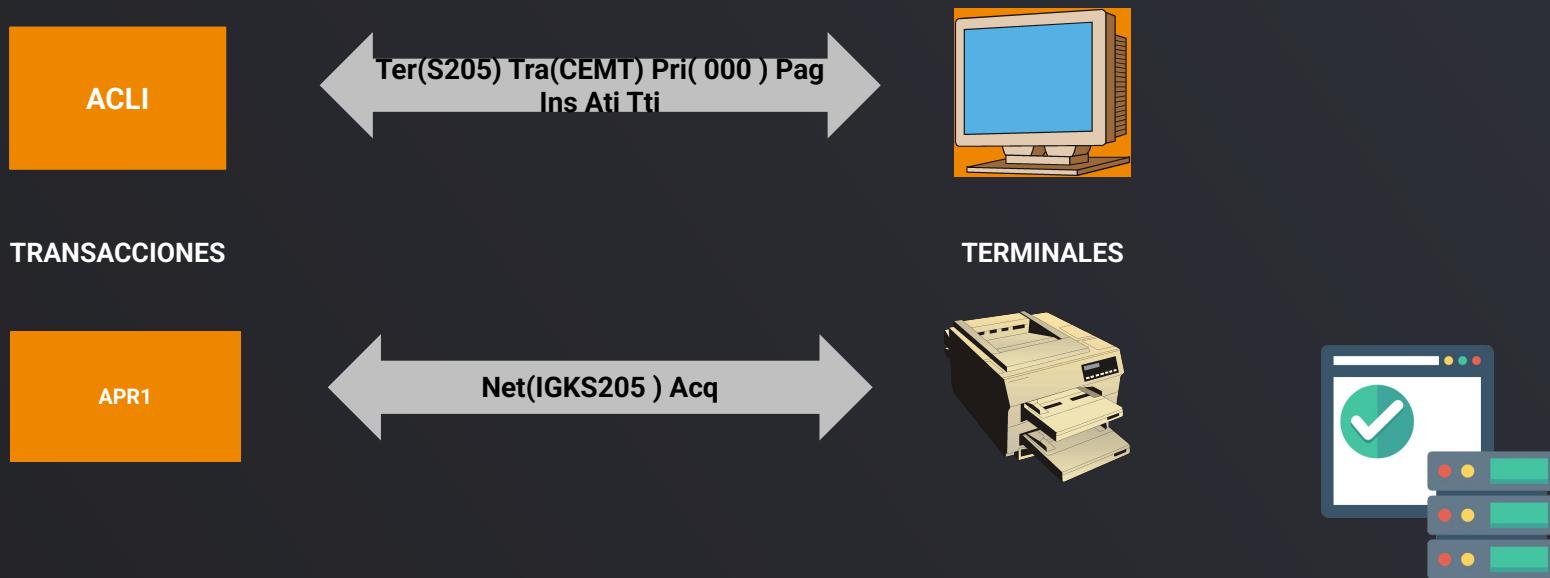




01

...

ESTRUCTURA DEL CICS



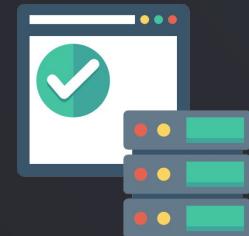


01

...

ESTRUCTURA DEL CICS

-PCT (Program Control Table): Establece la relación transacción/programa. Tiene una entrada por cada transacción a invocarse en el ambiente de CICS. Al ser solicitada la ejecución de una transacción, el CICS automáticamente le asigna un número único de tarea. Esta administración ejercida por el CICS nos permite que una misma transacción pueda ser invocada desde una o distintas terminales repetidamente. Las tareas simultáneas compartirán el mismo código ejecutable, archivos, bases de datos, pero no compartirán las áreas de memoria. Una vez accionada una transacción, el CICS busca el programa asociado en la PCT y lo cargará en memoria si es la primera vez que es invocado (por consulta a la PPT) y lo asociará a la terminal que corresponda.

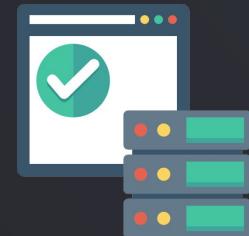
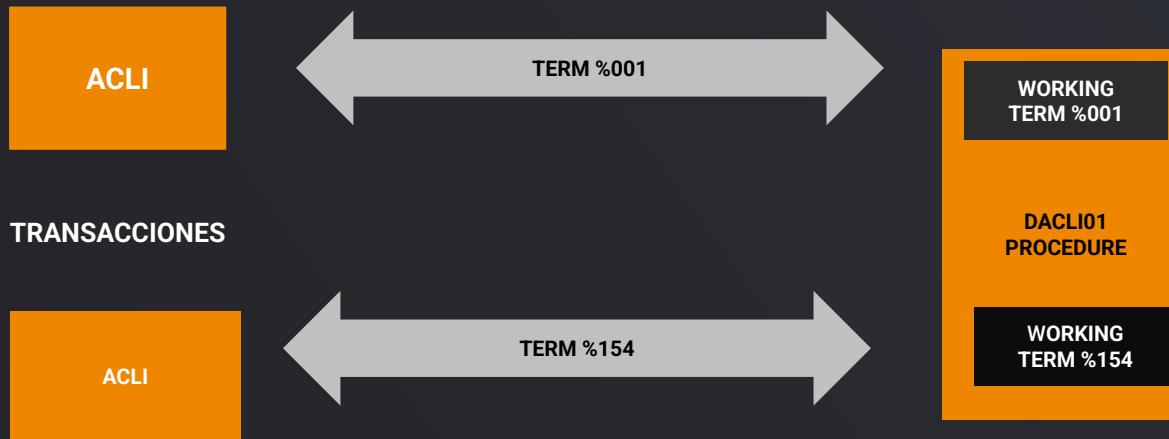




01

...

ESTRUCTURA DEL CICS





01

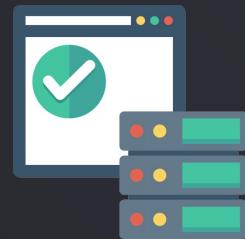
...

ESTRUCTURA DEL CICS

PPT (Processing Program Table): Tiene una entrada por cada programa y mapa a ser utilizado. Genera una asociación programa/ubicación de memoria en que reside y que se establece con el manejo de punteros (pointers) .

Si el programa asociado está ya en memoria, la PPT proporciona su ubicación; de lo contrario, lo carga previamente.

A diferencia de la PCT, en esta tabla también se incluyen los programas que no serán invocados por una transacción (tal el caso de los mapas).

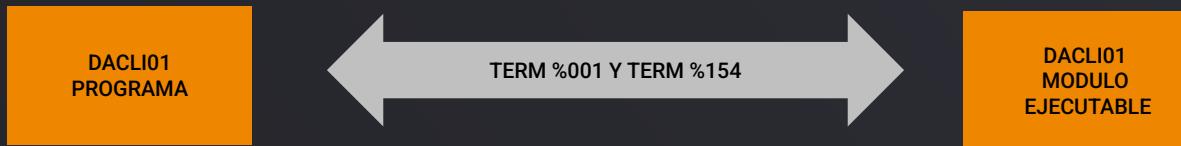




01

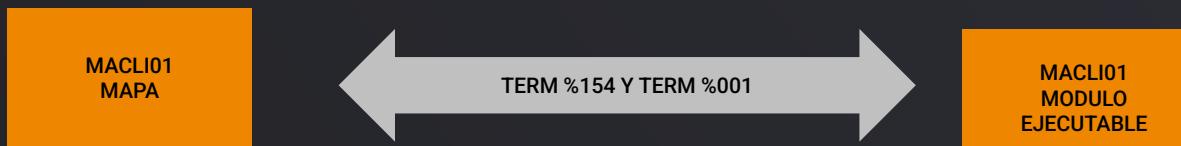
...

ESTRUCTURA DEL CICS



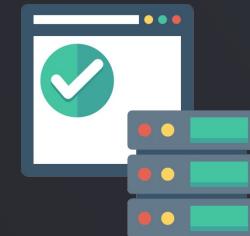
PROGRAMAS

DIRECCIÓN DEL
EJECUTABLE



MAPA

MODULO
EJECUTABLE





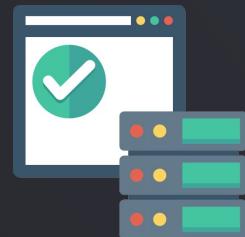
01

...

ESTRUCTURA DEL CICS

Todos los elementos mencionados precedentemente, programas, archivos, terminales, etc., pueden ser modificados con la facilidad de RDO (Resource Definition Online) que se accede mediante el uso de la transacción CEDA. Esta permite agregar, modificar e instalar dinámicamente todos los recursos necesarios para las aplicaciones de negocio que corren bajo CICS.

Esto nos permite disponer rápidamente de los programas, mapas, etc., sin tener que esperar a que se recarguen las tablas por la bajada y subida del CICS.



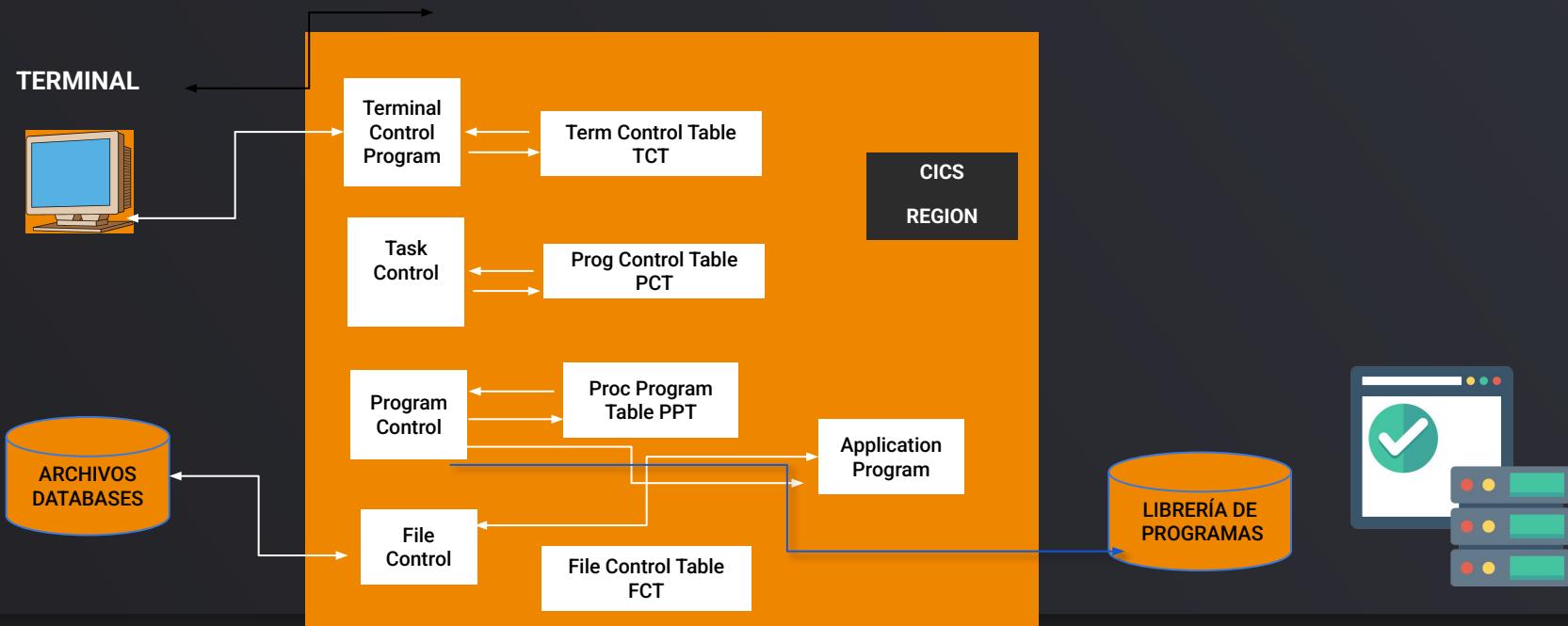


01

...

ESTRUCTURA DEL CICS

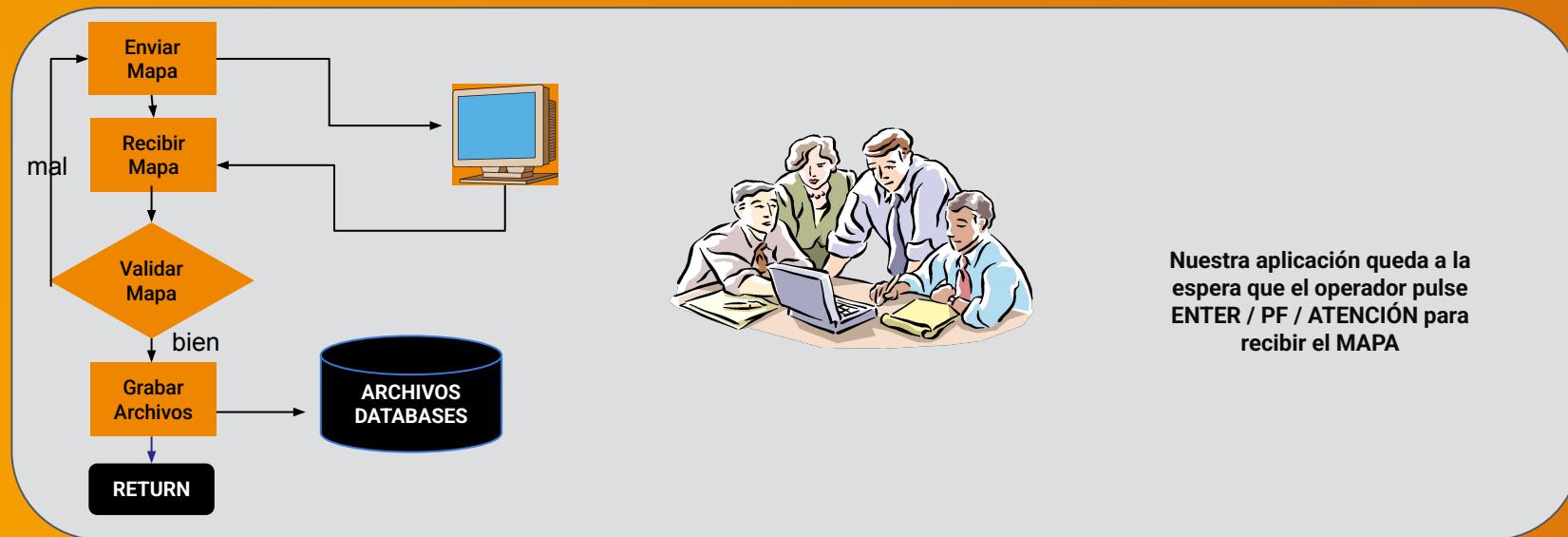
Los administradores de CICS son los encargados del mantenimiento de estas tablas (macros de Assembler) de relaciones entre transacciones/programas/archivos.





PROGRAMACIÓN CONVERSACIONAL

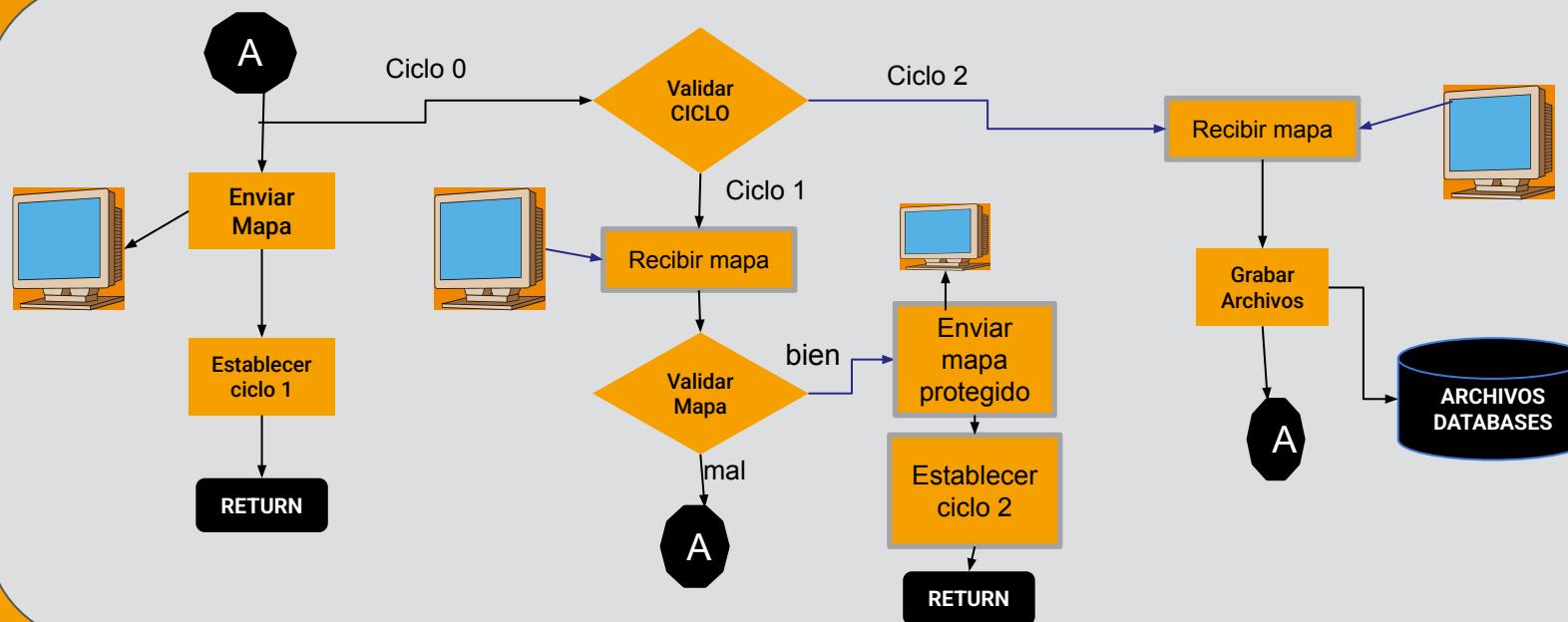
Este método de programación bajo el CICS no es el recomendable, ya que los recursos asignados a la transacción quedan tomados hasta la finalización de la misma. El tiempo de respuesta de la transacción es dependiente de la respuesta del usuario de la misma.





PROGRAMACIÓN PSEUDO-CONVERSACIONAL

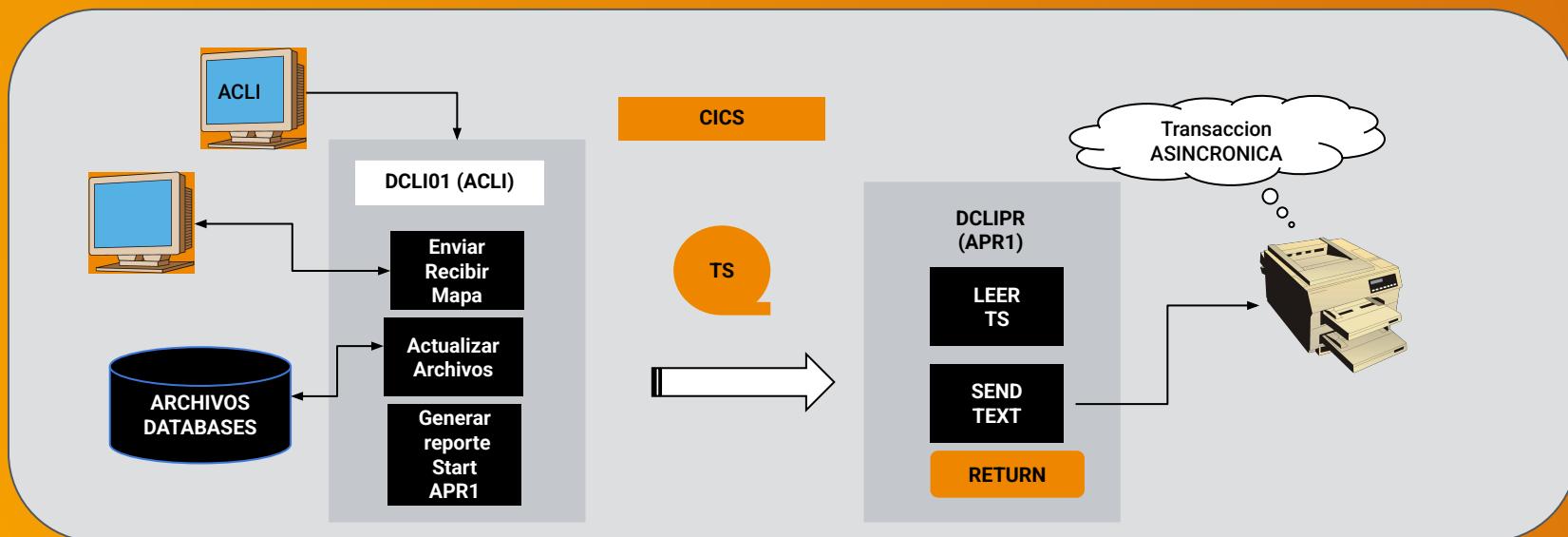
Este método de programación bajo el CICS **es el más recomendable**, ya que los recursos quedan liberados al momento de terminar nuestra transacción y ésta no depende del tiempo de respuesta del Usuario/Operador. Ejemplo de disparo serializado de transacciones.





PROGRAMACIÓN CONVERSACIONAL

Este método de programación bajo el CICS no es el recomendable, ya que los recursos asignados a la transacción quedan tomados hasta la finalización de la misma. El tiempo de respuesta de la transacción es dependiente de la respuesta del usuario de la misma.





... Mapa Conceptual

01
CONCEPTOS Y
FACILIDADES

02
Preparación
de un
programa

03
Generación de
mapas - BMS

04
Control de
programas





02

...

PREPARACIÓN DE UN PROGRAMA CICS

- **Estructura de comandos de CICS dentro de un programa**
- **Codificación del programa fuente**
- **Proceso de compilación**
- **Alta del programa en tabla de CICS**
- **Alta de transacción en tabla de CICS**
- **NEW COPY (programa objeto)**



02

...

ESTRUCTURA DE COMANDOS CICS

Dentro la PROCEDURE DIVISION de un programa COBOL ON-LINE, las llamadas a las funciones del CICS deberán tener siempre la siguiente estructura:

EXEC CICS

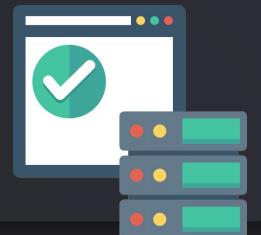
...

**SINTAXIS: FUNCTION, OPCION, ARGUMENTOS Y PARÁMETROS
PROPIOS DEL COMANDO DE CICS**

...

END-EXEC.

Al Mantener esta estructura, le estamos indicando al 'TRANSLATOR' (precompilador de comandos de CICS) donde comienza y termina el comando de CICS que deberá traducir y resolver. Luego de convertido el comando, se entrega al compilador COBOL una fuente que este comprende.





02

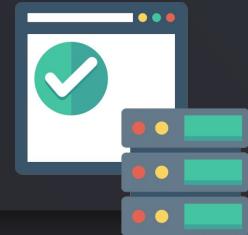
...

ESTRUCTURA DE COMANDOS CICS

El formato en líneas generales de un comando CICS es EXECUTE CICS (o EXEC CICS) seguido por el nombre de la función requerida, y la posibilidad de una o más opciones, según lo siguiente:

EXEC CICS command option(arg).... END-EXEC. Donde:

- **Command:** Describe la operación requerida (por ejemplo READ).
- **Option:** Describe cualquiera de las tantas facilidades opcionales disponibles para cada función requerida. Algunas opciones pueden estar seguidas por un argumento el que va entre paréntesis. El orden de los argumentos no es condicionante de error de sintaxis.
- **Arg:**(abreviatura de argumento) es un valor tal como un "data-value" o "name". Un "data-value" puede ser tanto una variable como una constante. De esta forma un argumento que envía datos al CICS se denomina "data-value", mientras que un argumento que espera recibir datos del CICS se denomina "data-area". Algunos argumentos descritos en un comando como "data-area" pueden cumplir ambas características (tal el caso de LENGTH). En tal situación, deberemos asegurarnos que la "data-area" no se encuentre en una parte protegida de la memoria. Si el argumento hace referencia a nombres externos al programa, este deberá estar contenido en una variable de working o codificarlo entre apóstrofos (ws-file) o ('XCLI01U').





02

...

ESTRUCTURA DE COMANDOS CICS

Valores posibles para un Arg (argumento) en un programa COBOL:

- ‘**data-value**’ o ‘**data-área**’ podrán ser reemplazados por cualquiera de las siguientes opciones, siempre que su contenido esté acorde al tipo de dato esperado por el comando y podrán ser, por ejemplo, correspondientes a alguna de las siguientes definiciones:
 - Halfword binary - PIC S9(4) COMP
 - Fullword binary - PIC S9(8) COMP
 - Character string - PIC X(n) where ‘n’ is the number of bytes

Donde el tipo de dato no está estrictamente especificado, la ‘data-área’ bien podrá ser un campo elemental o un ítem de grupo



02

...

ESTRUCTURA DE COMANDOS CICS

Valores posibles para un Arg (argumento) en un programa COBOL(cont.):

- **Ptr-ref(pointer-ref):** nombre de una celda BLL (**B**ase **L**ocator for **L**inkage)
- **Ptr-val(pointer-value):** nombre de una celda BLL o un área de datos que contiene el nombre de la celda BLL.
- **Name:** Literal que referencia nombres externos al programa o un área de datos que contenga un literal. Si es literal debe estar entre apóstrofes.
- **Label:** Un nombre de párrafo o de SECTION de COBOL (se realiza una derivación de control incondicional)
- **Hhmmss:** Literal numérico o área de datos PIC S9(7) PACK, que contiene la hora expresada como +0hhmmss



...

02

ESTRUCTURA DE COMANDOS CICS

Ejemplo de comando READ:

Sintaxis:

```
EXEC CICS READ  
    DATASET ('filename')  
    [UPDATE]  
    RIDFIELD (data-area)          (WS-KEY-CLIENTE)  
    [KEYLENGTH(data-value)[GENERIC]] (WS-LEN-CLAVE)  
    [RBA : RRN]  
    {SET(pointer-ref) : INTO(data-area)} (WS-PTR-REG-CLI) (WS-REG-CLI)  
    [LENGTH(data-area)]           (WS-LEN-REG-CLI)  
    [GTEG : EQUAL]  
END-EXEC.
```

WORKING STORAGE SECTION.

```
77 WS-FILE      PIC X(08) VALUE 'XCLI01U'.  
77 WS-LEN-CLAVE PIC S9(9) COMP.  
77 WS-PTR-REG-CLI PIC S9(9) POINTER.  
77 WS-LEN-REG-CLI PIC S9(9) COMP.  
01 WS-REG-CLI.  
    05 WS-KEY-CLIENTE   PIC X(19) VALUE ''.  
    05 WS-RESTO-CLIENTE PIC X(1001) VALUE ''.
```



Diferencias en la organización del programa ON-LINE respecto de un BATCH:

```
*****  
IDENTIFICATION DIVISION.  
*****
```

```
PROGRAM-ID. DDTC001.  
AUTHOR. EDUARDO A. PALMEYRO  
DATE-WRITTEN. 01-10-96.  
*****
```

```
ENVIRONMENT DIVISION.  
*****
```

```
CONFIGURATION SECTION.  
SOURCE-COMPUTER. IBM-HOST.  
OBJECT-COMPUTER. IBM-HOST.  
*****
```

```
SPECIAL-NAMES.  
DECIMAL-POINT IS COMMA.  
*****
```

```
DATA DIVISION.  
*****
```

```
WORKING-STORAGE SECTION.  
01 FILLER      PIC X(20) VALUE '*-> WORKING'.  
*****
```

En un ON-LINE no se necesitan declarar sentencias SELECT, ya que los archivos son accedidos por sentencias propias del CICS como por ejemplo el READ.

~~INPUT-OUTPUT SECTION.
-----*
FILE-CONTROL.
-----*~~

~~SELECT XPTM01I ASSIGN TO XPTM01I
FILE STATUS FS-XPTM01I.~~





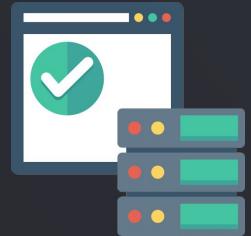
02

CODIFICACIÓN PROGRAMA FUENTE

...
Inclusión en nuestra WORKING STORAGE SECTION de los copys que se encuentran disponibles para desarrollar programas bajo CICS:

WORKING-STORAGE SECTION.

```
01 FILLER      PIC X(20) VALUE '*-> WORKING'.  
*--> AREAS DE COPIES          *  
01 FILLER      PIC X(20) VALUE '*-> MAP PPTC 001  '.  
COPY MPTC001.  
  
01 FILLER      PIC X(20) VALUE '*-> DFHAID      '.  
COPY DFHAID.  
  
01 FILLER      PIC X(20) VALUE '*-> DFHBMSCA      '.  
COPY DFHBMSCA.  
  
01 FILLER      PIC X(20) VALUE '*-> ATRIBUTO      '.  
COPY ATRIBUTO.  
  
01 FILLER      PIC X(20) VALUE '*-> COM-COMMAREA  '.  
COPY WAPTCOM.
```





02

CODIFICACIÓN PROGRAMA FUENTE

Inclusión en nuestra WORKING STORAGE SECTION de los copys que se encuentran disponibles para desarrollar programas bajo CICS:

WORKING-STORAGE SECTION.

```
*-----*
01 FILLER      PIC X(20) VALUE '*-> WORKING'.

*-----*
*--> AREAS DE COPIES          *
*-----*
01 FILLER      PIC X(20) VALUE '*-> MAP PPTC 001'.
COPY MPTC001.

01 FILLER      PIC X(20) VALUE '*-> DFHAID.
COPY DFHAID.

01 FILLER      PIC X(20) VALUE '*-> DFHBMSCA.
COPY DFHBMSCA.

01 FILLER      PIC X(20) VALUE '*-> ATRIBUTO.
COPY ATRIBUTO.

01 FILLER      PIC X(20) VALUE '*-> COM-COMMAREA.
COPY WAPTCOM.
```

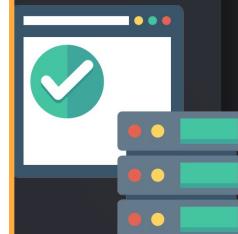
Esta copy representa el diseño del área de datos común que vamos a utilizar entre dos o más programas de nuestra aplicación. Se refiere al copy de la Commarea (Common area).

Los programas que utilicen Commarea como facilidad para intercomunicación de datos deberán formatearla tal como se espera que otro programa la reciba; de lo contrario, estaremos pasando parámetros que no se los podrá interpretar por formato inválido.

01 COM-COMMAREA.

03 COM-SUCURSAL	PIC 9(003).
03 COM-NRO-CUENTA	PIC 9(007).
03 COM-TIPO-CUENTA	PIC X(004).

Si nuestro programa utiliza TS (Temporary Storage) o TD (Transient Data) como facilidades de intercomunicación de datos, también deberemos incorporar sus copys o diseños de registro en nuestra working.





02 CODIFICACIÓN PROGRAMA FUENTE

...

Inclusión en nuestra WORKING STORAGE SECTION de los copys que se encuentran disponibles para desarrollar programas bajo CICS:

WORKING-STORAGE SECTION.

01 FILLER PIC X(20) VALUE '*-> WORKING'.

*--> AREAS DE COPIES *

01 FILLER PIC X(20) VALUE '*-> MAP PPTC 001 :
COPY MPTC001.

01 FILLER PIC X(20) VALUE '*-> DFHAID :
COPY DFHAID.

01 FILLER PIC X(20) VALUE '*-> DFHBMSCA :
COPY DFHBMSCA.

01 FILLER PIC X(20) VALUE '*-> ATRIBUTO :
COPY ATRIBUTO.

01 FILLER PIC X(20) VALUE '*-> COM-COMMAREA :
COPY WAPTCOM.

Este Copy contiene el diseño de registro del mapa asociado al programa.
Esta asociación se realiza por el solo hecho de utilizarlo.

Incorporaremos tantos copys de mapas como sean necesarios para completar el objetivo del programa y que hemos diseñado bajo BMS o algún utilitario para generación de mapas (SDF).

En la mayoría de las instalaciones se determina un mapa por programa. Tales es así que la nomenclatura del mapa y del programa cambia solamente en la primera letra (o en la última, según la instalación)

Programa: DDT001

Mapa : MPTC001

Los campos del copy siguen la siguiente convención:

Campos de input, terminados en I (NOMBRE)

Campos de Output, terminados en O (NOMBREO)

Campos de atributos, terminados en A (NOMBREA)

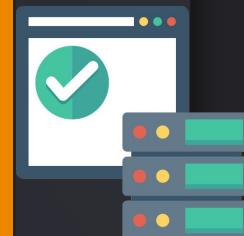
Campo para posicionar el cursor, terminados en L (NOMBRE)

MOVE 'APP' TO NOMBREO.

MOVE -1 TO NOMBREL.

MOVE UAHS TO NOMBREA.

PERFORM 5000-SEND-MAPA THRU FIN-5000.





02 CODIFICACIÓN PROGRAMA FUENTE

Inclusión en nuestra WORKING STORAGE SECTION de los copys que se encuentran disponibles para desarrollar programas bajo CICS:

WORKING-STORAGE SECTION.

```
*-----*  
01 FILLER      PIC X(20) VALUE '*-> WORKING'.  
*-----*
```

```
*-----*  
*-> AREAS DE COPIES           *  
*-----*
```

```
01 FILLER      PIC X(20) VALUE '*-> MAP PPTC 001'.  
COPY MPTC001.
```

```
01 FILLER      PIC X(20) VALUE '*-> DFHAID'.  
COPY DFHAID.
```

```
01 FILLER      PIC X(20) VALUE '*-> DFHBMSCA'.  
COPY DFHBMSCA.
```

```
01 FILLER      PIC X(20) VALUE '*-> ATRIBUTO'.  
COPY ATRIBUTO.
```

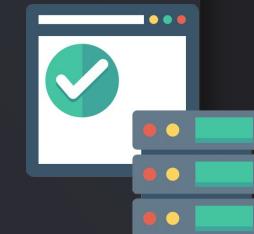
```
01 FILLER      PIC X(20) VALUE '*-> COM-COMMAREA'.  
COPY WAPTCOM.
```

Este Copy contiene los valores para hacer posible la interpretación de las teclas pulsadas por usuario operador de nuestra aplicación.

El valor de cada tecla pulsador es devuelto por la **EIB EXEC INTERFASE BLOCK** y dicho valor se compara con cada variable incluida en el copy DFHAID. Todos los campos de la EIB están disponibles por nuestro programa sin que haya que declarar copy alguna para ello.

El campo de la **EIB** que nos devuelve la tecla pulsada es **EIBAID** y se evalúa de la siguiente forma:

```
EVALUATE EIBAID  
  WHEN DFHENTER  
    PERFORM PFKEY-ENTER  
    THRU FIN-PFKEY-ENTER  
  WHEN DFHPF2  
    PERFORM PFKEY-PF2  
    THRU FIN-PFKEY-PF2  
  WHEN OTHER  
    PERFORM PFKEY-INVALIDA  
    THRU FIN-PFKEY-INVALIDA  
END-EVALUATE.
```





02 CODIFICACIÓN PROGRAMA FUENTE

...

Inclusión en nuestra WORKING STORAGE SECTION de los copys que se encuentran disponibles para desarrollar programas bajo CICS:

```
WORKING-STORAGE SECTION.  
*-----*  
01 FILLER      PIC X(20) VALUE '*-> WORKING'.  
  
*-----*  
*--> AREAS DE COPIES           *  
*-----*  
  
01 FILLER      PIC X(20) VALUE '*-> MAP PPTC 001'.  
COPY MPTC001.  
  
01 FILLER      PIC X(20) VALUE '*-> DFHAID'.  
COPY DFHAID.  
  
01 FILLER      PIC X(20) VALUE '*-> DFHBMSCA'.  
COPY DFHBMSCA.  
  
01 FILLER      PIC X(20) VALUE '*-> ATRIBUTO'.  
COPY ATRIBUTO.  
  
01 FILLER      PIC X(20) VALUE '*-> COM-COMMAREA'.  
COPY WAPTCOM.
```

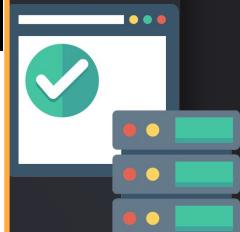
Esta Copy contiene los valores para hacer posible la asignación de atributos a los campos del mapa con todas las combinaciones posibles. (para mapas de pantallas como de impresión).

La asignación de atributos al campo del mapa se efectúa moviendo la variable deseada al campo del mapa terminado en 'A'.

MOVE DFHUNNOD TO NOMBREA

Este move hace que campo del mapa donde se muestra el nombre asuma los siguientes atributos:

Unprotected, nondisplay, nonprint, nondetectable, MDT





02 CODIFICACIÓN PROGRAMA FUENTE

Inclusión del área de comunicación COMMAREA (COMMON AREA) en la LINKAGE SECTION:

WORKING-STORAGE SECTION.

```
*-----*  
01 FILLER      PIC X(20) VALUE '*-> WORKING'.
```

LINKAGE SECTION.

```
*****  
01 DFHCOMMAREA    PIC X(17408).
```

```
*****
```

PROCEDURE DIVISION.

```
*****
```

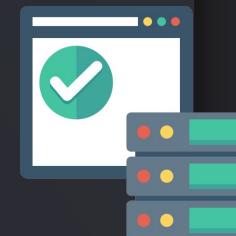
*CUERPO PRINCIPAL DEL PROGRAMA.

```
PERFORM 100-INICIO  
THRU 100-F-INICIO.
```

Esta área definida en la Linkage Section nos permitirá recibir un área común de datos que otro programa nos envía.

El tamaño máximo de esta área varía con las posibilidades de cada equipo.

El tamaño mínimo será el requerido por nuestra aplicación para efectuar el proceso en la forma esperada.

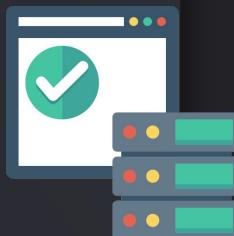




02 CODIFICACIÓN PROGRAMA FUENTE

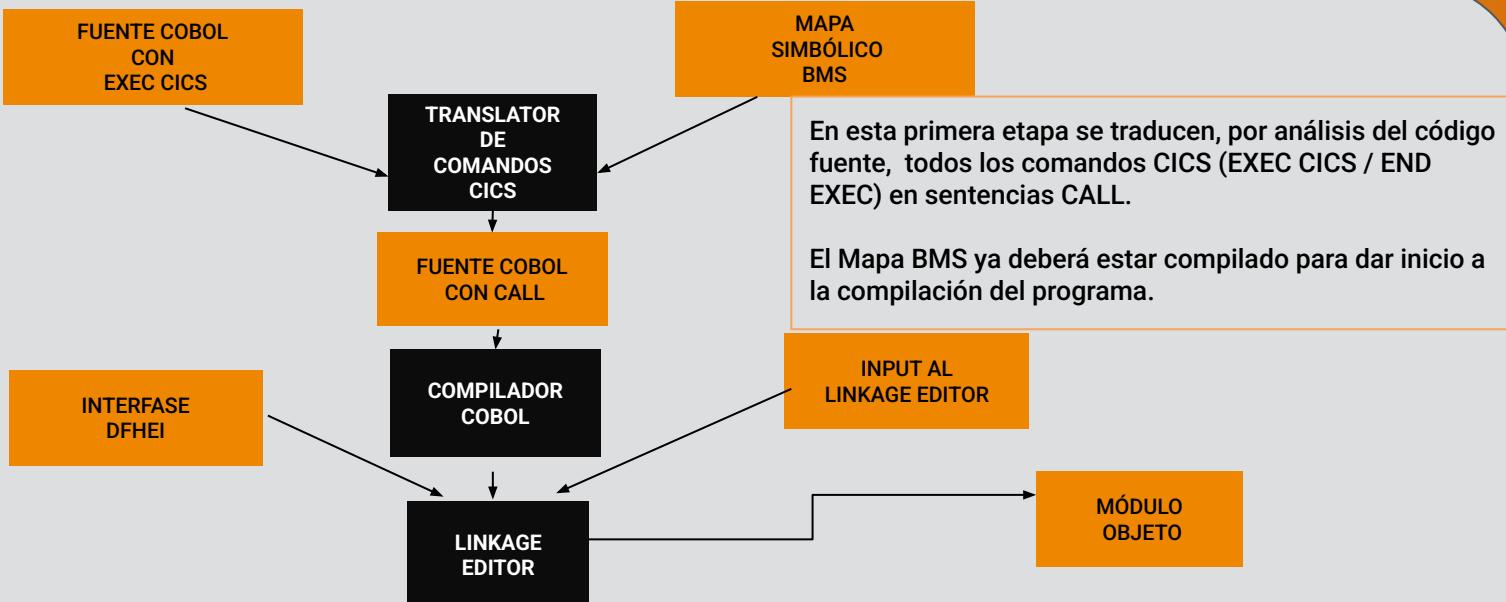
Ejemplo de codificación de PROCEDURE DIVISION con testeo de recepción de commarea y de un mapa. Se puede verificar la condición de MAPFILE con el comando HANDLE CONDITION para verificar si es primera vez o no.

```
*****
PROCEDURE DIVISION.
*****  
  
IF EIBCALEN > 0  
  MOVE 'S'      TO WS-HAY-COMMAREA  
  MOVE DFHCOMMAREA TO COM-COMMAREA  
ELSE  
  MOVE ''      TO WS-HAY-COMMAREA  
  INITIALIZE    COM-COMMAREA  
END-IF.  
  
EXEC CICS  
  RECEIVE MAP  ('MLTR 001')  
    MAPSET ('MLTR 001')  
    RESP   (WS-RESP)  
END-EXEC.  
  
MOVE 2          TO C20-OPTION.  
MOVE LENGTH OF COMMAREA-DSPT020 TO COM-LEN.  
  
EXEC CICS LINK PROGRAM ('DSPT020')  
  COMMAREA (COMMAREA-DSPT020)  
  LENGTH (COM-LEN)  
  RESP   (WS-RESP)  
END-EXEC.
```



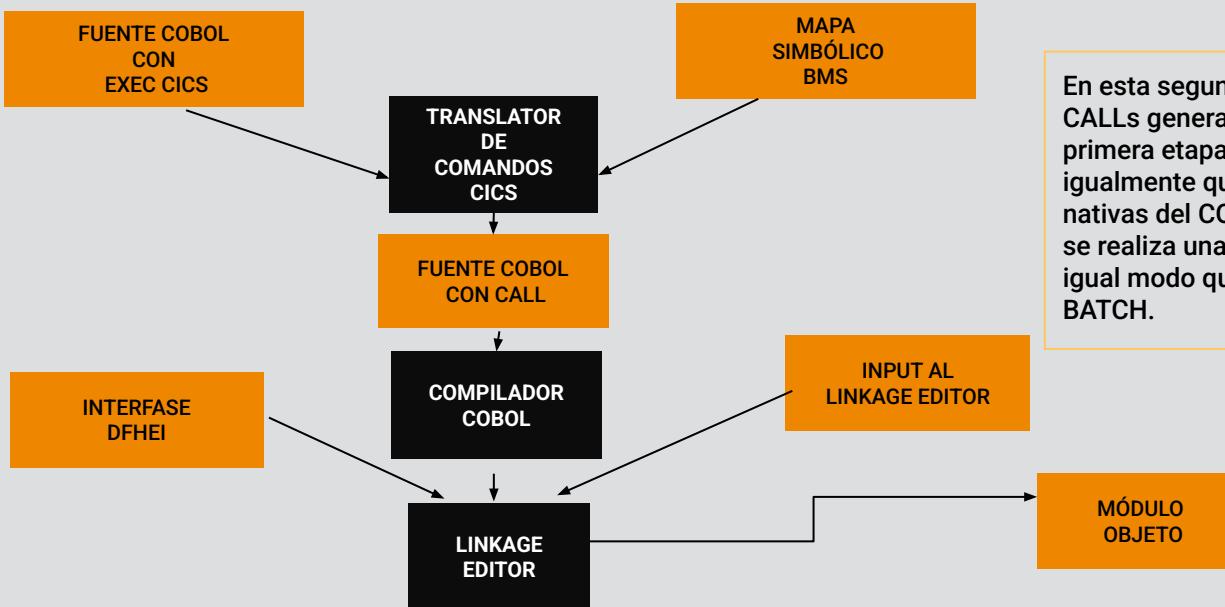


PROCESO DE COMPILEACIÓN





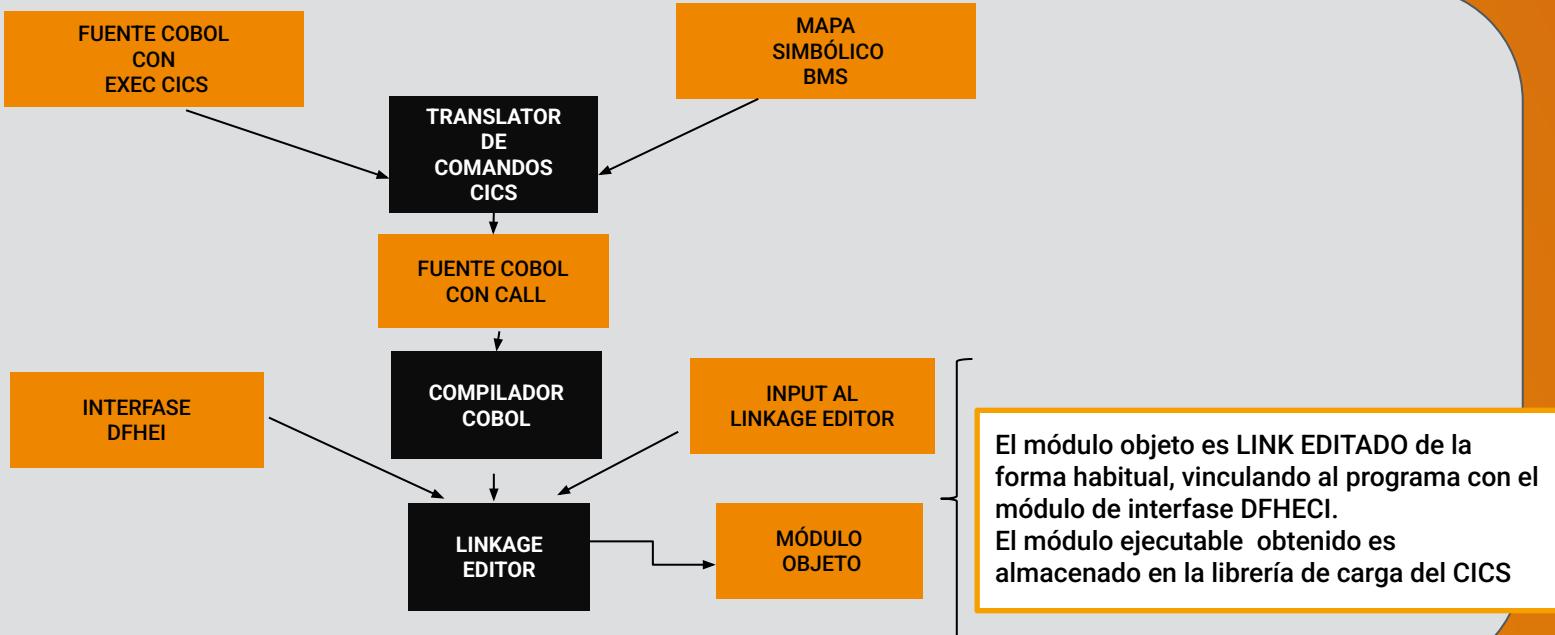
PROCESO DE COMPILEACIÓN



En esta segunda etapa los CALLs generados por la primera etapa son tratados igualmente que las sentencias nativas del COBOL, por lo que se realiza una compilación de igual modo que un fuente BATCH.



PROCESO DE COMPILEACIÓN





02

ALTA DEL PROGRAMA CICS

...

Una vez compilado nuestro programa, deberemos darlo de alta en el CICS.

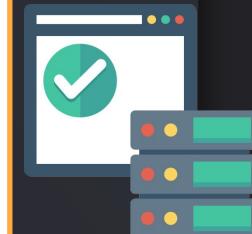
Para ello, nos valdremos de la aplicación **CEDA**, que nos permitirá dar de alta el programa, el mapa, archivos nuevos que requiere nuestra aplicación, etc.

Al **CEDA** solo pueden ingresar usuarios avanzados, por lo que en la mayoría de las Instalaciones se autoriza su uso a administradores de CICS, con la transacción **CEDC** se puede consultar el contenido de los grupos definidos en el CSD.

Programa	Lenguaje	Transacción	Mapa
DLTR001	cobol	DLT1	MLTR001
DLTR002	cobol	DLT2	MLTR002

Si por el contrario, ya está declarado nuestro programa en el CICS pero lo hemos modificado por alguna razón, deberemos hacer un refresh del módulo de carga para que el CICS tome la nueva versión del programa con la transacción **CEMT**.

IMPORTANTE: Las versiones de los programas permanecerán congeladas para el CICS hasta que se levante de nuevo el CICS o hasta que se le indique NEW COPY con la transacción **CEMT**.
(CEMT SET PROGRAM(DL TR001) NEWS)





... Mapa Conceptual

01
CONCEPTOS Y
FACILIDADES

02
Preparación
de un
programa

03
Generación de
mapas - BMS

04
Control de
programas





03

...

GENERACIÓN DE MAPAS - BMS

- Codificación de fuente Basic Mapping Support
- Proceso de compilación
- Alta del Mapa en el CICS
- NEW COPY



03

...

BMS (Basic Mapping Support)

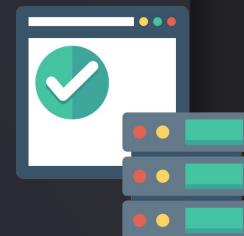
El **BMS** es una interfase entre el TCP y nuestros programas de aplicación. Nos permite aislarnos de las características de una terminal a la hora de enviar o recibir la información.

En la operación de enviar datos desde el programa hacia la terminal (de SALIDA), el BMS toma los datos del área de trabajo del programa y los sitúa en las posiciones correctas de la pantalla y le intercala los caracteres de control dependientes del hardware necesarios.

En la Operación de recibir datos provenientes de una terminal (de ENTRADA), elimina los caracteres de control dependientes del hardware y distribuye los datos que vienen en posiciones específicas del mapa en el área de trabajo del programa.

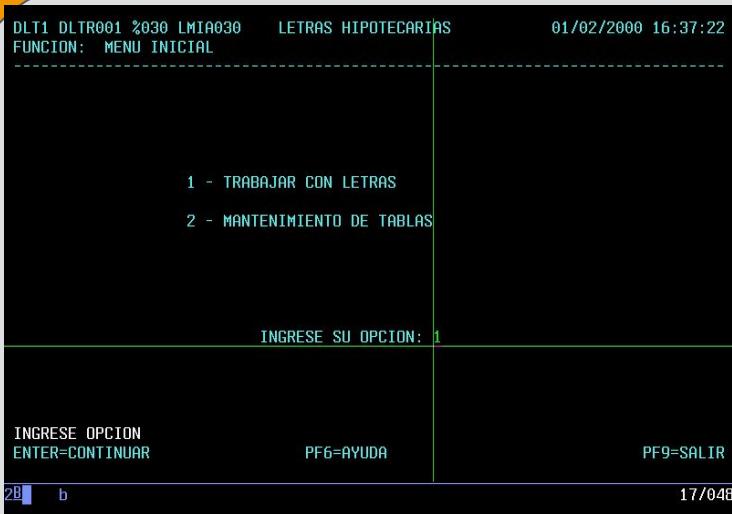
Esta interfaz nos brinda una total **independencia de los dispositivos** de I/O ya que nos permite escribir programas sin ocuparnos de las características físicas de las terminales utilizadas por nuestra aplicación.

Por otra parte, nos brinda **independencia de formatos**, ya que nos olvidamos al momento de la codificación del programa de situar los campos de datos en posiciones específicas. El BMS se encarga de asociar el nombre de campo con la posición en la pantalla.





BMS (Basic Mapping Support)



VISUALIZACIÓN DEL
MAPA FÍSICO
(TERMINAL)

01 MLTR001.
02 FILLER PIC X(12).
02 TRAN1L COMP PIC S9(4).
02 TRAN1F PICTURE X.
02 FILLER REDEFINES TRAN1F.
03 TRAN1A PICTURE X.
02 TRAN1I PIC X(4).
02 TERML COMP PIC S9(4).
02 TERMF PICTURE X.
02 FILLER REDEFINES TERMF.
03 TERMA PICTURE X.
02 TERMI PIC X(4).

MAPA
SIMBÓLICO
(WORKING)

EXEC CICS
SEND MAP ('MLTR001')
MAPSET ('MLTR001')
ERASE
CURSOR (WS-CURSOR)
END-EXEC.



03

BMS (Basic Mapping Support)

MAPA FÍSICO:

Describe el Formato de la representación de un tipo de terminal dada. Adopta la forma de una tabla que contiene la siguiente información:

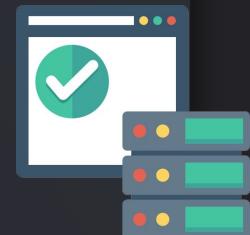
- Longitud y ubicación de los campos de datos
- Atributos de los campos de datos (Protegidos, brillantes, dark, etc.)
- Constantes (encabezamiento, descripciones)
- Características de las terminales

MAPA SIMBÓLICO:

Define los campos de datos a ser accedidos por el programa de aplicación. El mapa simbólico se copia en la memoria de trabajo del programa.

Cada campo se descompone en subcampos a los que se adiciona un sufijo. Veamos qué sucede entonces con el campo NOMBRE:

- **NOMBRE** Número de caracteres tipeados en NOMBRE (-1 posiciona el cursor)
- **NOMBREA** Byte de atributo con las características de NOMBRE (Brillante)
- **NOMBREL** Byte de atributo que indica el largo en bytes de la variable NOMBRE
- **NOMBREF** Flag Byte. Generalmente en X'00'. Si NOMBRE fue borrado asume X'80'
- **NOMBREI** Campo de Input. Si no se ingresó NOMBRE contendrá Low-Values.
- **NOMBREO** Campo de Output. Si contiene Low-Values no se transmite a la terminal.





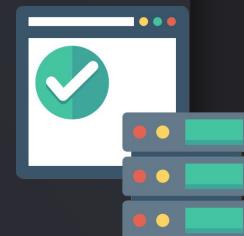
03

...

Código fuente BMS

EDITANDO EL FUENTE DE UN MAPA BMS

```
File Edit Confirm Menu Utilities Compilers Test Help
EDIT      AEND.ADMINPRD.SOURCE(MLTR001) - 01.00          Columns 00001 00072
Command ==> _____ Scroll ==> CSR
***** **** Top of Data ****
=COLS> ----+---1---+---2---+---3---+---4---+---5---+---6---+---7---
000001 MLTR001 DFHMSD TYPE=&SYSPARM,
000002           CTRL=(FRSET,FREEKB),
000003           TIOAPFX=YES,
000004           MODE=INOUT,
000005           LANG=COBOL,
000006           STORAGE=AUTO
000007 MLTR001 DFHMDFI SIZE=(24,80)
000008 *
000009 TRAN1   DFHMDF INITIAL='DLT1',
000010           POS=(1,1),
000011           LENGTH=4,
000012           ATTRB=(PROT,NORM,FSET)
000013 *
000014       DFHMDF INITIAL='DLTR001',
000015           POS=(1,6),
000016           LENGTH=7,
000017           ATTRB=(ASKIP,NORM)
2B a          08/002
```





03

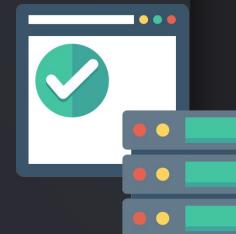
Código fuente BMS

EDITANDO EL FUENTE DE UN MAPA BMS

```
File Edit Configuration Utilities (MLTR001)
***** Top ***** Data *****
COLUMNA 1
SÍMBOLO
** INDICA REM
CÓDIGO DE
OPERACIÓN
DEJAR AL MENOS
1 BLANCO
000001 MLTR001 DFHMSD TYPE=&SYSPARM,
000002           CTRL=(FSET,FREEKB),
000003           TIOAPFX=YES,
000004           MODE=INOUT,
000005           LANG=COBOL,
000006           STORAGE=AUTO
000007 MLTR001 DFHMDI SIZE=(24,80)
000008 *
000009 TRAN1   DFHMDF INITIAL='DLT1',
000010           POS=(1,1),
000011           LENGTH=4,
000012           ATTRB=(PROT,NORM,FSET)
000013 *
000014           DFHMDF INITIAL='DLTR001',
000015           POS=(1,6),
000016           LENGTH=7,
000017           ATTRB=(ASKIP,NORM)

PARÁMETROS
Separados por comas. Un blanco
indica fin de parámetros
COLUMNA 72
'C' Carácter de
Continuación
COLUMN 16
Parámetros Adicionales.
La línea a ser continuada
debe terminar con coma y
un carácter de
continuación la columna
72.
```

28 a 08/002



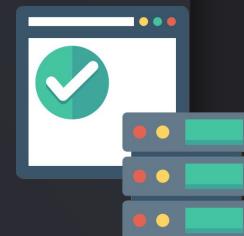


03

Código fuente BMS

EDITANDO EL FUENTE DE UN MAPA BMS

```
Utilities Compilers Test Help
Columns 00001 00072
Scroll ==> CSR
*****
***** Top of Data *****
=COLS> 1-----2-----3-----4-----5-----6-----7-----
00001 MLTR001 DFHMSD TYPE=&SYSPARM,
          CTRL=(FRSET,FREEKB),
          TIOAPFX=YES,
          MODE=INOUT,
          LANG=COBOL,
          STORAGE=AUTO
00006 0007 MLTR001 DFHMDI SIZE=(24,80)
00008 *
00009 TRAN1   DFHMDF INITIAL='DLT1',
00010           POS=(1,1),
00011           LENGTH=4,
00012           ATTRB=(PROT,NORM,FSET)
00013 *
00014   DFHMDF INITIAL='DLTR001',
00015   POS=(1,6),
00016   LENGTH=7,
00017   ATTRB=(ASKIP,NORM)
```





03

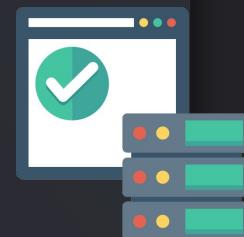
MACROS BMS

DFHMSD: (DEFINE UN CONJUNTO DE MAPAS – MAPSET)

```
Mapset DFHMSD TYPE={DSECT:MAP:&SYSPARM},  
        CTRL={[FRSET],[FREEKB],[PRINT],[ALARM]},  
        TIOAPFX=YES,  
        MODE={IN:INOUT:OUT},  
        LANG={COBOL:PLI:ASM:RPG},  
        STORAGE={AUTO:BASE=name}
```

Por supuesto, aquí vemos solo una pequeña visión de las opciones que soporta esta macro de CICS.

- **Mapset** Nombre del MAPSET (1 a 7 caracteres)
- Todo mapa BMS debe comenzar con **DFHMSD TYPE={DSECT:MAP:&SYSPARM}** y debe terminar con **DFHMSD TYPE=FINAL**.
- **DSECT** se utiliza para crear un conjunto de mapas simbólicos
- **MAP** se utiliza para crear un conjunto de mapas físicos
- **FINAL** Indica la finalización de la definición del MAPSET o conjunto de mapas.





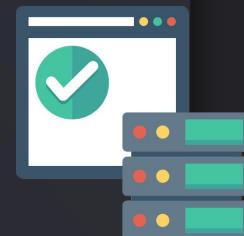
03

MACROS BMS

DFHMSD: (DEFINE UN CONJUNTO DE MAPAS – MAPSET)

```
Mapset DFHMSD TYPE={DSECT:MAP:FINAL:&SYSPARM},  
    CTRL={[FRSET],[FREEKB],[ALARM]},  
    TIOAPFX=YES,  
    MODE={IN:INOUT:OUT},  
    LANG={COBOL:PLI:ASM:RPG},  
    STORAGE={AUTO:BASE=name}
```

- **CTRL** Define las características de la terminal
- **FRSET** Setea en OFF los MDT (modified data tag) de los campos enviados a la terminal salvo aquellos que el programa setea en ON. Recordemos que solo los campos cuyo MDT esté en ON viajan desde la terminal a la aplicación.
- **FREEKB** Desbloquea el teclado. De otra forma, el operador deberá pulsar RESET antes de iniciar la carga de datos.
- **ALARM** Provoca la emisión de un pitido en la terminal al desplegarse cualquier mapa del MAPSET.





03

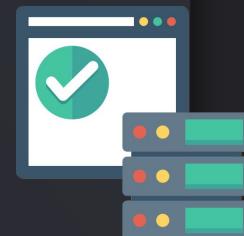
MACROS BMS

DFHMSD: (DEFINE UN CONJUNTO DE MAPAS – MAPSET)

```
Mapset DFHMSD TYPE={DSECT:MAP:FINAL:&SYSPARM},  
CTRL=([FRSET],[FREEKB],[ALARM]),  
TIOAPFX=YES,  
MODE={IN:INOUT:OUT},  
LANG={COBOL:PLI:ASM:RPG},  
STORAGE={AUTO:BASE=name}
```

- **TIOAPFX=YES** Se usa este parámetro si TYPE=DSEC. El BMS inserta 12 Bytes de control al comienzo del mapa simbólico. Para nuestros programas que son a nivel de comandos siempre debe especificarse esta opción.
- **MODE** Indica el objetivo del mapa
- **IN** El mapa será utilizado solo para entrada (input)
- **INOUT** El mapa será tanto de entrada (input) como de salida (output)
- **OUT** El mapa será solo de salida (output)

Se recomienda siempre utilizar INOUT salvo fines específicos. El copy emergente será acorde al valor aquí indicado (contendrá las definiciones de input y output, solo de input o solo de output).





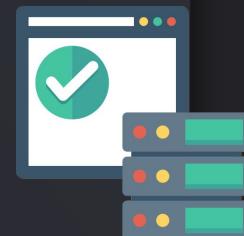
03

MACROS BMS

DFHMSD: (DEFINE UN CONJUNTO DE MAPAS – MAPSET)

```
Mapset DFHMSD TYPE={DSECT:MAP:FINAL:&SYSPARM},  
          CTRL={[FRSET],[FREEKB],[ALARM]},  
          TIOAPFX=YES,  
          MODE={IN:INOUT:OUT},  
          LANG={COBOL:PLI:ASM:RPG},  
          STORAGE={AUTO:BASE=name}
```

- **LANG** Indica el lenguaje en que estará escrito el programa que lo va a tratar. De acuerdo a este parámetro se genera el Copy en el lenguaje indicado al momento de la compilación del MAPSET por el BMS.
- **COBOL** El programa estará codificado en COBOL.
- **PLI** El programa estará codificado en PL/I
- **ASM** El programa estará codificado en Assembler
- **RPG** El programa estará codificado en RPG
- **STORAGE** Indica el almacenamiento necesario para la carga del MAPSET
- **AUTO** El CICS adquirirá un área de memoria separada para cada mapa
- **BASE=name** El MAPSET debe ser copiado en la LINKAGE SECTION. También se deberá ejecutar la sentencia GETMAIN para adquirir suficiente almacenamiento principal para contener el mapa más grande del MAPSET.





03

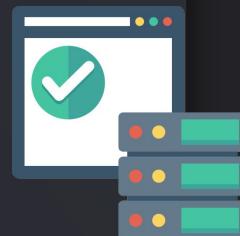
MACROS BMS

...

DFHMDI: (DEFINE UN MAPA DENTRO DEL MAPSET)

[mapname] DFHMDI [SIZE=(line,column)]

- **Mapname** Nombre del MAPA dentro del MAPSET (1 a 7 caracteres)
- **SIZE** Indica el tamaño de la pantalla expresado en cantidad de líneas y de columnas. Generalmente, en terminales del Host, los valores a indicar son (24,80)





03

MACROS BMS

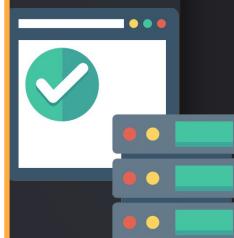
...

DFHMDF: (DEFINE UN CAMPO DENTRO DE UN MAPA)

```
[LENGTH=number],  
[JUSTIFY={({LEFT:RIGHT}) [, {BLANK:ZERO})}],  
[INITIAL='literal character'],  
[XINIT='literal hexadecimal'],  
[ATTRB={({ASKIP:PROT:UNPROT[,NUM])}),  
    ({BRT:NORM:DRK} [,DET] [,IC] [,FSET])}],  
[GRPNAMEx=group-name],  
[OCCURS=number],  
[PICIN='value'],  
[PICOUT='value']
```

ALGUNOS OPERANDOS

- **Fname** Nombre del campo (1 a 7 caracteres). Si es omitido desde el programa no podremos asignarle valores ni cambiar sus atributos. Se utiliza para títulos o descripciones dentro del mapa a las que no es necesario alterar su contenido.
- **POS** Indica la posición del campo en la pantalla
- **Number** indica la posición relativa desde el valor 0 (cero)
- **Line** indica la línea donde se ubicará el campo (la primera es la 1. Máximo 240)
- **Column** Indica la columna dentro de la línea (la primera es la 1. Máx 240). Tener cuidado ya que indica la posición del byte de atributo que precede al campo y no la primera posición del campo.





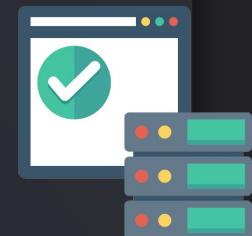
03

MACROS BMS

DFHMDI: (DEFINE UN MAPA DENTRO DEL MAPSET)

```
[mapname] DFHMDI [SIZE=(line,column)]
[LENGTH=number],
[JUSTIFY={({LEFT:RIGHT}) [{BLANK:ZERO}]},
[INITIAL='literal character'],
[XINIT='literal hexadecimal'],
[ATTRB={({ASKIP:PROT:UNPROT,NUM})},
  {{BRT:NORM:DRK} [{DET} [{IC} [{FSET}]}}},
[GRPNNAME=group-name],
[OCCURS=number],
[PICIN='value'],
[PICOUT='value']
```

- **LENGTH=number** Indica la longitud del campo (Excluyendo el byte de atributo). No debe exceder el tamaño de la línea. Puede omitirse este parámetro si PICIN o PICOUT están presentes.
- **JUSTIFY** Permite el relleno automático con blancos o ceros de campos parcialmente ingresados. Los valores default son:
 - LEFT, BLANK para campos alfanuméricos (si no se especificó NUM)
 - RIGHT, ZERO para campos numéricos (si se especificó NUM)
- **INITIAL** Asigna la 'constante carácter' al campo en tiempo de salida. Para asumir el valor inicial desde nuestro programa debe ser inicializado con LOW-VALUES, de lo contrario contendrá el valor asignado por el programa.





03

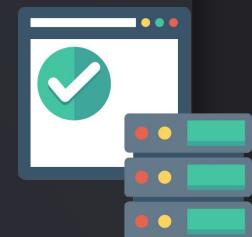
MACROS BMS

DFHMDI: (DEFINE UN MAPA DENTRO DEL MAPSET)

[mapname] DFHMDI [SIZE=(line,column)]

[LENGTH=number],
[JUSTIFY={({LEFT:RIGHT})} [{BLANK:ZERO}]],
[INITIAL='literal character'],
[XINIT='literal hexadecimal'],
[ATTRB={({ASKIP:PROT:UNPROT,NUM})}],
 [{{BRT:NORM:DRK} [,DET] [,IC] [,FSET]}],
[GRPNNAME=group-name],
[OCCURS=number],
[PICIN='value'],
[PICOUT='value']

- **XINIT** Igual que INITIAL pero la constante está en valor HEXADECIMAL.
- **ATTRB** Permite la especificación de variados valores de atributos. Si se omite el default es ASKIP y NORM; si no se omite, además de los parámetros especificados asume NORM y UNPROT, salvo que se indique lo contrario.
- **ASKIP** El cursor salta al campo en forma automática.
- **PROT** No se pueden asignar valores al campo (está protegido).
- **UNPROT** Se permite el ingreso de datos (está desprotegido).
- **NUM** Solo se permite el ingreso de datos numéricos, salvo que el operador pulse la tecla de 'mayúsculas'.





03

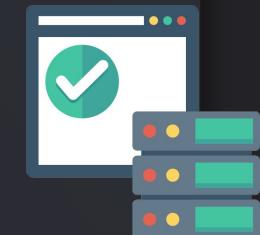
MACROS BMS

DFHMDI: (DEFINE UN MAPA DENTRO DEL MAPSET)

[mapname] DFHMDI [SIZE=(line,column)]

[LENGTH=number],
[JUSTIFY={({LEFT:RIGHT})} [{BLANK:ZERO}]],
[INITIAL='literal character'],
[XINIT='literal hexadecimal'],
[ATTRB={({ASKIP:PROT:UNPROT[,NUM]}},
 [{{BRT:NORM:DRK} [DET] [IC] [,FSET]}]),
[GRPNNAME=group-name],
[OCCURS=number],
[PICIN='value'],
[PICOUT='value']

- **BRT** El campo será visualizado con BRILLO.
- **NORM** El campo será visualizado a una intensidad normal.
- **DRK** El campo no será visualizado en la pantalla (DARK).
- **DET** Permite la utilización de lápiz óptico.
- **IC** El cursor se posicionará inicialmente en este campo. No deberá especificarse esta opción en otro campo.
- **FSET** El MDT de este campo siempre estará en ON. Se utiliza cuando el valor de un campo protegido debe retornar a nuestro programa o, si el valor enviado en el campo debe retornar en nueva transacción.





03

MACROS BMS

DFHMDI: (DEFINE UN MAPA DENTRO DEL MAPSET)

[mapname] DFHMDI [SIZE=(line,column)]

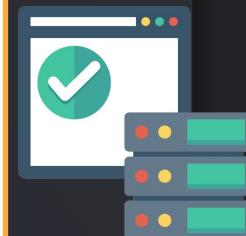
```
[LENGTH=number],  
[JUSTIFY={({LEFT:RIGHT})} [{BLANK:ZERO})]],  
[INITIAL='literal character'],  
[XINIT='literal hexadecimal'],  
[ATTRB={({ASKIP:PROT:UNPROT[NUM])}},{  
    [{BRT:NORM:DRK} [DET] [IC] [FSET])}],  
[GRPNAME=group-name],
```

GRPNAME Permite definir varios campos bajo un mismo campo grupal para que el programa pueda tratarlos simultáneamente.

```
MO  DFHMDF POS=(10,1),LENGTH=2,ATTRB=BRT,GRPNAME=DATE  
SEP1 DFHMDF POS=(10,3),LENGTH=1,GRPNAME=DATE,INITIAL='-'  
DAY DFHMDF POS=(10,4),LENGTH=2,GRPNAME=DATE  
SEP2 DFHMDF POS=(10,6),LENGTH=1,GRPNAME=DATE,INITIAL='-'  
YR  DFHMDF POS=(10,7),LENGTH=2,GRPNAME=DATE
```

02 DATE.

- 03 FILLER PICTURE X(2).
- 03 MOA PICTURE X.
- 03 MOO PIC X(2).
- 03 SEP1 PIC X(1).
- 03 DAO PIC X(2).
- 03 SEP2 PIC X(1).
- 03 YRO PIC X(2).





03

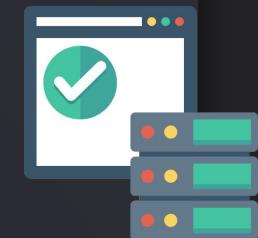
MACROS BMS

DFHMDI: (DEFINE UN MAPA DENTRO DEL MAPSET)

[mapname] DFHMDI [SIZE=(line,column)]

[LENGTH=number],
[JUSTIFY={({LEFT:RIGHT})} [{BLANK:ZERO}]],
[INITIAL='literal character'],
[XINIT='literal hexadecimal'],
[ATTRB={({ASKIP:PROT:UNPROT[NUM]}},
 [{BRT:NORM:DRK} [DET] [IC] [,FSET]}]),
[GRPNNAME=group-name],
[OCCURS=number],
[PICIN='value'],
[PICOUT='value']

- **OCCURS** El campo será repetido 'number' veces en los mapas físico y simbólico. En caso de COBOL el mapa simbólico (copy) contendrá la sentencia OCCURS. OCCURS y GRPNNAME son excluyentes.
- **PICIN** Indica que una cláusula PICTURE será aplicada al Copy de Cobol en la versión input del mapa simbólico. PICIN='999v99' indica a nuestro programa que debe asumir 2 decimales. Si PICIN no es especificado, en el copy Cobol se definirá como 'carácter' aún si se especifica NUM como atributo del campo en el mapa.
- **PICOUT** Genera una cláusula PICTURE en el Copy Cobol que será usada en la versión output del mapa simbólico.
PICOUT='\$\$\$\$\$,99' displayará el valor '\$123,50' .





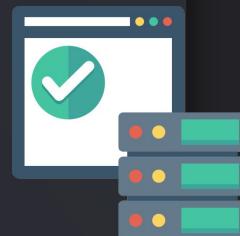
...

03

ALTA DEL MAPA EN EL CICS

Una vez compilado nuestro mapa (que el CICS lo trata como programa) deberemos darlo de alta en el CICS con la transacción **CEDA**, que nos permitirá dar el mapa. Si por el contrario, ya está declarado nuestro mapa en el CICS pero lo hemos modificado por alguna razón, deberemos hacer un “new copy” del módulo de carga para que el CICS tome la nueva versión del mapa con la transacción **CEMT**.

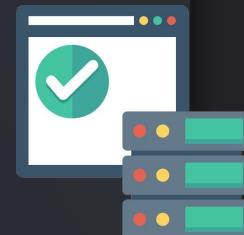
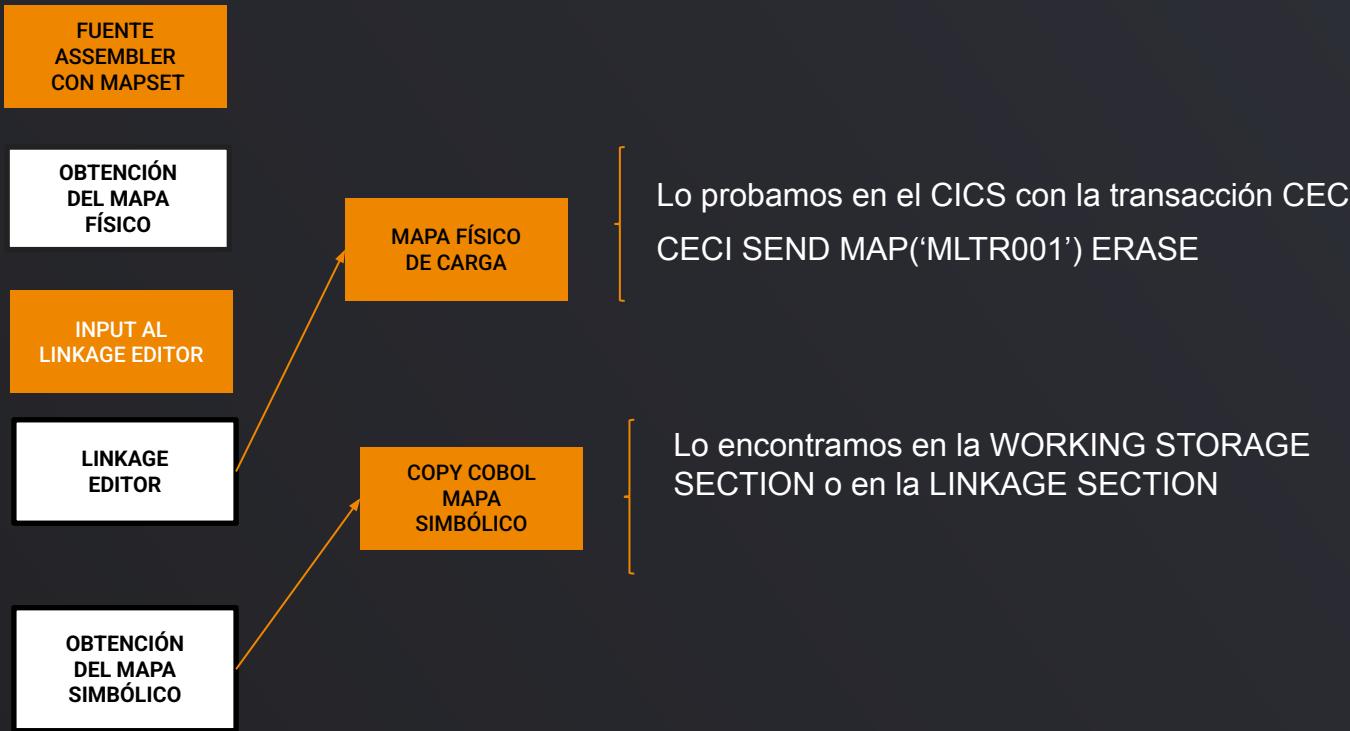
IMPORTANTE: Las versiones de los mapas permanecerán congeladas para el CICS hasta que se levante de nuevo el CICS o hasta que se le indique NEW COPY con la transacción **CEMT. (CEMT SET PROGRAM(MLTR001) NEWC)**





03

PROCESO COMPILACIÓN BMS





... Mapa Conceptual

01
CONCEPTOS Y
FACILIDADES

02
Preparación
de un
programa

03
Generación de
mapas - BMS

04
Control de
programas





04

...

CONTROL DE PROGRAMAS

- COMMAREA
- LINK
- XCTL
- RETURN
- START/RETRIEVE
- EIB – EXEC INTERFACE BLOCK



04

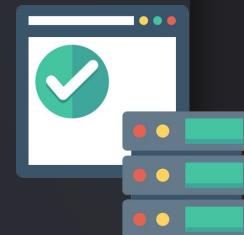
COMMAREA

La **COMMAREA** o **DFHCOMMAREA** es un área de comunicación entre programas. Deberemos incorporar entonces el concepto de programa emisor o generador de la COMMAREA y el programa receptor de la COMMAREA armada por su predecesor.

Una COMMAREA solo será accedida por aquellos programas a los que explícitamente se les pase. El programa emisor deberá tener definida una 'data-area' en la WORKING o en la LINKAGE, que será referenciada al momento de entregar el control a otro programa.

El programa receptor, obligatoriamente, deberá tener definida una 'data-area' de igual tamaño (o mayor) en la LINKAGE y de igual estructura en la WORKING a fin de garantizar la correspondencia de los datos transferidos. No resulta necesario ejecutar ningún comando de CICS para que el área de la LINKAGE, asignada en el programa receptor a la COMMAREA, sea completada con los datos de la transferencia.

Para que el programa receptor esté enterado si le llegó o no una COMMAREA, deberá testear el campo EIBCALEN de la EIB (EXEC INTERFASE BLOCK del CICS). Si este campo es mayor que CERO, nos indica la longitud de la COMMAREA recibida (que deberá corresponder con la longitud esperada). Si es igual a CERO, no se ha recibido ningún dato en la COMMAREA.





04

COMMAREA

WORKING STORAGE SECTION

```
77 COM-LEN      PIC S9(4) COMP.  
01 COMMAREA-DSPT020.  
  03 PARM1      PIC X(03).  
  03 PARM2      PIC X(03).
```

PROCEDURE DIVISION.

```
MOVE LENGTH OF COMMAREA-DSPT020 TO COM-LEN.  
EXEC CICS LINK PROGRAM ('DSPT020')  
    COMMAREA (COMMAREA-DSPT020)  
    LENGTH (COM-LEN)  
    RESP   (WS-RESP)  
END-EXEC.
```

WORKING STORAGE SECTION.

```
77 COM-LEN      PIC S9(4) COMP.  
01 COMMAREA-DSPT020.  
  03 PARM1      PIC X(03).  
  03 PARM2      PIC X(03).
```

LINKAGE SECTION.

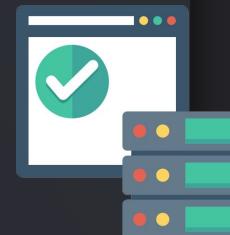
```
01 DFHCOMMAREA  PIC X(06).
```

PROCEDURE DIVISION.

```
IF EIBCALEN > 0  
  MOVE DFHCOMMAREA TO COMMAREA-DSPT020  
END-IF.
```

PROGRAMA
EMISOR

PROGRAMA
RECEPTOR





04

...

COMANDO LINK

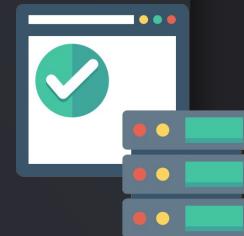
```
EXEC CICS LINK  
    PROGRAM ('name')  
    [COMMAREA ('data-area')  
     LENGTH   ('data-value')]  
END-EXEC.
```

Este comando es utilizado para disparar sincrónicamente un programa de nivel lógico inferior, retornado a la próxima instrucción del programa llamante.

Como ya sabemos, si el programa invocado no está residente en memoria, será cargado previo a su ejecución. El retorno al programa de mayor nivel se produce cuando en el programa llamado se ejecuta el comando RETURN. Entonces, el control es devuelto al programa principal en la siguiente instrucción al LINK.

Durante la ejecución del comando LINK, los recursos del programa principal permanecen en memoria mientras el programa invocado se está ejecutando.

La utilización del comando LINK se recomienda en casos de tener tareas repetitivas y utilizadas por numerosos programas de aplicación, tal el caso de una rutina de control de seguridad o de cálculo de un dígito verificador, pero debe evitarse en lo general la utilización del comando LINK.





04

COMANDO LINK

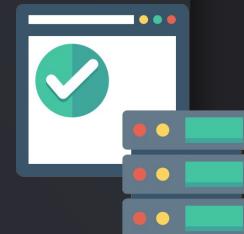
PROGRAMA PRINCIPAL

```
MOVE LENGTH OF COMMAREA-DSPT020 TO COM-LEN.  
EXEC CICS LINK PROGRAM ('DSPT020')  
    COMMAREA (COMMAREA-DSPT020)  
    LENGTH (COM-LEN)  
END-EXEC.
```

PROGRAMA INVOCADO

```
IF EIBCALEN > 0  
    MOVE '1' TO COM-NIVEL-SEGURIDAD  
ELSE  
    MOVE '0' TO COM-NIVEL-SEGURIDAD  
END-IF.  
EXEC CICS  
    RETURN  
END-EXEC.
```

```
EVALUATE COM-NIVEL-SEGURIDAD  
    WHEN '1'  
        CONTINUE  
    WHEN OTHER  
        PERFORM NIVEL-SEGURIDAD-INSUF  
        THRU F- NIVEL-SEGURIDAD-INSUF  
END-EVALUATE.
```





...

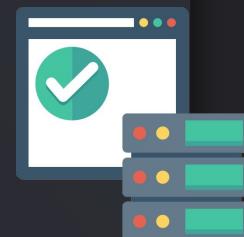
04

COMANDO XCTL

```
EXEC CICS XCTL  
    PROGRAM ('name')  
    [COMMAREA ('data-area')  
     LENGTH   ('data-value')]  
END-EXEC.
```

Este comando es utilizado para disparar asincrónicamente una transacción, no hay retorno al programa que emite el XCTL.

Al ejecutarse este comando, el programa invocado comienza su ejecución inmediatamente, mientras que el CICS libera de la memoria principal al programa llamador.





04

COMANDO XCTL

PROGRAMA PRINCIPAL

```
MOVE LENGTH OF COMMAREA-DSPT020 TO COM-LEN.  
EXEC CICS XCTL PROGRAM ('DSPT020')  
    COMMAREA (COMMAREA-DSPT020)  
    LENGTH (COM-LEN)  
END-EXEC.
```

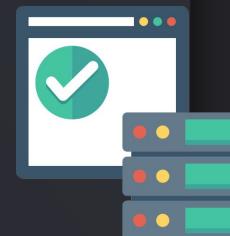


PROGRAMA INVOCADO

```
IF EIBCALEN > 0  
    MOVE '1' TO COM-NIVEL-SEGURIDAD  
ELSE  
    MOVE '0' TO COM-NIVEL-SEGURIDAD  
END-IF.  
EXEC CICS  
    RETURN  
END-EXEC.
```

```
EVALUATE COM-NIVEL-SEGURIDAD  
WHEN '1'  
    CONTINUE  
WHEN OTHER  
    PERFORM NIVEL-SEGURIDAD-INSUF  
        THRU F- NIVEL-SEGURIDAD-INSUF  
END-EVALUATE.
```

CICS





04

...

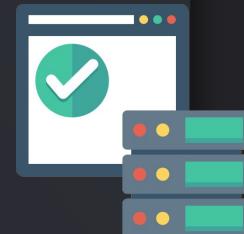
COMANDO RETURN

```
EXEC CICS RETURN  
  [TRANSID  ('name') ]  
  [COMMAREA ('data-area')]  
  LENGTH    ('data-value') ]  
END-EXEC.
```

Este comando es utilizado para TERMINAR una tarea. Provoca la liberación de todos los recursos y memoria asociados a esa TASK o tarea.

Devuelve el control a un programa de aplicación de nivel más alto o al CICS.

TRANSID Si se especifica esta opción, se le indica al CICS que inicie la transacción indicada en '**name**' cuando haya una respuesta del usuario operador en la terminal (por ejemplo, ENTER, alguna PF, ATTN). Esta opción prevalece sobre cualquier especificación de transacción del usuario sobre la pantalla.





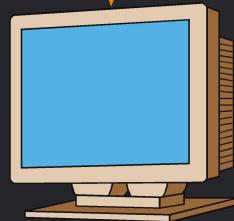
04

...

COMANDO RETURN

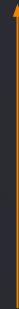
TRANSACCIÓN AAAA

```
MOVE LENGTH OF COMMAREA-BBBB TO COM-LEN.  
EXEC CICS RETURN  
    TRANSID ('BBBB')  
    COMMAREA (COMMAREA-BBBB)  
    LENGTH (COM-LEN)  
END-EXEC.
```



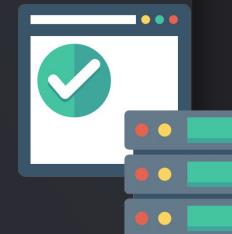
ENTER
PF1 – PF24
ATTN

CICS



TRANSACCIÓN BBBB

```
IF EIBCALEN > 0  
    MOVE '1' TO COM-NIVEL-SEGURIDAD  
ELSE  
    MOVE '0' TO COM-NIVEL-SEGURIDAD  
END-IF.  
EXEC CICS  
    RETURN  
END-EXEC.
```





04

...

COMANDO START

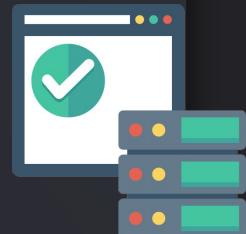
```
EXEC CICS START  
    TRANSID    ('name')  
    [INTERVAL   (hhmmss) : TIME(hhmmss)]  
    [FROM       ('data-area')]  
    LENGTH     ('data-value') ]  
    [TERMID     ('name')]  
    [REQID      ('name')]  
END-EXEC.
```

Este comando permite especificar que otra transacción sea invocada (**TRANSID**), en forma inmediata, o temporizada a una determinada hora del día, o luego de transcurrido determinado lapso de tiempo (**INTERVAL**).

Permite enviar datos a la nueva tarea (**FROM**) y en qué terminal deberá iniciarse (**TERMID**).

También permite identificar el comando para una posterior cancelación de la TASK iniciada o pendiente de ejecución (**REQID 'name'** donde 'name' tiene longitud de 1 a 8 caracteres).

Se utiliza generalmente el comando START para tareas de IMPRESIÓN.



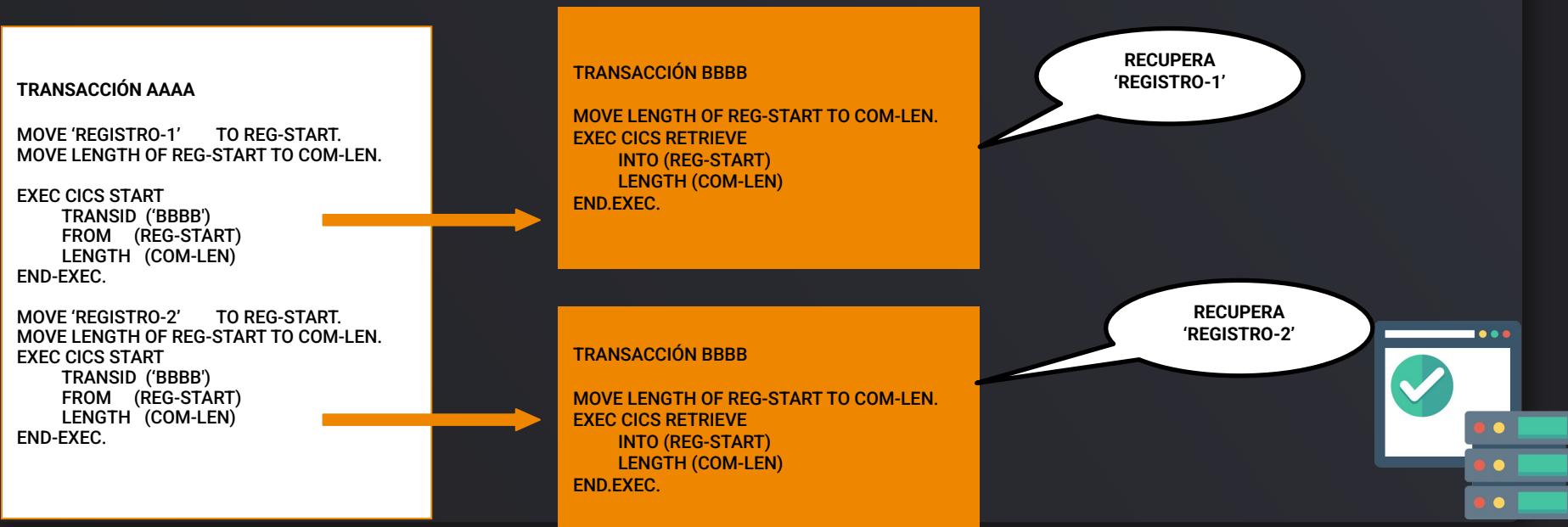


04

COMANDO START

Si la nueva tarea **NO** está asociada con una terminal, cada comando **START** emitido desde el programa llamador genera una tarea separada para iniciarse.

Cada tarea contendrá, como consecuencia, al menos un registro de datos a recuperar.





04

COMANDO START

Si la nueva tarea **está asociada con una terminal**, cada comando **START** emitido desde el programa llamador genera una única tarea hacia esa terminal.

Esta tarea contendrá, como consecuencia, la acumulación de tantos registros de datos a recuperar como comandos **START** se ejecutaron.

TRANSACCIÓN AAAA

```
MOVE 'REGISTRO-1' TO REG-START.  
MOVE LENGTH OF REG-START TO COM-LEN.
```

```
EXEC CICS START  
    TRANSID ('BBBB')  
    FROM (REG-START)  
    LENGTH (COM-LEN)  
END-EXEC.
```

```
MOVE 'REGISTRO-2' TO REG-START.  
MOVE LENGTH OF REG-START TO COM-LEN.  
EXEC CICS START  
    TRANSID ('BBBB')  
    FROM (REG-START)  
    LENGTH (COM-LEN)  
END-EXEC.
```

TRANSACCIÓN BBBB

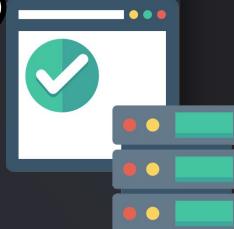
```
MOVE LENGTH OF REG-START TO COM-LEN.  
EXEC CICS RETRIEVE  
    INTO (REG-START)  
    LENGTH (COM-LEN)  
END-EXEC.
```

RECUPERA
'REGISTRO-1'

TRANSACCIÓN BBBB

```
MOVE LENGTH OF REG-START TO COM-LEN.  
EXEC CICS RETRIEVE  
    INTO (REG-START)  
    LENGTH (COM-LEN)  
END-EXEC.
```

RECUPERA
'REGISTRO-2'





04

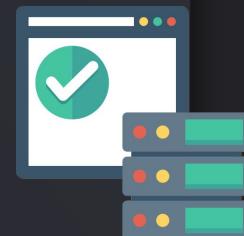
...

COMANDO RETRIEVE

```
EXEC CICS RETRIEVE  
    {INTO ('data-area') : SET (pointer-ref)}  
    LENGTH      ('data-value')  
END-EXEC.
```

Este comando permite recuperar un área de datos enviada desde otro programa con el comando **START**.

Para determinar si luego del **RETRIEVE** se han obtenido datos en la '**data-area**' del **INTO**, deberá chequearse la condición de excepción **ENDDATA**.





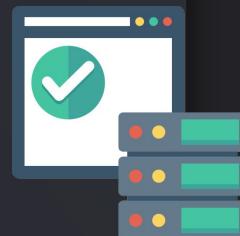
04

...

COMANDO CANCEL

```
EXEC CICS CANCEL  
    REQID ('name')  
END-EXEC.
```

Este comando permite cancelar la ejecución de una tarea cuya ejecución fue solicitada vía **START**. El parámetro '**name**' del argumento **REQID** debe coincidir con el especificado en el **START** correspondiente.



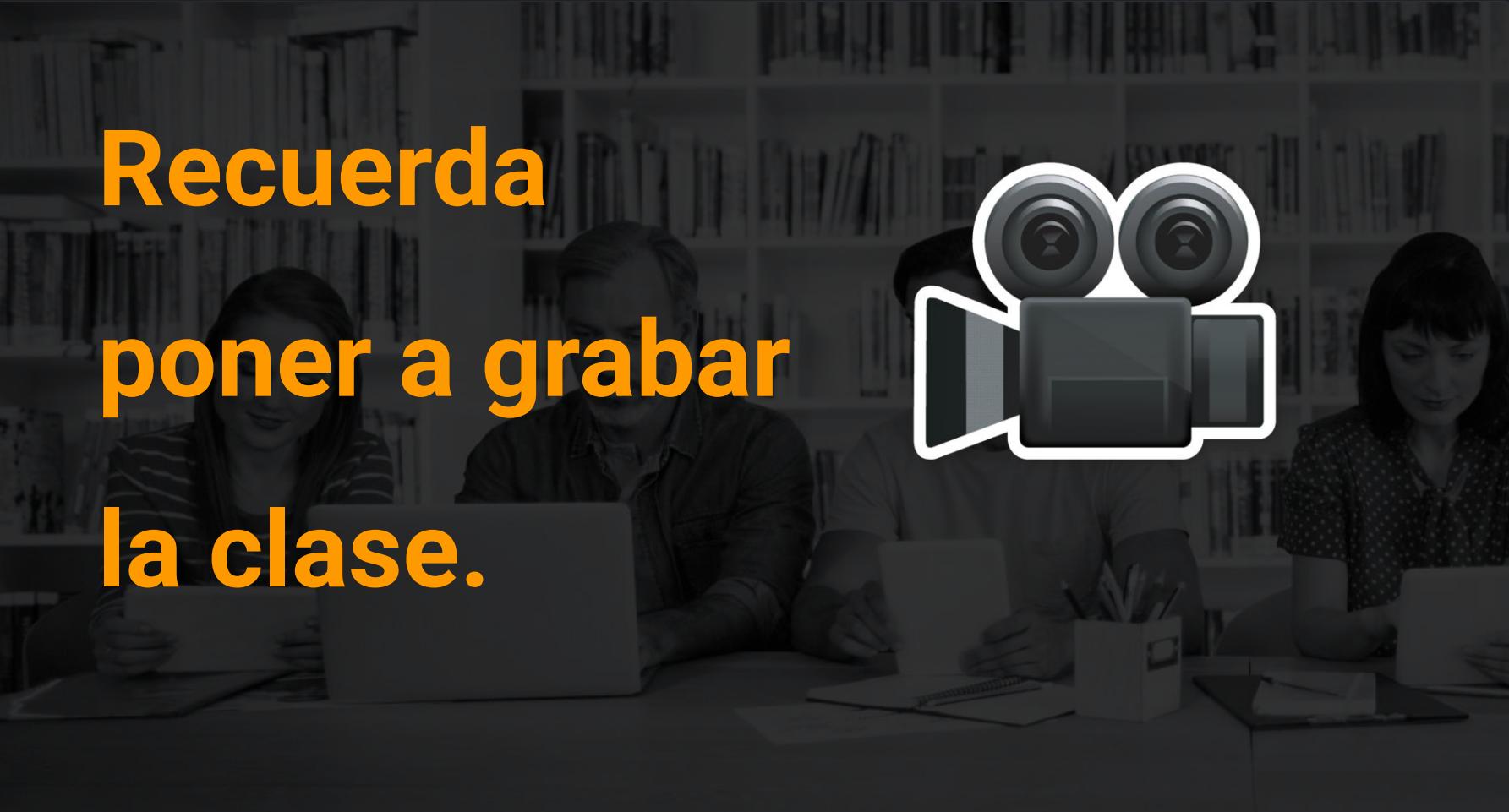
¿Consultas?



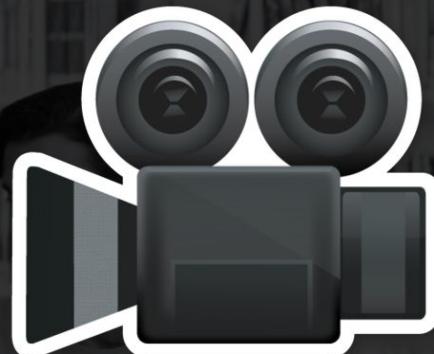
**¡Muchas
gracias!**



Programador COBOL CICS



Recuerda
poner a grabar
la clase.





Conceptos adicionales en CICS



... Mapa Conceptual

01

TEMPORARY
STORAGE

02

TRANSIENT
DATA

03

UNIDAD DE
TRABAJO



... **Agenda**

- 1 Temporary Storage
- 2 Transient Data
- 3 Concepto de Unidad de Trabajo





01 Manejo de Temporary Storage

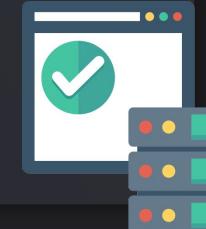
La **TEMPORARY STORAGE** o **TS** es un recurso que posibilita el pasaje de información entre transacciones. Físicamente se puede almacenar en la Memoria Principal o en Memoria Auxiliar, según los requerimientos del programa.

Puede ser utilizada, por ejemplo, para los siguientes objetivos:

1. Al efectuar la muestra de la selección de varias filas de una tabla o de registros que resulta demasiado grande como para utilizar solo una pantalla para lograrlo. Con la utilización de las PFs podemos acceder a diferentes visiones de la información (Paginado con **TS**).
2. Para pasarle los registros a imprimir a una transacción Asincrónica.
3. Para obtener una lista de registros al final de una cadena de transacciones donde cada una genera uno o más ítems de la **TS** y la última procesa todo el conjunto.

Cuando el programa determina que la **TS** será asignada en Memoria Auxiliar, los ítems generados se grabarán en un único archivo **VSAM** para cada **CICS** y que fue definido en el stream de arranque del CICS.

Los grupos de registros de **TS** se denominan **COLAS**. Las colas no se definen en ninguna tabla de **CICS**, ya que son definidas dentro de cada programa con indicación de su nombre (hasta 8 caracteres) y grabando uno o más registros utilizando ese nombre.





01

Manejo de Temporary Storage

...

La **COLA** una vez que fue creada, permanecerá hasta que sea borrada por alguna transacción. Cualquier programa que conozca el nombre de la cola podrá acceder a sus datos en forma Secuencial o Random.

Para acceder en forma Random, se debe especificar el nombre de la **COLA** y el número de registro al que se desea acceder (**ITEM**). Por cada registro que se grabe en la **COLA**, el **ÍTEM** se incrementará automáticamente en 1.

Al asignar el nombre de la cola, es importante considerar que este debe identificarse únicamente. La forma más tradicional es utilizar los primeros 4 caracteres de los ocho del nombre con el nombre de la transacción que genera la cola y los cuatro restantes con el nombre de la terminal donde se está ejecutando la transacción.





01

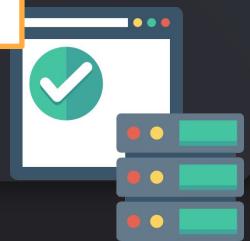
...

Manejo de TS - WRITEQ

El comando **WRITEQ** es usado para grabar nuevos registros o actualizar registros existentes en la Temporary Storage. Si la cola especificada en el comando **WRITEQ TS** ya existe, agrega nuevos registros al final de la cola, caso contrario crea la cola y graba el primer registro.

Si se especifica **ITEM**, el **CICS** retornará el número de ítem del nuevo registro en el área especificada. El número del nuevo registro será el número de ítem del registro anterior más 1. Si **ITEM** no se especifica, el número de ítem no será retornado al programa.

REWRITE es usado para que un registro existente en la cola sea actualizado. Si **REWRITE** es especificado, **ITEM** también debe ser especificado, colocando el número de ítem que se quiera actualizar. Esto causará que el **WRITEQ TS** actualice el registro de Temporary Storage cuyo número de ítem coincida con el especificado en **ITEM**. No es obligatorio leer la entrada de la cola antes de actualizar, pero es una buena práctica de programación hacerlo.





... 01

Manejo de TS - READQ

El comando **READQ** permite recuperar un item de **TS**.

EXEC CICS READQ TS

```
QUEUE ('name')
{ SET (pointer-ref) : INTO (data-area) }
LENGTH (data-value)
[ ITEM (data-value) : NEXT ]
[ NUMITEMS ]
```

END-EXEC.

- **LENGTH** identifica la longitud del registro a grabar. Los registros de las **TS** no necesitan tener la misma longitud. Es un campo **PIC S9(4) COMP**. Si la longitud del registro excede este valor, será truncado y la condición de excepción **LENGERR** se activará. Para evitar este inconveniente, es recomendable asignar siempre la longitud máxima de registro antes de efectuar la lectura.
- **ITEM** en caso de acceso **RANDOM**, especifica el número de registro a leer. Si el registro requerido no existe, la condición de excepción **ITEMERR** se activa. Si la Cola no existe, se activa entonces **QIDERR**.
- **NEXT** especifica que se lea el siguiente registro (opción default). Si ya no hay más registros, será activada la condición de excepción **ITEMERR** como consecuencia de haber alcanzado el final de la **TS**.
- **NUMITEMS** es un área de datos a ser actualizada por el **CICS** donde se especifica la cantidad de items total que posee la **COLA**.





... 01

Manejo de TS - DELETEQ

El comando **DELETEQ** permite el borrado de una **COLA**.

```
EXEC CICS DELETEQ TS  
  QUEUE ('name')  
END-EXEC.
```

Este comando se puede usar para borrar una cola completa de **TS**. Si la cola no existe, se activa la condición de excepción **QIDERR**

Antes de crear una nueva **COLA**, resulta conveniente ejecutar este comando para asegurarnos que la misma no existe.

IMPORTANTE: No se puede borrar un **ÍTEM** en forma aislada de una **TS**. Si debemos trabajar con bajas de **ÍTEMS**, deberemos habilitar una Flag dentro de nuestro diseño de **COLA** para considerar esta situación.





... 01

TS – CONDICIONES DE EXCEPCIÓN

- **ITEMERR** para acceso random, el número de registro especificado no existe en la cola para acceso secuencial, se alcanzó el fin de la cola
- **LENGERR** el registro leído es más largo que el especificado
- **NOSPACE** el archivo VSAM en el cual se graban los registros de Temporary Storage carece de espacio
- **IOERR** error de Entrada/Salida
- **INVREQ** la longitud de los datos a grabar es cero o excede la capacidad especificada para el archivo VSAM
- **QIDERR** no existe la cola especificada

La acción asumida por el CICS en todas estas condiciones de excepción es terminar anormalmente la tarea excepto en el caso de NOSPACE, donde la tarea se suspende hasta que se disponga de espacio.





... Mapa Conceptual

01
TEMPORARY
STORAGE

02
TRANSIENT
DATA

03
UNIDAD DE
TRABAJO





... 02

MANEJO DE TRANSIENT DATA

En aplicaciones batch, podemos especificar un archivo temporal que será borrado al final del job, para pasar datos entre pasos del job.

A diferencia de las colas de temporary storage que pueden ser accedidas en modo random o secuencial tantas veces como se desee, en una cola de **TRANSIENT DATA** los registros son tomados de la cola en el mismo orden en que ellos fueron grabados. No pueden ser actualizados. Si una tarea lee un registro de una cola, el registro se torna inaccesible. Ni esa tarea ni ninguna otra podrá leerlo nuevamente.

Una cola de **TRANSIENT DATA** debe ser definida en la Tabla de Control de Destinos (Destination Control Table). Esta tabla indica qué tipo de cola transient data será usada: Intrapartition Destination o Extrapartition Destination.

Intrapartition Destination

La utilización de esta cola solo estará disponible dentro del ambiente de **CICS** en la que está definida. Esta cola soporta la función de **TRIGGER**, es decir que por ejemplo cuando supere una cierta cantidad de ítems se disparará automáticamente la transacción indicada en la **DCT**.

Extrapartition Destination

La utilización de esta cola estará disponible dentro y fuera del ambiente de **CICS** en la que está definida. Puede ser de **INPUT** o de **OUTPUT**, pero no ambas cosas. No permite la función de **TRIGGER**.





... 02

MANEJO DE TD - WRITEQ

El comando **WRITEQ** permite la generar un nuevo item de **TD**.

EXEC CICS WRITEQ TD

```
QUEUE ('name')
FROM (data-area)
[ LENGTH (data-value) ]
```

END-EXEC.

- **QUEUE** especifica el nombre de la cola donde se generará el registro nuevo.
- **FROM** identifica el área de trabajo en la que se encuentra el registro a grabar.
- **LENGTH** identifica la longitud del registro a grabar. No debe especificarse si la **TD** es una Extrapartition Transient Data con registros de longitud fija.





... 02

MANEJO DE TD - READQ

El comando **READQ** permite recuperar un ítem de **TD**.

EXEC CICS READQ TD

```
QUEUE ('name')
INTO (data-area)
LENGTH (data-value)
```

END-EXEC.

Recordemos que los ítems de las **TD** se acceden en forma secuencial, por lo que no es necesario indicar el **ÍTEM de TD** a recuperar.

Una vez recuperado el registro, es eliminado automáticamente de la **TD**.

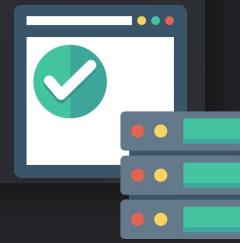




... 02

TD - CONDICIONES DE EXCEPCIÓN

- **QZERO** la **TD** que se está leyendo está vacía.
- **LENGERR** el registro leído es más largo que el especificado
- **NOSPACE** el archivo **VSAM** de Intrapartition en el cual se graban los registros de **TD** carece de espacio para grabar otro registro.
- **IOERR** error de Entrada/Salida
- **NOTOPEN** El archivo **VSAM** de destino no está abierto en el **CICS**.
- **QBUSY** otra tarea está utilizando el recurso de destino de Intrapartition cuando se ejecuta un comando **READQ**.
- **QIDERR** no existe la cola especificada.





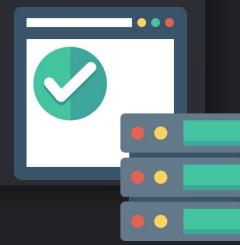
... 02

MANEJO DE TD - DELETEQ

El comando **DELETEQ** permite el borrado de una **COLA TRANSIENT DATA**.

```
EXEC CICS DELETEQ TD  
    QUEUE ('name')  
END-EXEC.
```

IMPORTANTE: Si la **COLA** es una Intrapartition Destination, es posible borrar el contenido de la cola entera con el comando **DELETEQ**. Todo el espacio asignado será liberado. Sin embargo, cualquier transacción CICS puede grabar nuevos registros en la **COLA** que podrán ser leídos por la misma transacción o por cualquier otra.





... Mapa Conceptual

01
TEMPORARY
STORAGE

02
TRANSIENT
DATA

03
UNIDAD DE
TRABAJO





...

03

CONCEPTO DE UNIDAD DE TRABAJO

UNIDAD LÓGICA DE TRABAJO

- CONCEPTOS
- SYNCPOINT
- SYNCPOINT ROLLBACK
- ABEND





03

CONCEPTO DE UNIDAD DE TRABAJO

El **CICS** posee una facilidad que es el **DYNAMIC TRANSACTION BACKOUT** y consiste en restaurar automáticamente todos los recursos utilizados por una transacción que cancela por condiciones de excepción no contempladas dentro del programa al estado del último **SYNCPOINT** ejecutado, o al estado que tenían antes del inicio de la transacción que cancela si no se ejecutó ningún **SYNCPOINT** dentro del programa.

Para aprovechar al máximo esta facilidad, es necesario conocer qué es una **UNIDAD LÓGICA de TRABAJO**, donde comienza y dónde termina.

Una **UNIDAD LÓGICA DE TRABAJO** es el agrupamiento de todas las tareas de actualización de recursos que deben ser ejecutadas por una o varias transacciones, y donde todas deben tener éxito en su ejecución. Si alguna de ellas falla, debemos considerar como de ejecución inválida a todas las transacciones intervinientes, aún a aquellas transacciones ejecutadas exitosamente, porque el resultado general es inválido por la falla en una de sus partes.

La **UNIDAD LÓGICA DE TRABAJO** comienza con la ejecución de la primera transacción de actualización de lo que nosotros, en nuestro diseño de la aplicación, consideramos como una **UNIDAD LÓGICA DE TRABAJO**, y finaliza cuando:

- A. Termina la Transacción de mayor nivel
- B. Al momento de la ejecución del comando SYNCPOINT





...

03

CONCEPTO DE UNIDAD DE TRABAJO

El **CICS**, por su parte, efectúa un **SYNCPOINT** (o punto de sincronismo) automático si la transacción principal termina exitosamente, es decir sin ninguna cancelación por condiciones de excepción.

Un ejemplo de **UNIDAD LÓGICA de TRABAJO** podría ser que un cliente efectúa una transferencia de dinero desde su cuenta de ahorro a su cuenta corriente. Esto se puede dividir en tres procesos

1. Efectuar el débito en la caja de ahorro
2. Efectuar el crédito en la cuenta corriente
3. Registrar el movimiento de transferencia

El conjunto de las tres transacciones podríamos considerarlo como nuestra **UNIDAD LÓGICA de TRABAJO**, ya que si alguna de ellas no se ejecuta en forma satisfactoria, toda la operación debe ser considerada como no efectuada.

Solamente pensemos qué pasaría si la transacción 1) finaliza normalmente, la transacción 2) finaliza normalmente y la transacción 3) falla en la registración. Tendríamos un problema en nuestros registros.

En este caso, deberíamos efectuar una vuelta atrás o **BACKOUT** de lo hecho por las transacciones 1) y 2) para invalidar toda la transacción en su conjunto y dar cuenta al usuario operador del problema encontrado al momento de ejecutar la transacción 3).





...

03

ABEND

El comando **ABEND** además de restaurar todos los recursos actualizados al estado inicial de nuestra **UNIDAD LÓGICA DE CONTROL**, permite efectuar un vuelco de memoria para su posterior análisis.

```
EXEC CICS ABEND  
  [ ABCODE (name) ]  
  [ CANCEL ]  
END-EXEC.
```

- **ABCODE** producirá un vuelco de memoria que será almacenado en el archivo especificado por 'name'
- **CANCEL** produce la suspensión de cualquier rutina de **HANDLE ABEND** vigente (para no entrar en loop).

Si nuestra aplicación utiliza recursos como el **DL/I** o **DB2**, el comando **ABEND** también restaura todos los cambios efectuados en ellos.





...

03

Transacciones propias de CICS

Transactions (1 of 2)

Function	CICS-supplied Transaction
Sign On	CESN
Sign Off	CESF
Monitor and Control CICS Resources	CEMT
Message Switching	CMSG
Resource Definition Online (RDO)	CEDA
	CEDB
	CEDC
CICS DB2 Interface	DSNC or CEMT





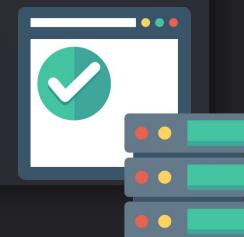
...

03

Transacciones propias de CICS

Transactions (2 of 2)

Function	CICS-supplied Transaction
Processing and Debugging Programs	CMAC
	CECI
	CEBR
	CEDF
	CEDX
Database Control Inquiry	CDBI
Database Control Interface	CDBM
Database Control Menu	CDBC



**¡Muchas
gracias!**



Clase 22 asincrónica - práctica con CICS

OBJETIVO

- Construir un código COBOL CICS para mostrar un Menú de opciones al usuario
- Al finalizar esta práctica; el estudiante habrá estado practicando la codificación de un programa pseudo conversacional COBOL CICS

ESPECIFICACIONES

PROGRAMA CICS - MODIFICACIONES

TRANSACCIÓN FXXX - PROGRAMA PGMMOXXX - MAPA MAP5XXX

(donde xxx = nro. alumno)

LAYOUT MAPA (MAP5XXX) : en página siguiente

<p>PGMMOC12-MODIFICACIÓN DE CLIENTES</p>	FC12-MAP5C12 99-99-9999
<p>Cargar los datos necesarios y presionar 'ENTER'</p> <p>Tipo de documento: XX Nro.de documento: ZZZZZZZZZ9</p> <p>Nombre y apellido: XXXXXXXXXXXXXXXXXXXXXXXXXXXX Fecha nacimiento: 99/99/9999 Sexo: X (F/M)</p> <p style="font-size: small; margin-top: 20px;">Presione tecla ENTER procesar F3 limpiar F12 salir</p> <p style="font-size: small; margin-top: 10px;">xx</p>	
 72 caracteres	

Modificar el programa CICS **PGMPRxss** para que acepte como IO , MAPA (MAP5XXX): MODIFICACIÓN CLIENTES o sea llamado desde el PGMMExxx; en este caso considerar que recibe una COMMAREA con los datos de tipo y nro de documento.

Luego de recibidos y validados los datos ingresados; se buscará el cliente en el archivo de Personas, con los datos validados OK:

- Archivo VSAM KSDS = CLIENTES

LAY OUT ALU9999.CURSOS.COPYLIB(CPPERSON)

datos en: KC03xxx.CURSOS.PERSONA.KSDS.VSAM (**DEFINIDO EN TABLA CICS CON DDNAME: PERSOxxx**)

y se enviará el mapa con los datos del cliente solicitado.

NOTA IMPORTANTE:

Los campos pintados de “amarillo” serán modificables por el usuario

Los campos pintados de “turquesa”; serán NO MODIFICABLES

Se mostrarán los errores al pie de página.

RESUMEN DE NAVEGABILIDAD DEL PROGRAMA

1) ¿Cómo ingresaron los datos?

- a) vía SEND/RECEIVE de MAPA? En este caso; la COMMAREA estará vacía.

Controlar que el usuario PRESIONA ENTER, en ese caso, se validarán los datos ingresados de TIPO Y NRO DE DOCUMENTO; y ante validación exitosa; se enviará el MAPA con los datos del Cliente. Si se encuentra algún problema; indicarlo en la línea de mensajes de 72 caracteres disponible.
Ejemplo: cliente dado de baja.

Para finalizar; lo hará al recibir la solicitud del usuario con PF12 (retornando a CICS)

Si el usuario presiona PF3; significa que hay que limpiar el MAPA y volver a enviarlo sin datos para una nueva consulta (mostrar solamente TIPO y NÚMERO DE DOCUMENTO)

- b) vía COMMAREA? En este caso; la COMMAREA tendrá datos.

En este caso; se validará los datos ingresados en COMMAREA de TIPO Y NRO DE DOCUMENTO; y luego se enviarán todos los datos del cliente y los necesarios para posibilitar la ejecución del programa llamador; también vía COMMAREA:

- Nombre de programa llamador
- Tipo y número de documento
- Nombre y apellido del cliente
- Fecha de nacimiento del cliente
- Sexo del cliente
- Código de retorno

TENER PRESENTE QUE LA COMMAREA EN ambos programas llamador y llamado, deberá ser de la misma estructura de datos y del mismo largo total; considerando un filler con espacios al final.

Para finalizar; lo hará retornando al programa LLAMADOR (RETURN TRANSID)

Cómo se integra el campo CÓDIGO DE RETORNO?

- Si la validación fue exitosa: devolver zeros en este campo
- Si la validación tuvo algún problema; integrar:
 - 01 para tipo documento erróneo
 - 02 para número de documento no informado
 - 03 para cliente inexistente
 - 04 para cliente dado de baj

NUEVA CONSIGNA:

Modificar el programa CICS **PGMMExxx** para que en caso que el USUARIO SOLICITE MODIFICACIÓN; realice XCTL PGMMOxxx con la COMMAREA correspondiente.

OBJETIVO

- Luego de leído y comprendido este documento el estudiante habrá APRENDIDO a trabajar con la metodología propuesta durante este curso respecto del diseño de mapas CICS bajo BMS Basic Mapping Support.

ESPECIFICACIONES

¿Cuál es el concepto fundamental?

Cada **MAPA** tendrá disponibles **24** filas y **80** columnas.

Estaremos dividiendo cada MAPA, en tres zonas bien determinadas:

1- **ENCABEZADO** (alto: primeras 5 filas. ancho: las 80 columnas)

campo 1.1 fila 1 columna 5: nombre del programa, ‘-’, título del mapa

campo 1.2 fila 1 columna 50: transacción, ‘-’ , nombre del mapa.

- Ejemplo: **AC12-MAP1C12**

campo 1.3 fila 1 columna 50: indicar fecha de ejecución

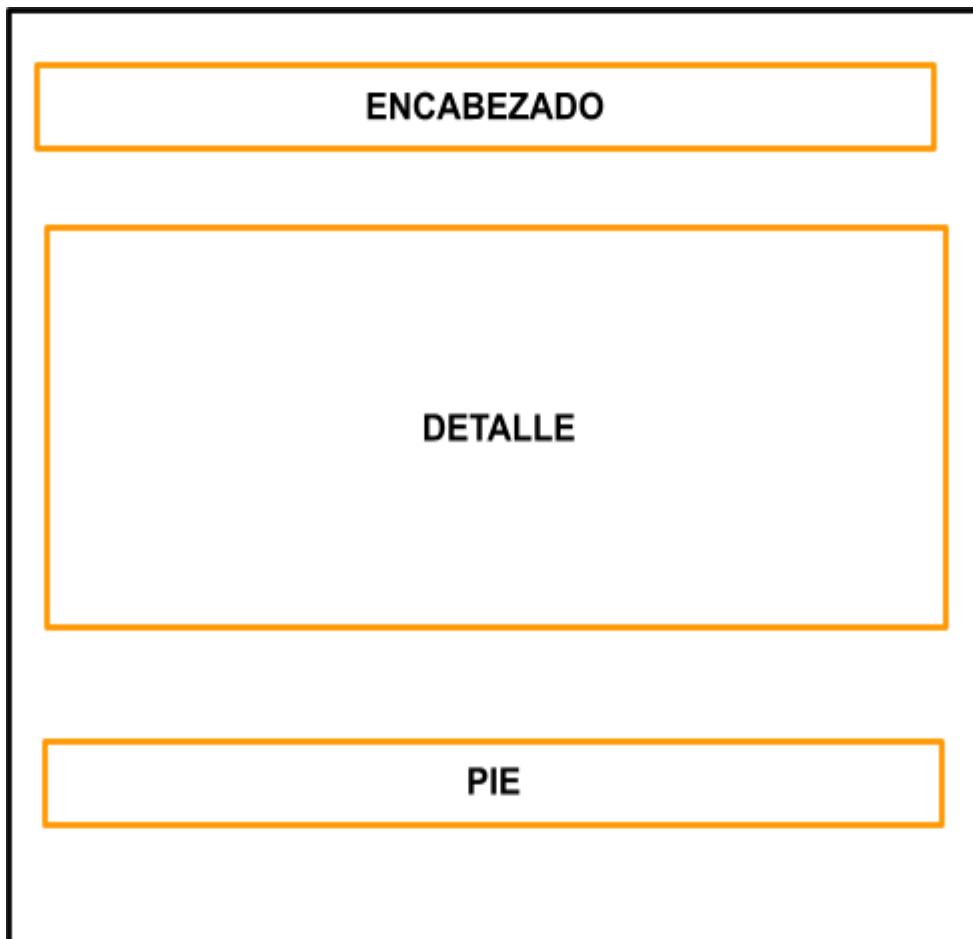
2- **DETALLE** (alto: fila 10 hasta fila 21. ancho: las 80 columnas)

campo 2.1 fila 10 columna 5: se definen los campos que se muestran

3- **PIE** (alto fila 22 hasta fila 24. ancho: las 80 columnas)

campo 3.1 filas 22 y 23 columna 5: teclas de función para la navegabilidad

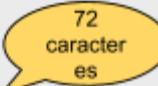
campo 3.2 fila 24 columna 5: mensaje enviado por la transacción CICS



Ejemplo de Menú:

PGMMEC12 - MENÚ DE CLIENTES F1 - CONSULTA F2 - ALTA F3 - BAJA F4 - MODIFICACIÓN	BC12-MAP2C12 99-99-9999
<p>Presione tecla de función o</p> <p>xx</p>	
F12 salir	

Ejemplo de Consulta:

PGMPRC12-CONSULTA DE CLIENTES	AC12-MAP1C12 99-99-9999
TIPO DOCUMENTO: XX	NÚMERO DOCUMENTO: ZZZZZZZZZZ9
<p>NRO. CLI.: ZZ9 NOMBRE: XXXXXXXXXXXXXXXXXXXXXXXXX DIRECCIÓN: XXXXXXXXXXXXXXXXXXXXXXXXX EMAIL: XXXXXXXXXXXXXXXXXXXXXXXXX TELÉFONO: XXXXXXXXXXXXXXXX</p> 	
<p>Presione tecla ENTER buscar F3 limpiar F4 siguiente xx</p> 	

Campos pintados de amarillo: son modificables para permitir al usuario cargar su contenido

El resto de los campos serán **NO MODIFICABLES**.

Ejemplo de Alta:

PGMALC12-ALTA DE CLIENTES	DC12-MAP3C12 99-99-9999
---------------------------	----------------------------

Cargar los datos y presionar 'ENTER'

Tipo de documento: **XX**
Nro.de documento: **ZZZZZZZZZ9**

Nombre y apellido: **XXXXXXXXXXXXXXXXXXXXXXXXXXXX**
Fecha nacimiento: **99/99/9999**
Sexo: **X (F/M)**

Presione tecla **ENTER** seleccionar **F3 limpiar**
xx **F12 salir**

72 caracteres

En este mapa **TODOS LOS CAMPOS PINTADOS DE AMARILLO**; resultan de carga OBLIGATORIA. Dado que son datos imprescindibles para generar el ALTA (WRITE) en el archivo VSAM DE PERSONAS.

Ejemplo de Baja:

PGMBAC12-BAJA DE CLIENTES	EC12-MAP4C12 99-99-9999
Cargar los datos y presionar 'ENTER'	
Tipo de documento: XX Nro.de documento: ZZZZZZZZZ9	
Presione tecla ENTER seleccionar F3 limpiar F4 confirmar F12 salir xx	

72
caracteres

En este mapa **TODOS LOS CAMPOS PINTADOS DE AMARILLO**; resultan de carga OBLIGATORIA. Dado que son datos imprescindibles para generar la BAJA lógica (REWRITE) en el archivo VSAM DE PERSONAS.

Ejemplo de Modificación:

PGMMOC12-MODIFICACIÓN DE CLIENTES	FC12-MAP5C12
	99-99-9999

Cargar los datos necesarios y presionar 'ENTER'

Tipo de documento: **XX**
Nro.de documento: **ZZZZZZZZZ9**

Nombre y apellido: **XXXXXXXXXXXXXXXXXXXXXXXXXXXX**
Fecha nacimiento: **99/99/9999**
Sexo: **X (F/M)**

Presione tecla **ENTER** procesar **F3 limpiar** **F12 salir**

xx

72
caract
eres

En este mapa **TODOS LOS CAMPOS PINTADOS DE AMARILLO**; resultan de carga **OPCIONAL (excepto el Tipo y Nro de documento)**. Dado que solo se cargarán los datos que se requieran para generar la MODIFICACIÓN (REWRITE) en el archivo VSAM DE PERSONAS.

CICS - CONCEPTOS RELEVANTES

1. EIB – Exec Interface Block

- **DFHEIBLK**: estructura automática incorporada en la **LINKAGE SECTION** durante la traducción CICS.
 - **Finalidad**: Proveer a nuestro programa de información contextual sobre la ejecución: terminal, usuario, hora, resultado de comandos, etc.
 - **Campos clave**: **EIBAID**, **EIBCALEN**, **EIBDATE**, **EIBTIME**, **EIBRCODE**, **EIBTRNID**, entre otros. Se pueden consultar pero no modificar.
-

2. Testeo de Ejecución de Comandos

CICS provee mecanismos para validar si los comandos fueron exitosos:

A. HANDLE CONDITION

- Maneja condiciones de error mediante bifurcaciones GO TO.
- Puede manejar hasta 16 condiciones.
- Usado generalmente al inicio del programa para controlar errores comunes y excepciones.

B. RESP

- Permite capturar el código de resultado en una variable, que se evalúa luego con **DFHRESP**.
- Es más programático que **HANDLE**.

C. Combinación de ambos

- Puede usarse **HANDLE** para condiciones generales y **RESP** para tratamiento más fino en ciertas secciones.
-

3. HANDLE AID

- Similar a **HANDLE CONDITION** pero enfocado en teclas presionadas.
 - Se activa tras un **RECEIVE**, detecta claves como **ENTER**, **PF1–PF24**, **CLEAR**, etc.
 - Alternativa eficiente: evaluar **EIBAID** para decidir el flujo según la tecla.
-

4. Manejo de Fecha y Hora

- **ASKTIME**: obtiene fecha y hora en formato binario absoluto.

- **FORMATTIME:** convierte esa info a formatos legibles como YYMMDD, DDMMYY, hora hh:mm:ss, día de la semana, etc.
 - Muy útil para trazabilidad en registros.
-

5. Manejo de Mapas

A. SEND MAP

- Envía un mapa a la terminal.
- Opciones como DATAONLY, MAPONLY, ERASE, CURSOR, FREEKB, entre otras.
- Se requiere inicializar el área simbólica con LOW-VALUES.

B. RECEIVE MAP

- Recibe datos desde la terminal.
 - Los formatea según la definición del mapa y los ubica en la work-area.
-

6. Tratamiento de Archivos VSAM

A. READ

- Lectura con opciones como RIDFLD, KEYLENGTH, GENERIC, GTEQ, EQUAL.
- Excepciones manejables: NOTFND, LENGERR, DUPREC, IOERR.

B. WRITE

- Graba nuevos registros. Puede usar MASSINSERT para múltiples ingresos secuenciales.

C. REWRITE

- Actualiza registros previamente leídos con READ UPDATE.

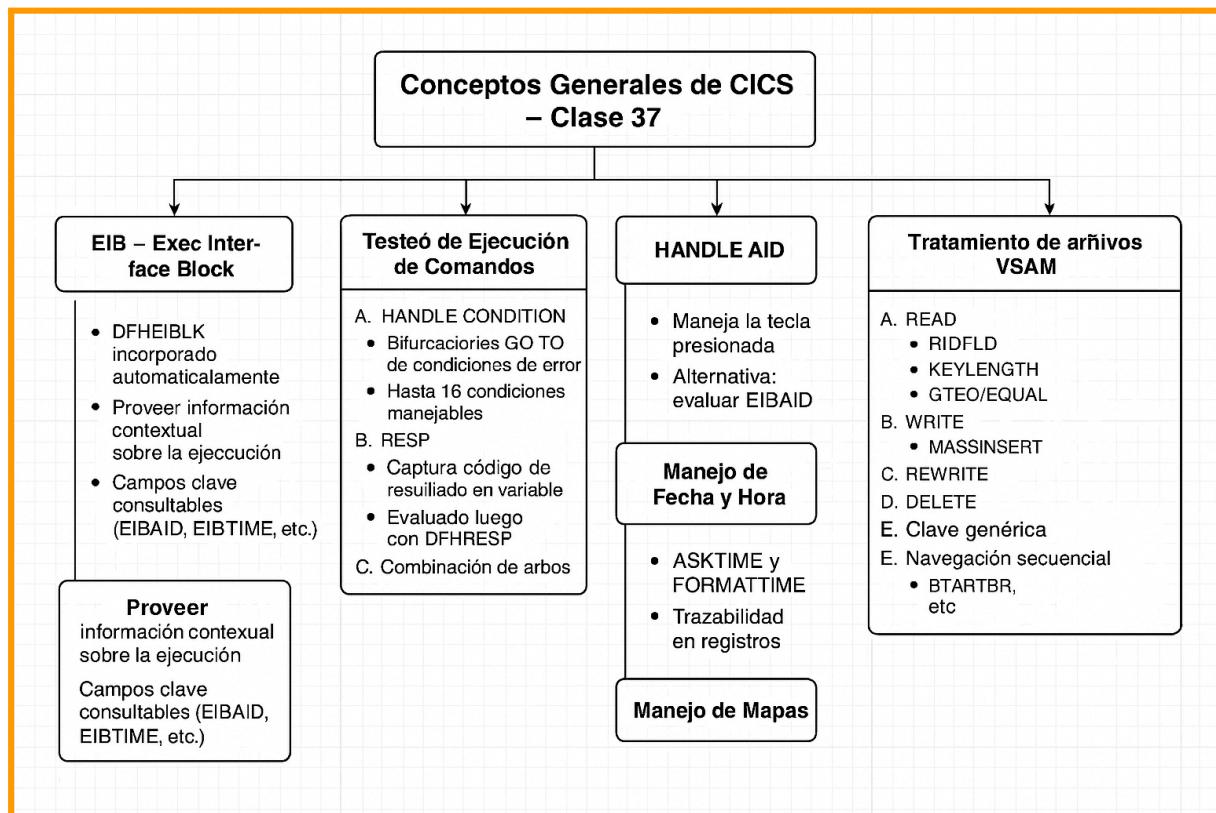
D. DELETE

- Borra registros. Puede usar clave genérica y NUMREC.

E. Navegación secuencial

- **STARTBR:** Posiciona para búsqueda.
 - **READNEXT, READPREV:** Lectura secuencial.
 - **RESETBR, ENDBR:** Control de navegación.
-

Todo esto forma la base de cómo interactuar con CICS para manejar terminales, datos, excepciones y archivos:



A continuación se explica cada uno de los conceptos que aparecen en el esquema visual anterior.

Este diagrama está dividido por bloques funcionales de una aplicación típica bajo CICS, donde cada elemento cumple un rol dentro del flujo de ejecución.

Explicando las partes:

1. EIB (Exec Interface Block)

- Es una estructura que CICS inyecta automáticamente en tu programa.
- Contiene información crítica sobre la ejecución, como el ID de transacción (**EIBTRNID**), terminal (**EIBTRMID**), hora (**EIBTIME**), estado del último comando (**EIBRCODE**), etc.
- Tu programa puede consultarla, pero no modificarla.**

2. Testeo del Resultado de Comandos CICS

Después de ejecutar cualquier comando CICS, debés verificar su resultado. Tenés tres formas:

✓ A. HANDLE CONDITION

- Intercepta condiciones de error predefinidas.
- Redirige el flujo del programa (como un `GO TO`) a un párrafo cuando se cumple esa condición.

▀ B. RESP

- Guarda el código de resultado de un comando en una variable.
- Luego se evalúa con `EVALUATE` y `DFHRESP()`.

💡 C. Combinación

- Podés usar ambos según el nivel de control que necesites: `HANDLE` para errores comunes, `RESP` para decisiones más específicas.
-

III 3. HANDLE AID y EIBAID

Controlan la lógica según la tecla que pulsa el usuario:

- `HANDLE AID` permite asociar teclas como `PF2`, `PF6`, `ENTER` a párrafos específicos del programa.
 - `EIBAID` contiene la última tecla presionada (te da más flexibilidad y mejor performance).
 - Usás uno u otro según tu estrategia de control de flujo.
-

4. Fecha y Hora – ASKTIME y FORMATTIME

- `ASKTIME` obtiene el tiempo absoluto desde 1900.
 - `FORMATTIME` convierte ese valor a formatos legibles: fecha (`DD/MM/AA, YYMMDD`), hora (`hh:mm:ss`), día del mes, día de la semana, etc.
 - Muy útil si necesitás trazabilidad o estampar marcas de tiempo.
-

5. Manejo de Mapas

⬆ A. SEND MAP

- Envía datos a la terminal en forma formateada.
- Opciones como `DATAONLY`, `ERASE`, `CURSOR` controlan qué se muestra y cómo.

⬇ B. RECEIVE MAP

- Recibe datos desde la terminal.
- Convierte la entrada en campos útiles según el mapa simbólico.

Ambos requieren haber definido los mapas previamente con BMS y mantener limpias las áreas de datos ([LOW-VALUES](#)).

6. Archivos VSAM

Una parte clave del flujo transaccional. Los comandos más importantes:

READ:

- Lee registros del archivo.
- Puede usar claves completas o genéricas ([KEYLENGTH](#), [GENERIC](#)).
- Opciones como [GTEQ](#) o [EQUAL](#) definen cómo buscar.

WRITE:

- Inserta nuevos registros.
- Puede usarse con [MASSINSERT](#) para inserciones múltiples.

REWRITE:

- Modifica registros que antes fueron leídos con [READ UPDATE](#).

DELETE:

- Elimina registros.
- Podés usar clave directa o genérica.

Navegación Secuencial:

- [STARTBR](#): posiciona el cursor para leer registros.
- [READNEXT / READPREV](#): lectura secuencial hacia adelante o atrás.
- [RESETBR](#): reinicia la búsqueda.
- [ENDBR](#): termina la búsqueda.

Todo esto se orquesta para que las aplicaciones CICS sean consistentes y eficientes al interactuar con el usuario y los datos.

HANDBOOK COBOL CICS

Se detallan los principales comandos y MACROS COBOL CICS.

SENTENCIAS DE ACCESO A LOS ARCHIVOS

Comando: DELETE

Resumen: para eliminar registros de un archivo VSAM

Ejemplo:

```
EXEC CICS DELETE DATASET ('name')
    [ RIDFLD (data-area) ]
    [ KEYLENGTH (data-value) ]
    [ GENERIC ]
    [ NUMREC (data-area) ]
    [ { RBA : RRN } ]
END-EXEC.
```

Comando: DELETEQ

Resumen: para eliminar registros en almacenamiento temporario (TS)

Ejemplo:

```
EXEC CICS DELETEQ
    TSQUEUE( name ) QNAME( name ) SYSID( systemname )
END-EXEC
```

Condiciones: INVREQ, ISCINVREQ, LOCKED, NOTAUTH, QIDERR, SYSIDERR

Comando: ENDBR

Resumen: permite terminar una búsqueda liberando los recursos afectados.

Ejemplo:

```
EXEC CICS ENDBR
    DATASET ('name')
END-EXEC.
```

Comando: READ

Resumen: para leer registros de archivo declarado en tablas propias de CICS (**FCT File Control Table**).

Ejemplo:

EXEC CICS

```
    READFILE( nombre de archivo )
    UNCOMMITTED CONSISTENT REPEATABLE  UPDATE TOKEN( Área-datos )
    INTO( Área-datos ) SET( ptr-ref )
    RIDFLD( Área-datos ) KEYLENGTH( Valor-datos )
    GENERICSYSID( nombre del sistema ) LENGTH( Área-datos )
END-EXEC.
```

Condiciones: DISABLED, DUPKEY, FILENOTFOUND, ILLOGIC, INVREQ, IOERR, ISCINVREQ, LENGERR, LOADING, LOCKED, NOSPACE, NOTAUTH, NOTFND, NOTOPEN, RECORDBUSY, SYSIDERR

Comando: READNEXT

Resumen: permite la lectura del siguiente registro de un archivo **VSAM** que se encuentra bajo el efecto del comando **STARTBR**

Ejemplo:

EXEC CICS READNEXT DATASET ('name')

```
    RIDFLD (data-area)
    [ KEYLENGTH (data-value) [ GENERIC ] ]
    [ {RBA : RRN } ]
    [ { SET (pointer-ref) : INTO (data-area) } ]
    [ LENGTH (data-area) ]
END-EXEC.
```

Comando: READPREV

Resumen: permite la lectura del registro anterior de un archivo **VSAM** que se encuentra bajo el efecto del comando **STARTBR**.

Ejemplo:

EXEC CICS READPREV DATASET ('name')

```
    RIDFLD (data-area)
    [ KEYLENGTH (data-value) [ GENERIC ] ]
    [ { RBA : RRN } ]
    [ { SET (pointer-ref) : INTO (data-area) } ]
    [ LENGTH (data-area) ]
END-EXEC.
```

Comando: RESETBR

Resumen: permite terminar una búsqueda y comenzar otra sobre el mismo archivo inmediatamente.

Ejemplo:

```
EXEC CICS RESETBR DATASET ('name')
    RIDFLD (data-area)
    [ KEYLENGTH (data-value) [ GENERIC ] ]
    [ { RBA : RRN } ]
    [ { GTEQ : EQUAL } ]
END-EXEC.
```

Comando: REWRITE

Resumen: permite la actualización de un registro existente en un archivo **VSAM**

Ejemplo:

```
EXEC CICS REWRITE DATASET ('name')
    FROM   (data-area)
    [ LENGTH (data-value) ]
END-EXEC.
```

Comando: STARTBR

Resumen: permite posicionar a la próxima operación de lectura en la clave que cumpla con determinada condición de búsqueda.

Ejemplo:

```
EXEC CICS STARTBR DATASET ('name')
    [ RIDFLD (data-area) ]
    [ KEYLENGTH (data-value) [ GENERIC ] ]
    [ { GTEQ : EQUAL } ]
    [ { RBA : RRN } ]
END-EXEC.
```

Comando: UNLOCK

Resumen: permite la liberación de los registros de un archivo que fueron accedidos con la opción **UPDATE**.

Ejemplo:

```
EXEC CICS UNLOCK
    DATASET ('name')
```

END-EXEC.

Comando: WRITE

Resumen: para grabar registros de archivo declarado en tablas propias de CICS (**FCT File Control Table**).

Ejemplo:

```
EXEC CICS  
    WRITE  
        FILE( filename ) MASSINSERT  
        FROM( data-area ) RIDFLD  
            ( data-area ) KEYLENGTH( data-value ) SYSID( systemname ) LENGTH( data-value )  
    END-EXEC.
```

Condiciones: DISABLED, DUPREC, FILENOTFOUND, ILLOGIC, INVREQ, IOERR, ISCINVREQ, LENGERR, LOADING, LOCKED, NOSPACE, NOTAUTH, NOTOPEN, RECORDBUSY, SUPPRESSED, SYSIDERR

Comando: WRITEQ

Resumen: para grabar registros en almacenamiento temporario (TS)

Ejemplo:

```
EXEC CICS  
    WRITEQ  
        TSQUEUE( name ) QNAME( name ) FROM( data-area )  
        LENGTH( data-value ) NUMITEMS( data-area ) ITEM( data-area )  
        REWRITESYSID( systemname ) AUXILIARY / MAIN / NOSUSPEND  
    END-EXEC.
```

Condiciones: INVREQ, IOERR, ISCINVREQ, ITEMERR, LENGERR, LOCKED, NOSPACE, NOTAUTH, QIDERR, SYSIDERR

SENTENCIAS DE CONTROL DE PROGRAMAS

- **CANCEL:** Este comando permite cancelar la ejecución de una tarea cuya ejecución fue solicitada vía **START**. El parámetro ‘**name**’ del argumento **REQID** debe coincidir con el especificado en el **START** correspondiente.

Ejemplo

```
EXEC CICS CANCEL  
    REQID ('name')  
END-EXEC.
```

- **Concepto de COMMAREA** = Área de comunicación entre programas. Por ejemplo se utiliza en comando **LINK**.

Ejemplo:

WORKING STORAGE SECTION.

```
77 COM-LEN      PIC S9(4) COMP.  
01 COMMAREA-DSPT020.  
  03 PARM1       PIC X(03).  
  03 PARM2       PIC X(03).
```

- **LINK** = Este comando es utilizado para disparar sincrónicamente un programa de nivel lógico inferior, retornado a la próxima instrucción del programa llamante.

Ejemplo:

```
EXEC CICS LINK  
    PROGRAM ('name')  
    [COMMAREA ('data-area')  
     LENGTH ('data-value')]  
END-EXEC.
```

- **RETURN** = Este comando es utilizado para TERMINAR una tarea. Provoca la liberación de todos los recursos y memoria asociados a esa TASK o tarea.

Ejemplo.

```
EXEC CICS RETURN  
    [ TRANSID ('name') ]  
    [COMMAREA ('data-area')  
     LENGTH ('data-value')]  
END-EXEC.
```

- **RETRIEVE:** Este comando permite recuperar un área de datos enviada desde otro programa con el comando START.

Ejemplo:

```
EXEC CICS RETRIEVE
  {INTO ('data-area') : SET (pointer-ref)}
    LENGTH      ('data-value')
END-EXEC.
```

- **START** = Este comando permite especificar que otra transacción sea invocada (**TRANSID**), en forma inmediata, o temporizada a una determinada hora del día, o luego de transcurrido determinado lapso de tiempo (**INTERVAL**).

Ejemplo:

```
EXEC CICS START
  TRANSID      ('name')
  [INTERVAL    (hhmmss) : TIME(hhmmss)]
  [FROM        ('data-area')
    LENGTH      ('data-value')]
  [TERMINAL   ('name')]
  [REQID      ('name')]
END-EXEC.
```

- **XCTL** = Este comando es utilizado para disparar asincrónicamente una transacción, no hay retorno al programa que emite el XCTL.

Ejemplo:

```
EXEC CICS XCTL
  PROGRAM     ('name')
  [COMMAREA   ('data-area')
    LENGTH      ('data-value')]
END-EXEC.
```

HANDBOOK BMS

Se detallan los principales comandos de proceso y macros de definición; para tratamiento de mapas en CICS

BMS – BASIC MAPPING SUPPORT

Macro DFHMSD: (DEFINE UN CONJUNTO DE MAPAS – MAPSET)

Mapset **DFHMSD TYPE={DSECT:MAP:FINAL:&SYSPARM},**
CTRL=([FRSET],[FREEKB],[PRINT],[ALARM]),
TIOAPFX=YES,
MODE={IN:INOUT:OUT},
LANG={COBOL:PLI:ASM:RPG},
STORAGE={AUTO:BASE=name}

Por supuesto, aquí vemos solo una pequeña visión de las opciones que soporta esta macro de CICS.

- **Mapset** Nombre del MAPSET (1 a 7 caracteres)
- Todo mapa BMS debe comenzar con **DFHMSD TYPE={DSECT:MAP:&SYSPARM}** y debe terminar con **DFHMSD TYPE=FINAL**.
- **DSECT** se utiliza para crear un conjunto de mapas simbólicos
- **MAP** se utiliza para crear un conjunto de mapas físicos

FINAL Indica la finalización de la definición del MAPSET o conjunto de mapas.

DFHMDI: (DEFINE UN MAPA DENTRO DEL MAPSET)

[mapname] DFHMDI [SIZE=(line,column)]

- **Mapname** Nombre del MAPA dentro del MAPSET (1 a 7 caracteres)
- **SIZE** Indica el tamaño de la pantalla expresado en cantidad de líneas y de columnas. Generalmente, en terminales del Host, los valores a indicar son (24,80)

Comando: ASKTIME

Resumen: para obtener la fecha y hora del Mainframe

Ejemplo:

EXEC CICS ASKTIME

[ABSTIME (data-area)]

END-EXEC.

ABSTIME (data-value)

[YYDDD(data-area)] [YYMMDD(data-area)] [YYDDMM(data-area)]

[DDMMYY(data-area)] [MMDDYY(data-area)] [DATE(data-area)]

[DATEFORM(data-area)] [DATESEP(data-value)] [DAYCOUNT(data-area)]

Comando: FORMATTIME

Resumen: para dar forma a la fecha y hora del Mainframe; según solicitud del usuario.

Ejemplo:

EXEC CICS FORMATTIME

[DAYOFWEEK(data-area)] [DAYOFMONTH(data-area)]

[MONTHOFYEAR(data-area)] [YEAR(data-area)]

[TIME(data-area) [TIMESEP(data-value)]]

END-EXEC.

Comando: RECEIVE

Resumen: para recibir un mapa desde la terminal.

Ejemplo:

EXEC CICS

RECEIVE MAP (WS-MAP)

MAPSET (WS-MAPSET)

INTO (MAP0299I)

RESP(WS-RESP)

END-EXEC

Comando: SEND

Resumen: para enviar un mapa a la terminal.

Ejemplo:

```
EXEC CICS SEND MAP ('name')
  [ MAPSET ('name') ]
  [ { DATAONLY : MAPONLY } ]
  [ FROM (data-area) ]
  [ LENGTH (data-value) ]
  [ CURSOR [ (data-value) ] ]
  [ { ERASE : ERASEUP } ]
  [ FREEKB ] [ FRSET ] [ ALARM ] [ PRINT ] [ FORMFEED ]
END-EXEC.
```

OBJETIVO

- Repasar paso a paso los principales conceptos teóricos de COBOL CICS a efectos que el estudiante pueda encarar su primer desafío de código COBOL pseudo-conversacional.

ESPECIFICACIONES

¿QUÉ ES CICS?

CICS, que es el acrónimo de: *Customer Information Control System*, es un servidor de transacciones desarrollado por IBM que se utiliza principalmente en sistemas mainframe para gestionar aplicaciones críticas de negocio en tiempo real.

Fue creado en 1968 y desde entonces ha evolucionado para soportar tecnologías modernas como Java, Node.js etc.

Su función principal es permitir que múltiples transacciones se procesen de forma simultánea, garantizando integridad, consistencia y disponibilidad. Esto lo hace ideal para sectores como la banca, los seguros y el gobierno, donde se requiere alta disponibilidad y rendimiento.

CICS permite desarrollar aplicaciones en lenguajes como COBOL y otros más, lo que facilita la integración de sistemas antiguos con nuevas tecnologías.

Además, ofrece herramientas para monitoreo, seguridad avanzada y recuperación ante fallos.

En el sector bancario, **CICS se utiliza como la columna vertebral de muchas operaciones críticas**. Su capacidad para manejar miles de transacciones por segundo lo convierte en una herramienta ideal para servicios como:

- **Procesamiento de transacciones financieras:** depósitos, retiros, transferencias y pagos se ejecutan en tiempo real con alta confiabilidad.
- **Gestión de cuentas:** permite acceder y actualizar información de cuentas bancarias de forma segura.
- **Integración con cajeros automáticos (ATM) y banca en línea:** CICS actúa como intermediario entre los sistemas front-end y las bases de datos centrales.
- **Cumplimiento normativo y auditoría:** registra cada transacción con precisión, lo que facilita el seguimiento y la trazabilidad.
- **Seguridad y control de acceso:** protege los datos sensibles mediante autenticación robusta y control de permisos.

Además de la banca, **CICS se utiliza en una variedad de industrias que requieren procesamiento de transacciones de alto volumen y alta disponibilidad.** Algunos ejemplos son:

- **Seguros:** para gestionar pólizas, procesar reclamaciones y calcular primas en tiempo real.
- **Gobierno:** en sistemas de recaudación de impuestos, seguridad social y registros civiles, donde se necesita integridad de datos y trazabilidad.
- **Retail y comercio electrónico:** para manejar inventarios, ventas, devoluciones y programas de fidelización de clientes.
- **Salud:** en la administración de historiales médicos, facturación y autorizaciones de tratamientos.
- **Transporte y logística:** para reservas, seguimiento de envíos y gestión de flotas.

CICS también se adapta a entornos modernos gracias a su compatibilidad con Apis, servicios web y lenguajes como Java y Node.js. Esto permite que empresas con sistemas heredados puedan integrarse con nuevas plataformas sin rehacer todo desde cero.

Vamos a ver cómo **CICS se está modernizando con inteligencia artificial (IA)** para seguir siendo relevante en los entornos empresariales actuales.

IBM ha desarrollado formas de **infundir IA directamente en las aplicaciones que corren sobre CICS**, lo que permite tomar decisiones en tiempo real dentro de las transacciones.

Por ejemplo:

- **Evaluación de riesgos en préstamos:** una transacción puede consultar un modelo de IA para decidir si aprobar o no un crédito en el momento exacto en que el cliente lo solicita.
- **Detección de fraude:** al procesar un reclamo de un seguro o una transacción bancaria, CICS puede invocar un modelo de IA que detecte patrones sospechosos sin salir del entorno mainframe.
- **Ofertas personalizadas:** mientras un cliente interactúa con un canal digital, CICS puede usar IA para recomendar productos financieros adaptados a su perfil.

CONFIGURACIÓN BÁSICA DE CICS

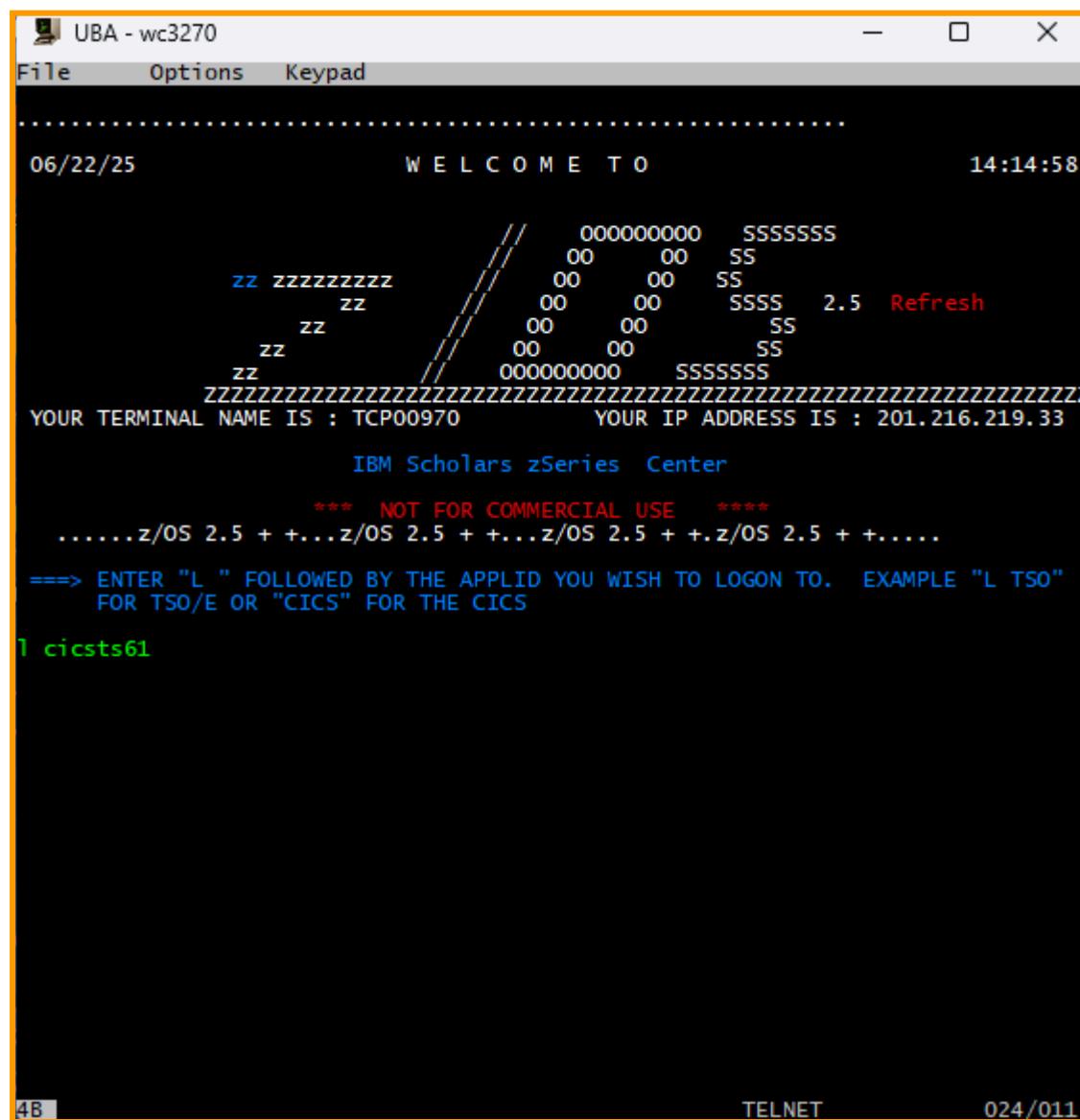
- LAS TABLAS PROPIAS

En CICS, las tablas propias son estructuras internas que definen cómo se comportan y se configuran los distintos componentes del sistema. Algunas de las más importantes incluyen:

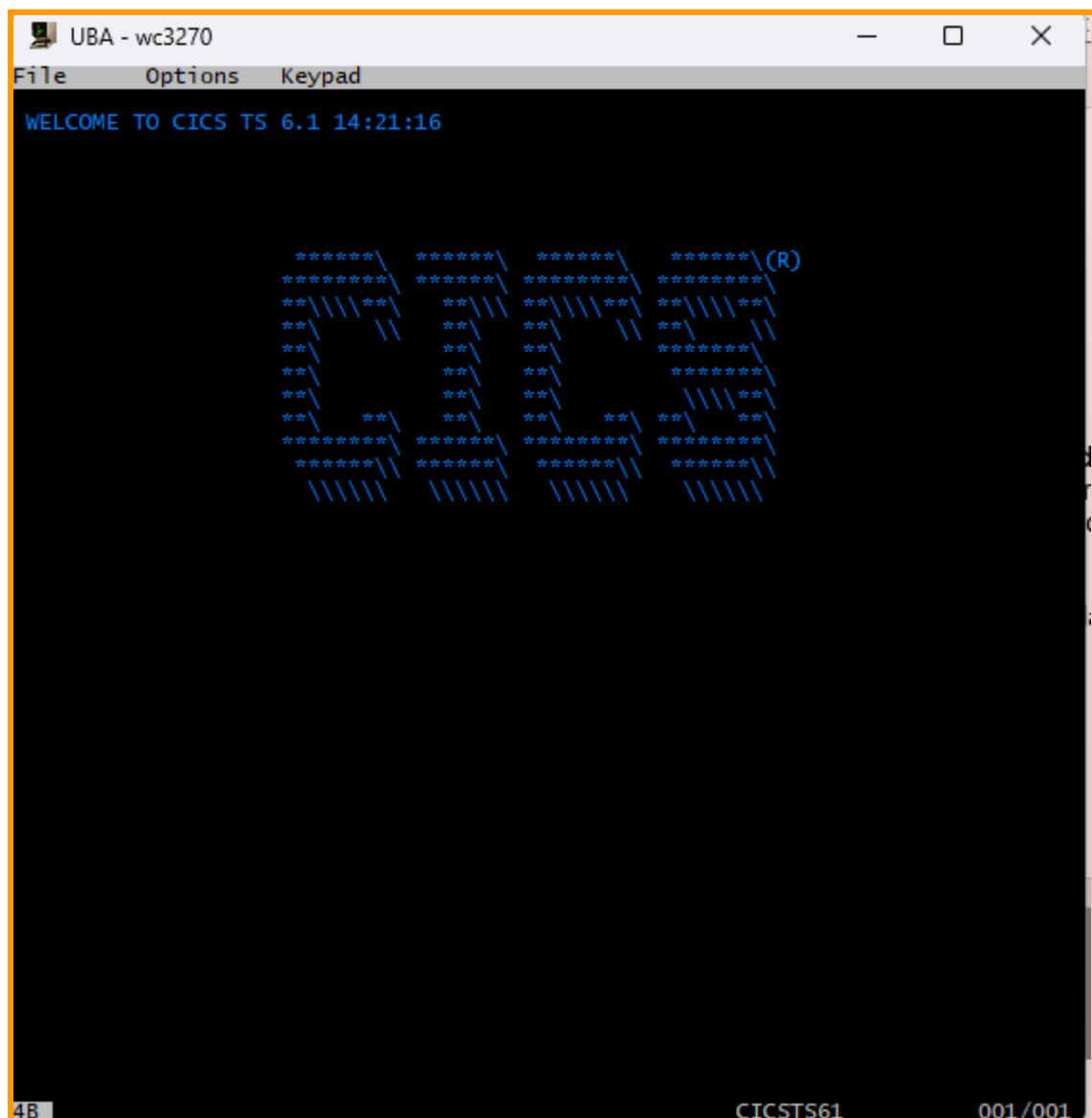
- **PCT (Program Control Table)**: define las transacciones disponibles y a qué programa están asociadas.
- **PPT (Processing Program Table)**: describe los programas que pueden ejecutarse, sus nombres y características.
- **FCT (File Control Table)**: especifica los archivos que CICS puede usar, como archivos VSAM.
- **TCT (Terminal Control Table)**: configura los terminales o dispositivos conectados al sistema.
- **DCT (Destination Control Table)**: se usa para definir destinos de impresión o colas de salida.
- **TST (Temporary Storage Table) y TDQ (Transient Data Queue)**: gestionan almacenamiento temporal y colas de datos transitorios respectivamente.

¿CÓMO INGRESAR A ENTORNO CICS?

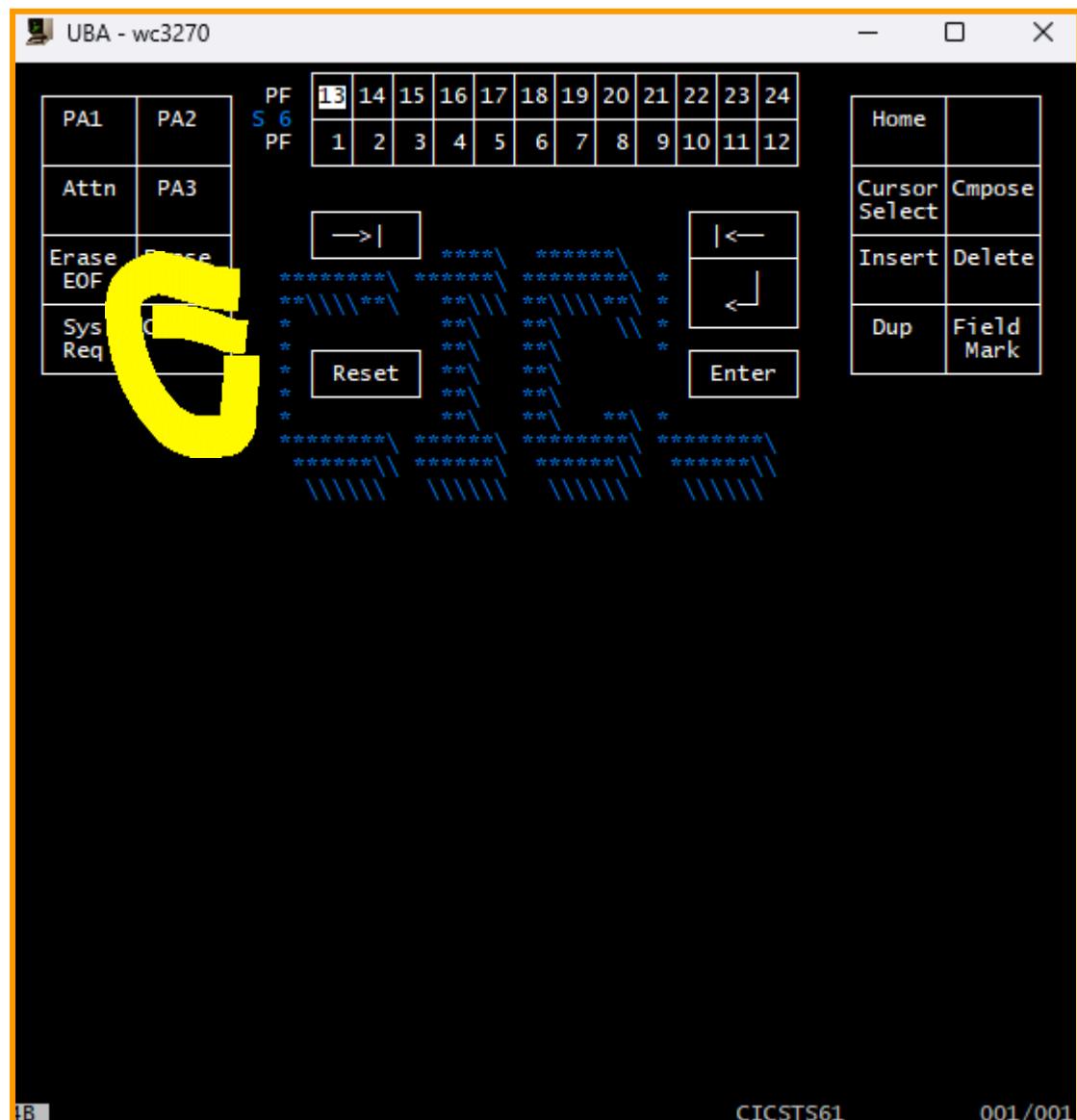
En pantalla de Z/OS: logon CICSTS61

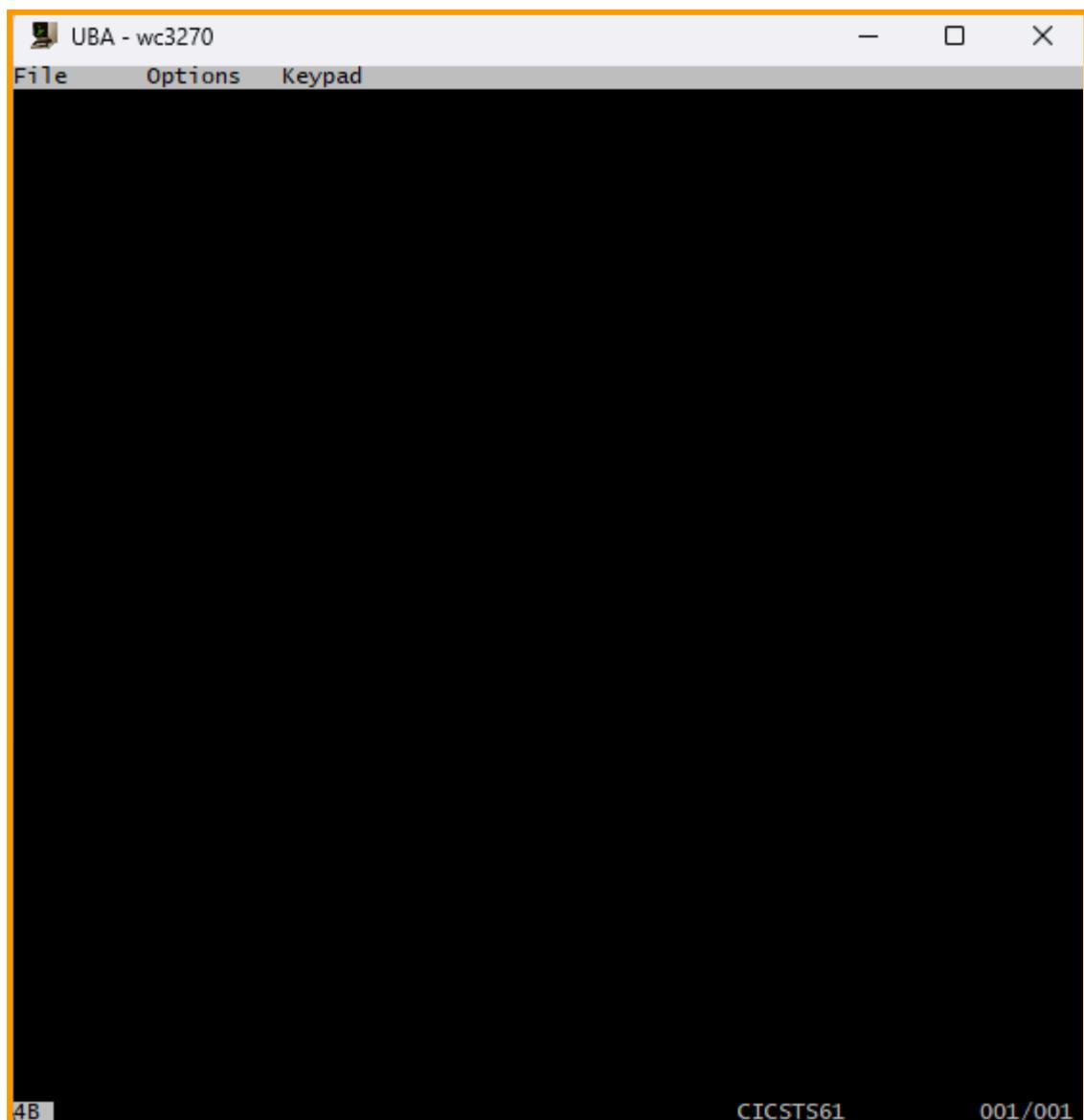


PRESIONAR 'ENTER'



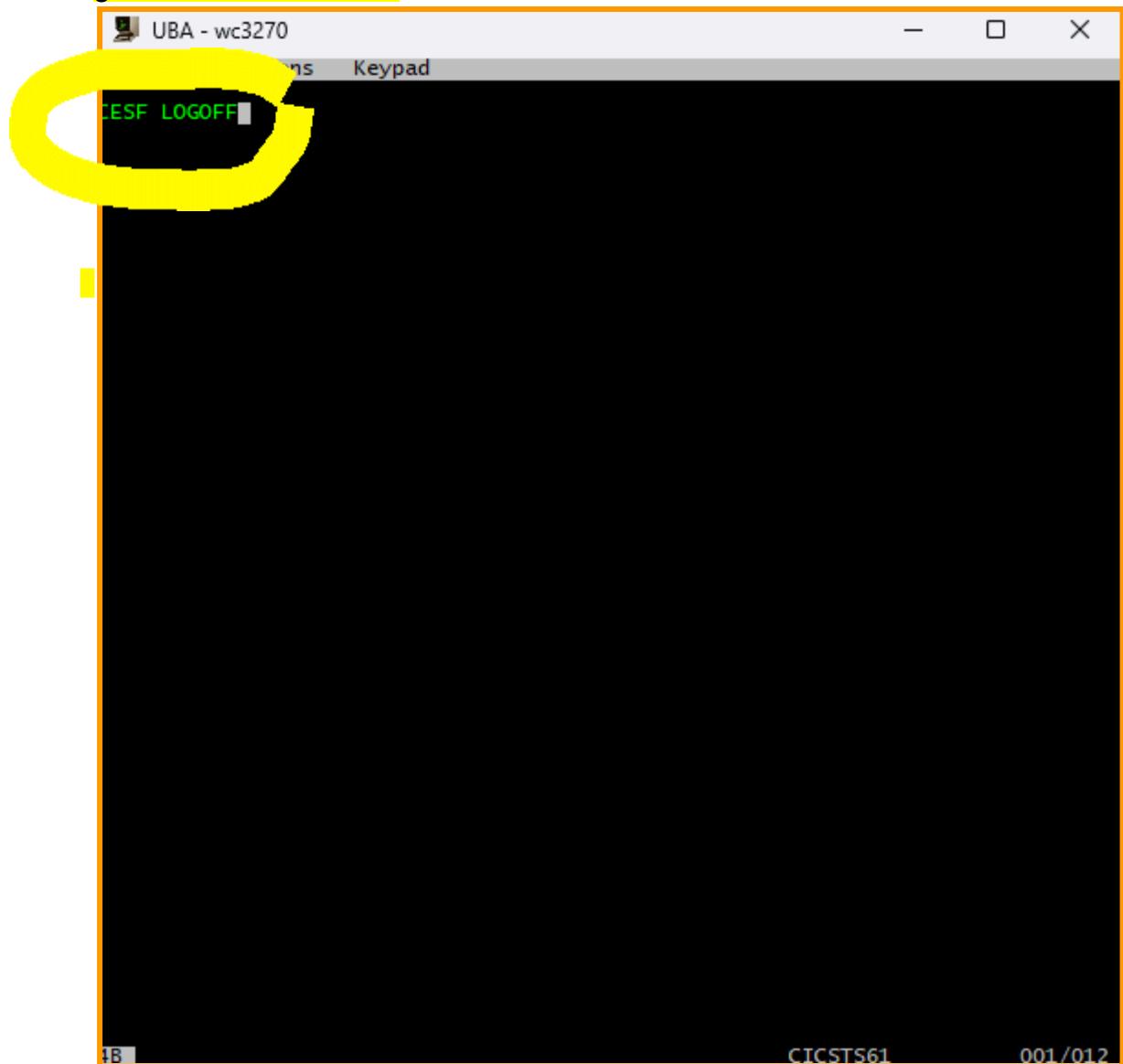
LIMPIAR PANTALLA CON 'CLEAR' (dentro de opción KEYPAD)





Recién en este espacio podrás ingresar la transacción que quieras arrancar.

¿ Cómo salir de CICS ?: CESF LOGOFF



PRESIONAR 'ENTER' y sale de la Section

COMANDOS DE CICS

CICS ofrece una amplia gama de **comandos propios** que permiten a los programas interactuar con recursos del sistema, gestionar transacciones y controlar el flujo de ejecución. Estos comandos se escriben generalmente con la sintaxis EXEC CICS ... END-EXEC y se utilizan en lenguajes como COBOL, PL/I o C.

Aquí te dejo una lista de algunos comandos clave:

- **SEND / RECEIVE**: para mostrar datos en pantalla o recibir entradas del usuario.
- **READ / WRITE / DELETE**: para interactuar con archivos (por ejemplo, VSAM).
- **LINK / XCTL**: para transferir el control entre programas.
- **RETURN**: finaliza una transacción y devuelve el control al sistema.
- **ABEND**: termina una tarea de forma anormal, útil para manejo de errores.
- **ADDRESS / ASSIGN**: acceden a áreas de memoria o información del entorno.
- **INQUIRE / SET**: permiten consultar o modificar atributos de recursos como archivos, programas o terminales.
-

Además, existen **transacciones administrativas (PROPIAS DE CICS)** como:

- **CECI**: (**CICS EXECUTE COMMAND INTERPRETER**) para probar comandos CICS directamente desde una terminal.
- **CEDA**: para definir recursos (programas, archivos, colas, etc.).
- **CEMT**: para gestionar recursos en tiempo de ejecución (por ejemplo, hacer un **newcopy** de un programa).

En general se deberá observar que, toda transacción que comienza con la letra '**C**' está reservada para uso exclusivo de CICS.

¿QUÉ ES EL NEWCOPY EN CICS?

El comando **NEWCOPY** en CICS se utiliza para **recargar una versión actualizada de un programa** en memoria sin necesidad de reiniciar la región de CICS. Es especialmente útil cuando se ha recompilado un programa y se quiere que CICS utilice la nueva versión inmediatamente.

Cuando se ejecuta un **CEMT SET PROGRAM(nombre) NEWCOPY**, CICS:

- Marca el programa como “no residente” en su tabla interna.
- La próxima vez que se invoque ese programa, lo cargará desde la biblioteca de carga (**loadlib**), trayendo así la versión más reciente.
- Si el programa está en uso (por ejemplo, en una sesión activa), la nueva copia se usará solo cuando finalicen esas sesiones.

Vamos con un ejemplo paso a paso de cómo hacer un **NEWCOPY** en CICS para actualizar un programa sin reiniciar la región:

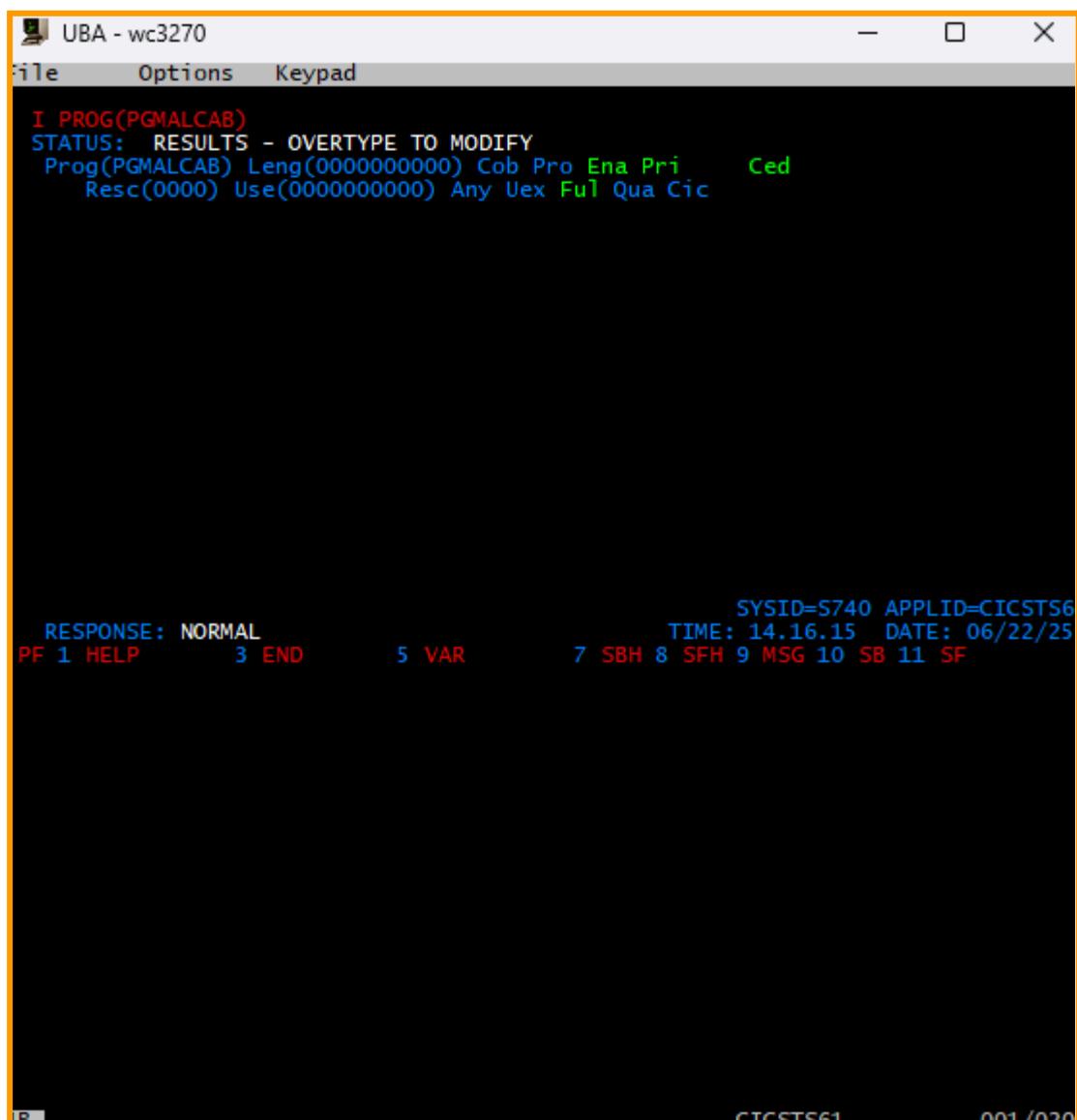
Supuestos

- Se tiene un programa llamado **PGMALCAB** que fue recompilado y cargado en la **loadlib** correspondiente.
- A efectos que CICS use esta nueva versión inmediatamente; se deberá seguir el siguiente:

Paso a paso

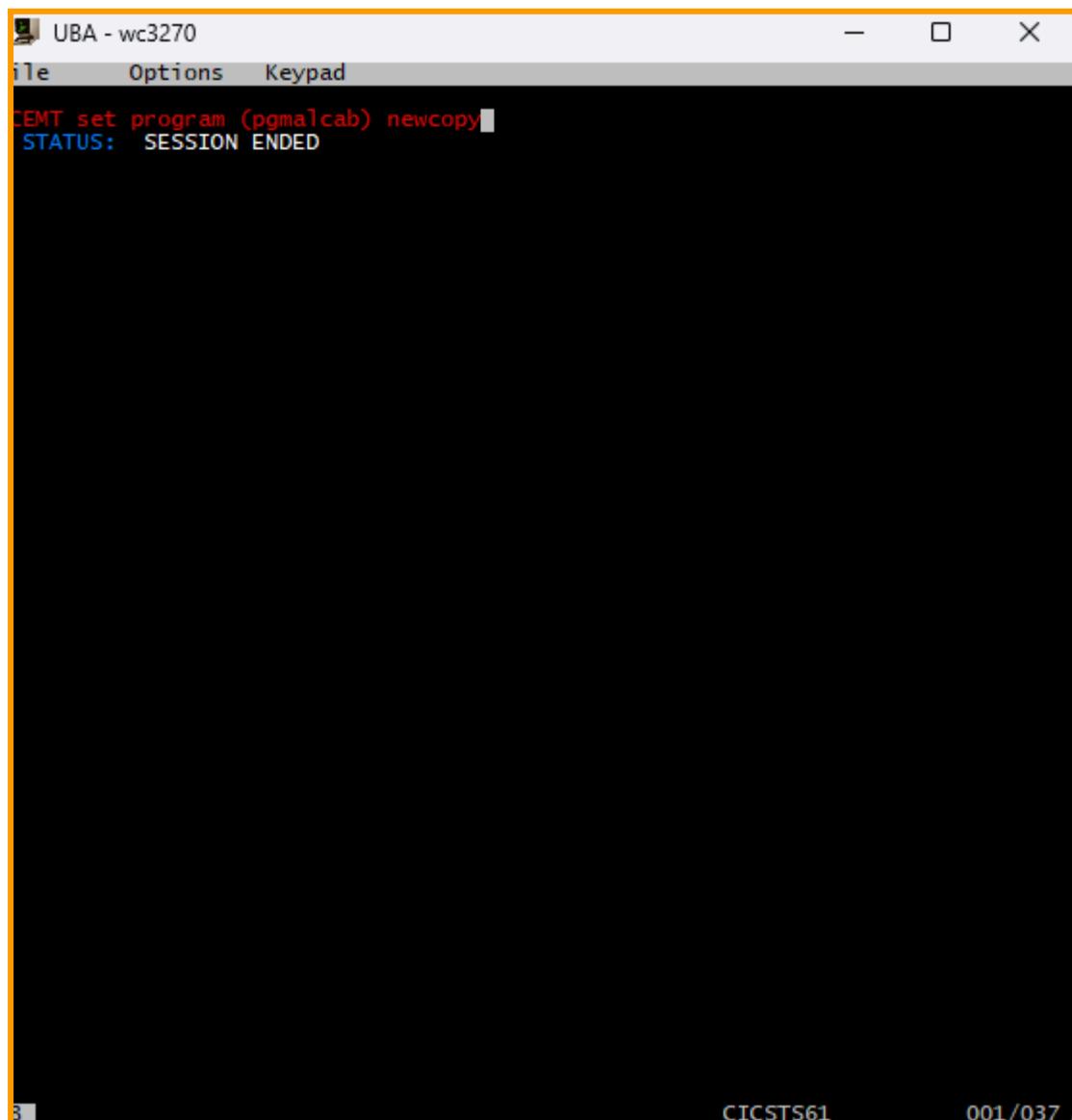
1. Verificar que el programa está definido en CICS

mediante comando CEMT:



Se escribirá NEW al final de la línea.

O ingresando directamente mediante comando SET:



Luego de presionar 'ENTER' en ambos casos, indicará a CICS que la próxima vez que se invoque **PGMALCAB**, lo cargue desde la biblioteca de carga (*LOADLIB*).

SIEMPRE verificar que el NEWCOPY fue exitoso; o sea que el largo en bytes del programa en tabla PPT se ha modificado (parámetro **Leng**).

¿CÓMO EJECUTAMOS LA COMPILACIÓN?

En entorno TSO:

IMPORTANTE - PASO PREVIO: Alocar biblioteca **KC03XXX.CURSOS.COPYLIB**

con la misma estructura que **KC02788.ALU9999.COPYLIB**

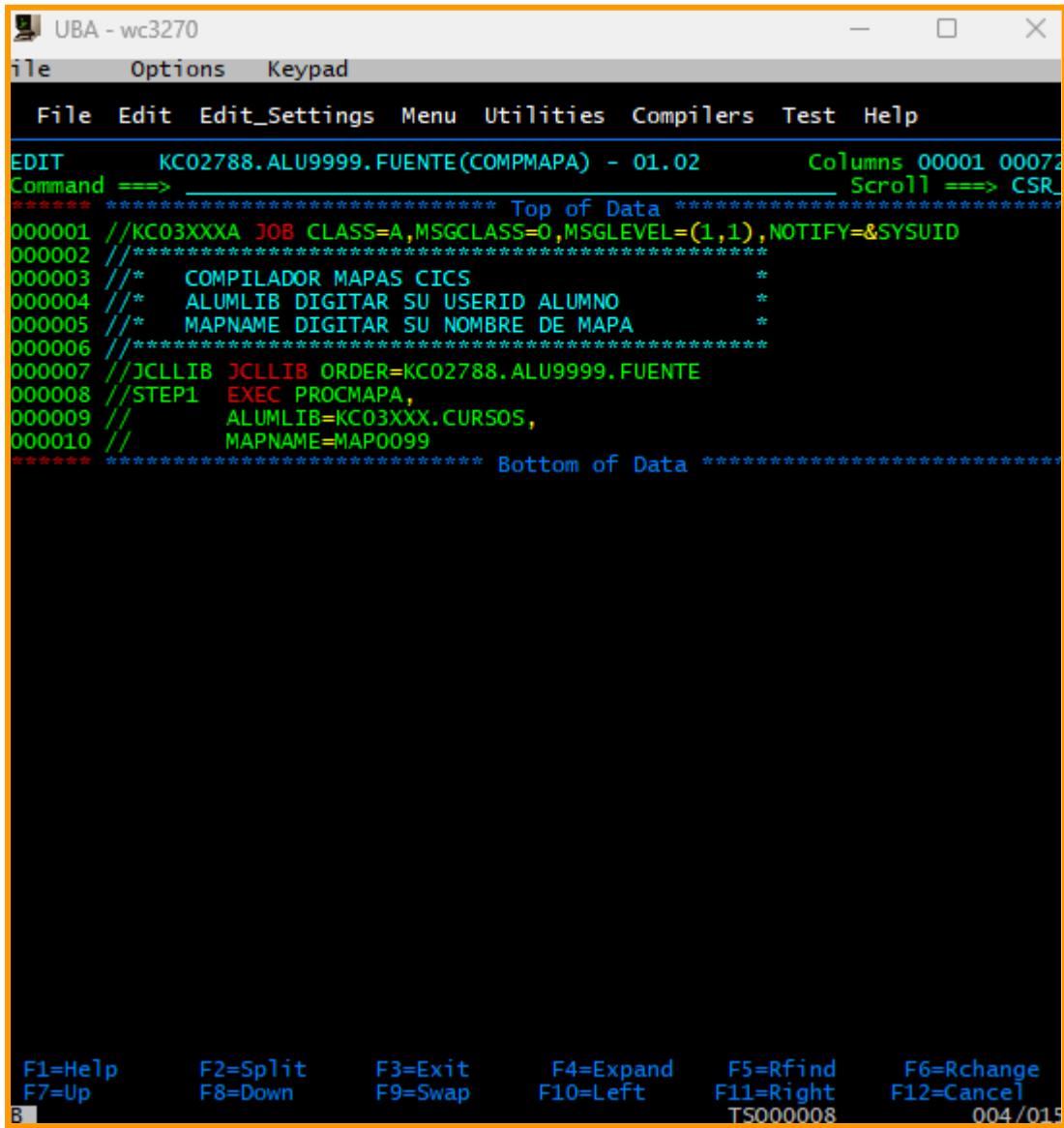
a efectos que el **COPY COBOL** de cada **MAPA** compilado quede disponible en dicha biblioteca para que, posteriormente, lo pueda leer la compilación del programa que utilice dicho mapa.

1) Compilar el mapa asociado a la transacción:

- a) KC02788.ALU9999.FUENTE(COMPMAPA)
KC03XXX.CURSOS.FUENTE

copiarlo

a



The screenshot shows a terminal window titled "UBA - wc3270". The window has a menu bar with "File", "Options", and "Keypad". Below the menu is a toolbar with "File", "Edit", "Edit_Settings", "Menu", "Utilities", "Compilers", "Test", and "Help". The main area displays JCL code for compilation:

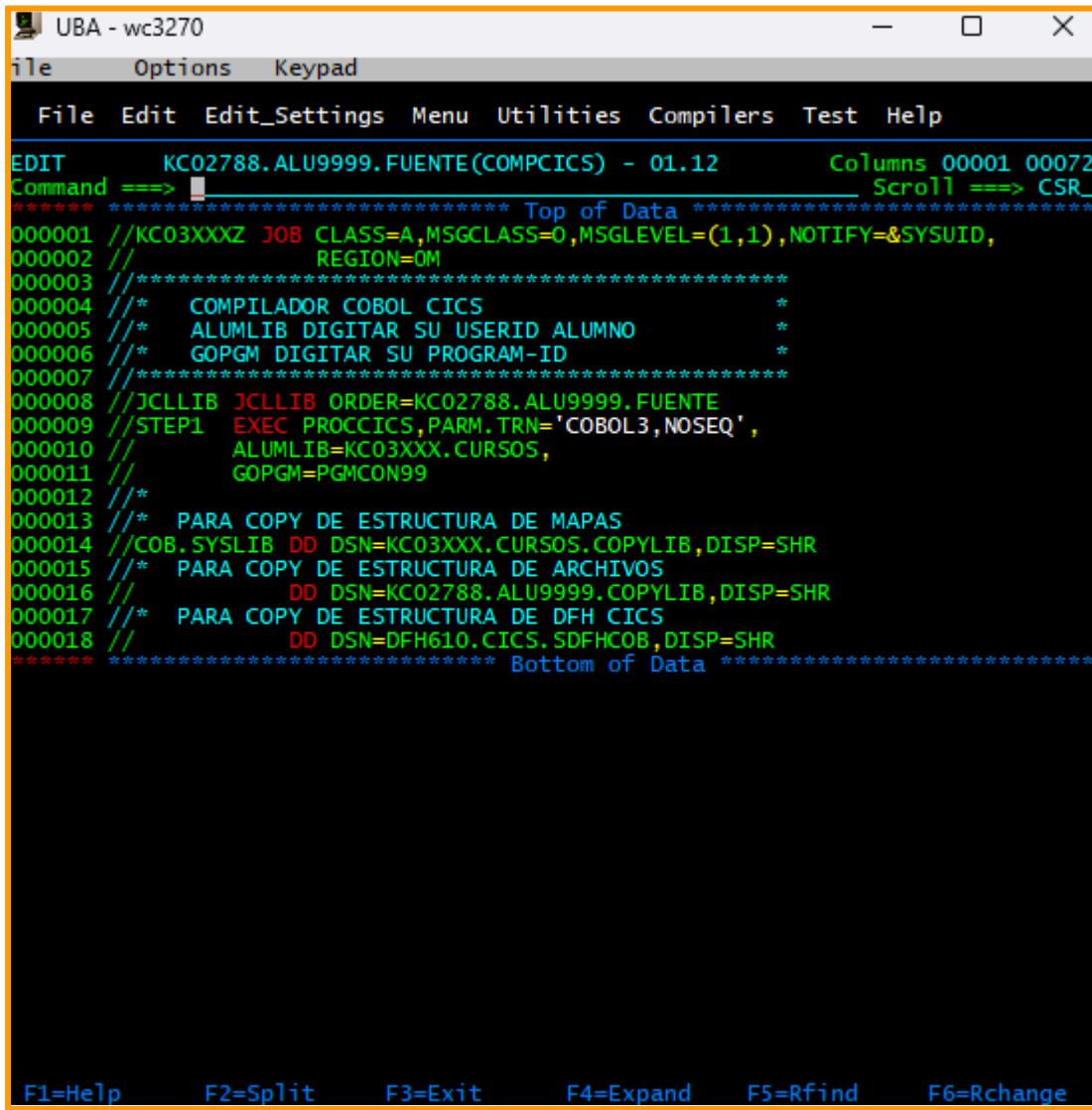
```
EDIT      KC02788.ALU9999.FUENTE(COMPMAPA) - 01.02      Columns 00001 00072
Command ==> _____
***** ***** Top of Data *****
000001 //KC03XXXA JOB CLASS=A,MSGCLASS=0,MSGLEVEL=(1,1),NOTIFY=&SYSUID
000002 //*****
000003 /* COMPILADOR MAPAS CICS *
000004 /* ALUMLIB DIGITAR SU USERID ALUMNO *
000005 /* MAPNAME DIGITAR SU NOMBRE DE MAPA *
000006 //*****
000007 //JCLLIB JCLLIB ORDER=KC02788.ALU9999.FUENTE
000008 //STEP1 EXEC PROCMAPA,
000009 //      ALUMLIB=KC03XXX.CURSOS,
000010 //      MAPNAME=MAP0099
***** ***** Bottom of Data *****
```

At the bottom of the window, there is a status bar with function key definitions and system information:

F1=Help F2=Split F3=Exit F4=Expand F5=Rfind F6=Rchange
F7=Up F8=Down F9=Swap F10=Left F11=Right F12=Cancel
TS000008 004/013

2) Compilar el programa asociado a la transacción:

- a) KC02788.ALU9999.FUENTE(COMPCICS) copiarlo a KC03XXX.CURSOS.FUENTE



The screenshot shows a window titled "UBA - wc3270" containing a COBOL job stream. The window has a menu bar with "File", "Options", "Keypad", "File", "Edit", "Edit_Settings", "Menu", "Utilities", "Compilers", "Test", and "Help". The main area displays the following job stream:

```
EDIT      KC02788.ALU9999.FUENTE(COMPCICS) - 01.12      Columns 00001 00072
Command ==> █
***** ***** Top of Data *****
000001 //KC03XXXZ JOB CLASS=A,MSGCLASS=0,MSGLEVEL=(1,1),NOTIFY=&SYSUID,
000002 //          REGION=0M
000003 //*****
000004 /*    COMPILADOR COBOL CICS
000005 /*    ALUMLIB DIGITAR SU USERID ALUMNO
000006 /*    GOPGM DIGITAR SU PROGRAM-ID
000007 //*****
000008 //JCLLIB JCLLIB ORDER=KC02788.ALU9999.FUENTE
000009 //STEP1 EXEC PROCCICS,PARM,TRN='COBOL3,NOSEQ',
000010 //          ALUMLIB=KC03XXX.CURSOS,
000011 //          GOPGM=PGMC0N99
000012 /*
000013 /*    PARA COPY DE ESTRUCTURA DE MAPAS
000014 //COB,SYSLIB DD DSN=KC03XXX.CURSOS.COPYLIB,DISP=SHR
000015 /*    PARA COPY DE ESTRUCTURA DE ARCHIVOS
000016 //          DD DSN=KC02788.ALU9999.COPYLIB,DISP=SHR
000017 /*    PARA COPY DE ESTRUCTURA DE DFH CICS
000018 //          DD DSN=DFH610.CICS.SDFHC0B,DISP=SHR
***** ***** Bottom of Data *****
```

At the bottom of the window, there is a status bar with function key labels: F1=Help, F2=Split, F3=Exit, F4=Expand, F5=Rfind, and F6=Rchange.

En entorno CICSTS61:

Antes de disparar la transacción asociada a la funcionalidad que se quiere testear;
RECORDATORIO: NEW COPY del MAPA y del PROGRAMA

Luego se tipea la transacción + '**ENTER**' y comienza la navegabilidad y el testeo.

OBJETIVO

- Repasar paso a paso los principales COMANDOS de código COBOL CICS a efectos que el estudiante pueda encarar su primer desafío de código COBOL pseudo-conversacional.

ESPECIFICACIONES

Dentro del entorno TSO se construirá el código COBOL CICS correspondiente; de la misma manera que el código COBOL BATCH.

La diferencia será:

- 1) el programa **COMPILEADOR** que se estará utilizando. Lo podrá ver en el siguiente componente:
 - a) KC02788.ALU9999.FUENTE(**COMPMAPA**) Para compilar el mapa asociado
 - b) KC02788.ALU9999.FUENTE(**COMPCICS**) Para compilar el código fuente
- 2) **NEWCOPY** dentro del entorno CICS para tomar el último objeto generado con la compilación EXITOSA.

¿Cómo arrancar el código fuente?

Dado que los archivos están predefinidos en tabla propia de CICS: **FCT (File Control Table)**.

El código se comenzará a escribir a partir de la **WORKING STORAGE SECTION**:

UBA - wc3270

File Options Keypad

File Edit Edit_Settings Menu Utilities Compilers Test Help

EDIT KC02788.CURSOS.FUENTE(PGMCON99) - 01.00 Columns 00001 00072
Command ==> ***** Top of Data *****

```

000001 IDENTIFICATION DIVISION.
000002 PROGRAM-ID. PGMCON99.
000003 DATA DIVISION.
000004 FILE SECTION.
000005 WORKING-STORAGE SECTION.
000006
000007 01 CT-CONSTANTES.
000008    03 CT-MSG0.
000009      05 CT-MNS-01          PIC X(72) VALUE
000010          'INGRESE LOS DATOS Y PRESIONE ENTER'.
000011      05 CT-MNS-02          PIC X(72) VALUE
000012          'DATOS INGRESADOS INCORRECTOS - REINGRESE'.
000013      05 CT-MNS-03          PIC X(72) VALUE
000014          'TIPO Y NÚMERO DOCUMENTO INEXISTENTES - REINGRESE'.
000015      05 CT-MNS-04          PIC X(72) VALUE
000016          'TIPO DE DOCUMENTO INVALIDO'.
000017      05 CT-MNS-05          PIC X(72) VALUE
000018          'NUMERO DE DOCUMENTO INVALIDO'.
000019      05 CT-MNS-06          PIC X(72) VALUE 'CLIENTE ENCONTRADO'.
000020      05 CT-MNS-08          PIC X(72) VALUE
000021          'PROBLEMA CON ARCHIVO PERSONA'.
000022      05 CT-MNS-09          PIC X(72) VALUE 'TECLA INVALIDA'.
000023      05 CT-MNS-EXIT        PIC X(72) VALUE
000024          'FIN TRANSACCION T199'.
000025      03 CT-DATASET         PIC X(08) VALUE 'CLIENTE'.
000026      03 CT-DATASET-LEN     PIC S9(04) COMP VALUE 50.
000027
000028 01 WS-VARIABLES.
000029    03 WS-MAP-00           PIC X(07)      VALUE 'MAP0099'.
000030    03 WS-MAPSET-00        PIC X(07)      VALUE 'MAP0099'.
000031    03 WS-MAP-01           PIC X(07)      VALUE 'MAP0199'.
000032    03 WS-MAPSET-01        PIC X(07)      VALUE 'MAP0199'.
000033    03 WS-LONG             PIC S9(04) COMP.
000034    03 WS-ABSTIME          PIC S9(16) COMP VALUE +0.
000035    03 WS-FECHA            PIC X(10)      VALUE SPACES.
```

F1=Help F2=Split F3=Exit F4=Expand F5=Rfind F6=Rchange
F7=Up F8=Down F9=Swap F10=Left F11=Right F12=Cancel

Dentro de la WORKING STORAGE SECTION; también se estarán definiendo:

- 1) **Nombre del mapa** que se utilizará dentro de la funcionalidad: MAPXXXX
- 2) Área de comunicación entre el mapa y la transacción: DFHBMSCA
- 3) Área de control de teclas de función: DFHAID
- 4) Copy de estructura de archivo que se está utilizando

También se estará definiendo la LINKAGE SECTION; dentro de la DATA DIVISION. Para permitir la navegabilidad.

El tamaño de la **DFHCOMMAREA** dependerá de la cantidad de bytes que viajen a la siguiente transacción.

UBA - wc3270

File Options Keypad

File Edit Edit_Settings Menu Utilities Compilers Test Help

```

EDIT      KC02788.CURSOS.FUENTE(PGMCON99) - 01.00          Columns 00001 00072
Command ==> _____
000036      03 WS-SEP-DATE          PIC X                 VALUE '/'.
000037      03 WS-HORA             PIC X(08)            VALUE SPACES.
000038      03 WS-SEP-HOUR         PIC X                 VALUE ':'.
000039      03 WS-RESP             PIC S9(04) COMP.
000040      03 SW-CONFIRMAR        PIC X VALUE 'Y'.
000041      03 WS-NORMAL           PIC X VALUE '*'.
000042      03 WS-ENTER            PIC X VALUE ''.
000043
000044      01 WS-COMMAREA.
000045          03 WS-USER-DATA.
000046              05 WS-USER-TIPDOC   PIC X(02).
000047              05 WS-USER-NRODOC  PIC 9(11).
000048          03 WS-TIP-DOC       PIC X(02).
000049              88 WS-TIP-DOC-BOOLEAN    VALUE 'DU'
000050                                'PA'
000051                                'PE'.
000052          03 FILLER            PIC X(5).
000053
000054      COPY MAP0099.
000055      COPY DFHBMSCA.
000056      COPY DFHAID.
000057      COPY CPCLIE.
000058
000059      LINKAGE SECTION.
000060
000061      01 DFHCOMMAREA PIC X(20).
000062
000063      PROCEDURE DIVISION.
000064      MAIN-PROGRAM.
000065
000066      PERFORM 1000-I-INICIO THRU 1000-F-INICIO.
000067
000068      PERFORM 2000-I-PROCESO THRU 2000-F-PROCESO.
000069
000070      PERFORM 9999-I-FINAL THRU 9999-F-FINAL.
000071
F1=Help     F2=Split      F3=Exit       F4=Expand      F5=Rfind      F6=Rchange
F7=Up       F8=Down       F9=Swap       F10=Left      F11=Right     F12=Cancel

```

Como se observa en la figura anterior, la estructura principal del programa continúa siendo idéntica a la expresada en un programa COBOL BATCH.

UBA - wc3270

File Options Keypad

File Edit Edit_Settings Menu Utilities Compilers Test Help

EDIT KC02788.CURSOS.FUENTE(PGMCON99) - 01.00 Columns 00001 00072
Command ==> Scroll ==> CSR_

```

000059      LINKAGE SECTION.
000060
000061      01 DFHCOMMAREA PIC X(20).
000062
000063      PROCEDURE DIVISION.
000064      MAIN-PROGRAM.
000065
000066          PERFORM 1000-I-INICIO THRU 1000-F-INICIO.
000067
000068          PERFORM 2000-I-PROCESO THRU 2000-F-PROCESO.
000069
000070          PERFORM 9999-I-FINAL THRU 9999-F-FINAL.
000071
000072      1000-I-INICIO.
000073
000074      MOVE DFHCOMMAREA TO WS-COMMAREA.
000075
000076      IF EIBCALEN = 0
000077          MOVE LENGTH OF MAP00990 TO WS-LONG
000078          INITIALIZE MAP00990
000079          PERFORM 7000-I-TIME THRU 7000-F-TIME
000080          MOVE CT-MNS-01 TO MSG0
000081          EXEC CICS
000082              SEND MAP (WS-MAP-00)
000083              MAPSET (WS-MAPSET-00)
000084              FROM (MAP00990)
000085              LENGTH (WS-LONG)
000086              ERASE
000087              FREEKB
000088          END-EXEC.
000089      1000-F-INICIO. EXIT.
000090
000091      2000-I-PROCESO.
000092          PERFORM 7000-I-TIME THRU 7000-F-TIME
000093          MOVE LENGTH OF MAP00990 TO WS-LONG
000094          EXEC CICS

```

F1=Help F2=Split F3=Exit F4=Expand F5=Rfind F6=Rchange
F7=Up F8=Down F9=Swap F10=Left F11=Right F12=Cancel

En la figura anterior se observa la modalidad de estructura del párrafo INICIO:

- 1) Se reserva la DFHCOMMAREA en la WORKING STORAGE (WS-COMMAREA) de **IDÉNTICA ESTRUCTURA DE DATOS**
- 2) Se consulta el parámetro **EIBCALEN** para saber si es la primera vez que se ingresa:
 - a) **EIBCALEN = 0** es la primera vez
- 3) Se arma el **MAPA** y se envía con la sentencia **SEND**; como se ve en la figura anterior (siempre anteponiendo **EXEC CICS** y finalizando con **END-EXEC**):

```

EXEC CICS
    SEND MAP.....
END-EXEC.

```

UBA - wc3270

File Options Keypad

File Edit Edit_Settings Menu Utilities Compilers Test Help

EDIT KC02788.CURSOS.FUENTE(PGMCON99) - 01.00 Columns 00001 0007
 Command ==> 1000-I-INICIO.

```

000072      1000-I-INICIO.
000073
000074      MOVE DFHCOMMAREA TO WS-COMMAREA.
000075
000076      IF EIBCALEN = 0
000077          MOVE LENGTH OF MAP00990 TO WS-LONG
000078          INITIALIZE MAP00990
000079          PERFORM 7000-I-TIME THRU 7000-F-TIME
000080          MOVE CT-MNS-01 TO MSG0
000081          EXEC CICS
000082              SEND MAP (WS-MAP-00)
000083                  MAPSET (WS-MAPSET-00)
000084                      FROM (MAP00990)
000085                      LENGTH (WS-LONG)
000086                      ERASE
000087                      FREEKB
000088          END-EXEC.
000089      1000-F-INICIO. EXIT.
000090
000091      2000-I-PROCESO.
000092          PERFORM 7000-I-TIME THRU 7000-F-TIME
000093          MOVE LENGTH OF MAP00990 TO WS-LONG
000094          EXEC CICS
000095              RECEIVE MAP (WS-MAP-00)
000096                  MAPSET (WS-MAPSET-00)
000097                      INTO (MAP0099I)
000098                      RESP (WS-RESP)
000099          END-EXEC
000100          PERFORM 2500-I-PULSAR-TECLA THRU 2500-F-PULSAR-TECLA.
000101
000102      2000-F-PROCESO. EXIT.
000103
000104
000105
000106      2500-I-PULSAR-TECLA.
000107          EVALUATE EIBAID
  
```

F1=Help F2=Split F3=Exit F4=Expand F5=Rfind F6=Rchange
 F7=Up F8=Down F9=Swap F10=Left F11=Right F12=Cancel

En la figura anterior se observa el párrafo 2000-I-PROCESO.

Donde se recibe el MAPA con la sentencia RECEIVE MAP y se convoca al párrafo que analizará las funcionalidades según la tecla de función utilizada por el usuario.

UBA - wc3270

File Options Keypad

File Edit Edit_Settings Menu Utilities Compilers Test Help

EDIT KC02788.CURSOS.FUENTE(PGMCON99) - 01.00 Columns 00001 00072
 Command ==> Scroll ==> CSR_

```

000105
000106      2500-I-PULSAR-TECLA.
000107          EVALUATE EIBAID
000108
000109          WHEN DFHENTER
000110              PERFORM 3000-I-ENTER THRU 3000-F-ENTER
000111
000112          WHEN DFHPF3
000113              PERFORM 3500-I-PF3 THRU 3500-F-PF3
000114
000115          WHEN DFHPF12
000116              PERFORM 9000-I-PF12 THRU 9000-F-PF12
000117          WHEN OTHER
000118              MOVE CT-MNS-09 TO MSG0
000119                  PERFORM 7000-I-TIME THRU 7000-F-TIME
000120          EXEC CICS
000121              SEND MAP      (WS-MAP-00)
000122                  MAPSET (WS-MAPSET-00)
000123                      FROM (MAP00990)
000124                          LENGTH (WS-LONG)
000125                      ERASE
000126          END-EXEC
000127      END-EVALUATE.
000128
000129          2500-F-PULSAR-TECLA. EXIT.
000130
000131          3000-I-ENTER.
000132              MOVE TIPDOCI TO WS-TIP-DOC.
000133              IF NOT WS-TIP-DOC-BOOLEAN
000134                  INITIALIZE MAP00990
000135                  MOVE CT-MNS-04 TO MSG0
000136              ELSE
000137                  IF NUMDOCI NOT NUMERIC
000138                      INITIALIZE MAP00990
000139                      MOVE CT-MNS-05 TO MSG0
000140              ELSE
  F1=Help      F2=Split    F3=Exit      F4=Expand    F5=Rfind    F6=Rchange
  F7=Up       F8=Down     F9=Swap      F10=Left     F11=Right   F12=Cancel

```

Luego de construida la funcionalidad; el párrafo final es:

UBA - wc3270

File Options Keypad

File Edit Edit_Settings Menu Utilities Compilers Test Help

EDIT KC02788.CURSOS.FUENTE(PGMCON99) - 01.00 Columns 00001 00072

Command ==> FROM (CT-MNS-EXIT)

```

000213          END-EXEC
000214
000215
000216          EXEC CICS
000217          RETURN
000218          END-EXEC.
000219
000220          9000-F-PF12. EXIT.
000221
000222          9999-I-FINAL.
000223
000224          EXEC CICS
000225          RETURN
000226          TRANSID ('T199')
000227          COMMAREA (WS-COMMAREA)
000228          END-EXEC.
000229
000230          9999-F-FINAL. EXIT.
000231
***** ***** Bottom of Data *****

```

F1=Help F2=Split F3=Exit F4=Expand F5=Rfind F6=Rchange

F7=Up F8=Down F9=Swap F10=Left F11=Right F12=Cancel

pasándole el control al CICS mediante:

```

EXEC CICS
  RETURN
END-EXEC.

```

O, como en la figura anterior; retornando a la transacción que la invocó inicialmente conjuntamente con la **COMMAREA** correspondiente y volver a comenzar el ciclo:

```

EXEC CICS
  RETURN TRANSID(T199)
  COMMAREA(WS-COMMAREA)
END-EXEC.

```

OBJETIVO

- Construir un código COBOL CICS para enviar un mensaje desde código COBOL CICS
- Al finalizar esta práctica, el estudiante habrá estado practicando la codificación de un programa pseudo conversacional COBOL CICS

ESPECIFICACIONES

PROGRAMA CICS - SEND

Construir un programa CICS para enviar un mensaje al usuario:

HOLA CICS!!!!

NOMBRE de la transacción: EXXX

NOMBRE del programa: PGMBAXXX

donde xxx = últimas 3 letras y/o números de user id

¿Cómo se envía mensaje a la pantalla CICS?

Con la sentencia CICS:

EXEC CICS

SEND TEXT

```
FROM (mmmmm)
```

```
LENGTH(longitud)
```

```
END-EXEC
```

donde mmmmm = mensaje que se desea enviar al usuario

longitud (PIC S9(04) COMP.) = longitud, en bytes, del mensaje
que se envía al usuario

Podrán ver la resolución en: KC02788.ALU9999.FUENTE(**PGMSED1F**)

OBJETIVO

- Conocer las transacciones más utilizadas de COBOL CICS a efectos que el estudiante pueda encarar su primer desafío de código COBOL pseudo-conversacional con la invaluable ayuda de ellas..

ESPECIFICACIONES

Estas transacciones CICS son comandos muy útiles para diagnosticar, administrar y probar aplicaciones dentro de un entorno CICS. A continuación se explican una por una con ejemplos prácticos:



CECI — Command Interpreter (Intérprete de comandos)

Sirve para **probar** comandos CICS sin que se ejecuten realmente dentro del programa.

- **¿Para qué se usa?** Simular el comportamiento de una instrucción CICS.

Ejemplo:

CECI SEND TEXT FROM('Hola Alicia') ERASE

- Esto no envía realmente el mensaje, pero te muestra cómo se ejecutaría.
-



CEDA — Define Abilities (Definición de recursos)

Permite **ver, definir y modificar** recursos CICS, como programas, mapas, transacciones, etc.

- **¿Para qué se usa?** Registrar o modificar atributos de un recurso.

Ejemplo:

CEDA DEFINE PROGRAM(MIPROG01)

- Esto define un nuevo programa llamado MIPROG01.
-



CEMT — Master Terminal (Administración en tiempo real)

Usado para **inspeccionar y controlar** recursos en tiempo de ejecución (muy útil para ver si algo está activo).

- **¿Para qué se usa?** Activar, desactivar, ver el estado de recursos.

Ejemplo:

CEMT INQUIRE PROGRAM(MIPROG01)

Muestra el estado actual del programa: si está en uso, cargado, etc.

CEMT SET PROGRAM(MIPROG01) NEW

- Fuerza la recarga del programa en memoria.
-

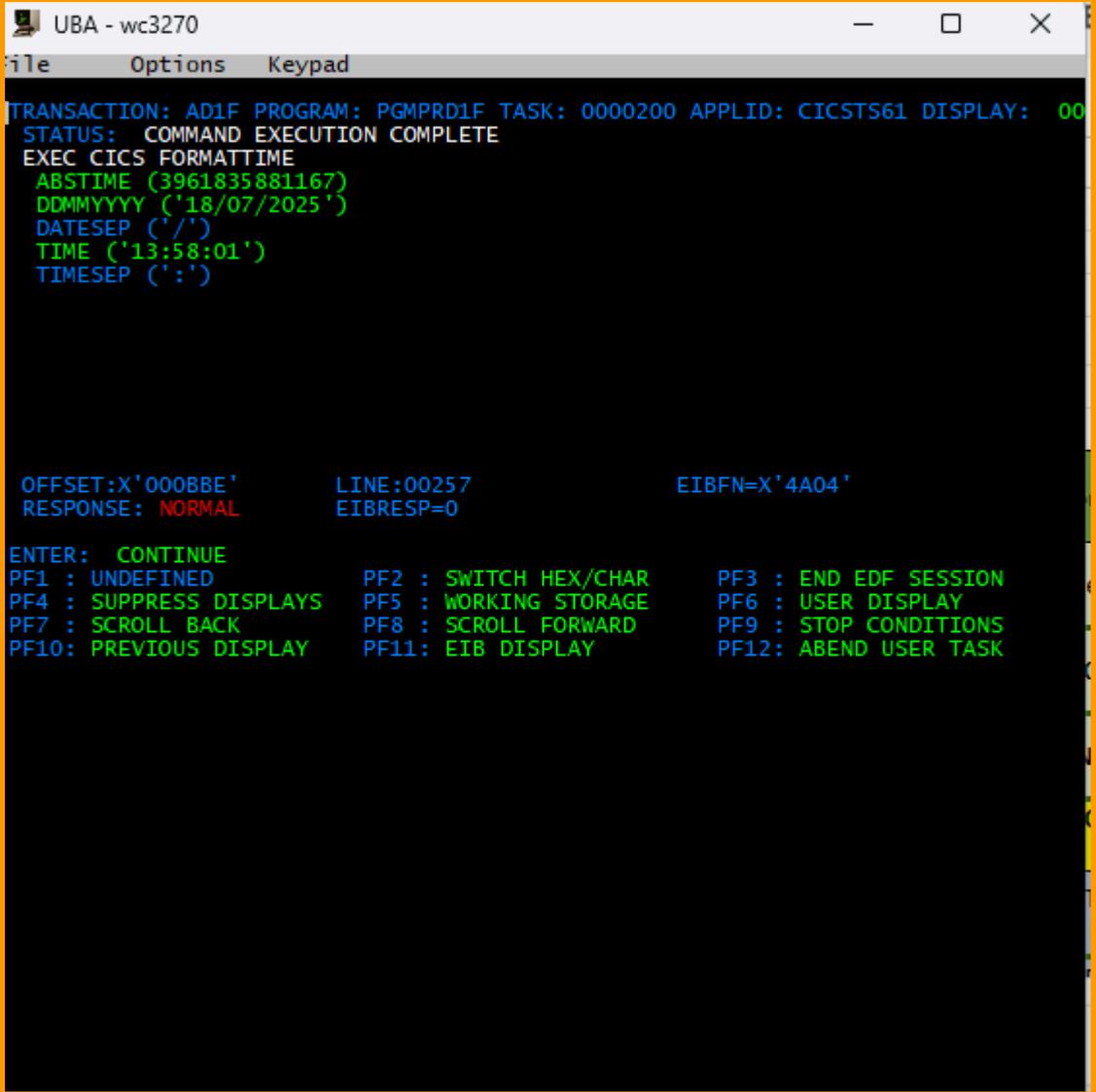


CEDF — Debug Facility (Depuración)

Permite **depurar** aplicaciones COBOL en CICS paso a paso.

- **¿Para qué se usa?** Detectar errores y analizar el flujo de ejecución de tu programa.
 - **Ejemplo:**
Activar CEDF y luego ejecutar una transacción. CICS irá mostrando cada paso del código CICS que se ejecuta.
-

Se muestra a continuación **CEDF de transacción de usuario AD1F**:



The screenshot shows a terminal window titled "UBA - wc3270". The menu bar includes "File", "Options", and "Keypad". The main display area shows the following text:

```
TRANSACTION: AD1F PROGRAM: PGMPRD1F TASK: 0000200 APPLID: CICSTS61 DISPLAY: 00
STATUS: COMMAND EXECUTION COMPLETE
EXEC CICS FORMATTIME
ABSTIME (3961835881167)
DDMMYYYY ('18/07/2025')
DATESEP ('/')
TIME ('13:58:01')
TIMESEP (':')

OFFSET:X'000BBE'      LINE:00257          EIBFN=X'4A04'
RESPONSE: NORMAL      EIBRESP=0

ENTER: CONTINUE
PF1 : UNDEFINED        PF2 : SWITCH HEX/CHAR    PF3 : END EDF SESSION
PF4 : SUPPRESS DISPLAYS PF5 : WORKING STORAGE  PF6 : USER DISPLAY
PF7 : SCROLL BACK      PF8 : SCROLL FORWARD   PF9 : STOP CONDITIONS
PF10: PREVIOUS DISPLAY PF11: EIB DISPLAY       PF12: ABEND USER TASK
```

'ENTER'

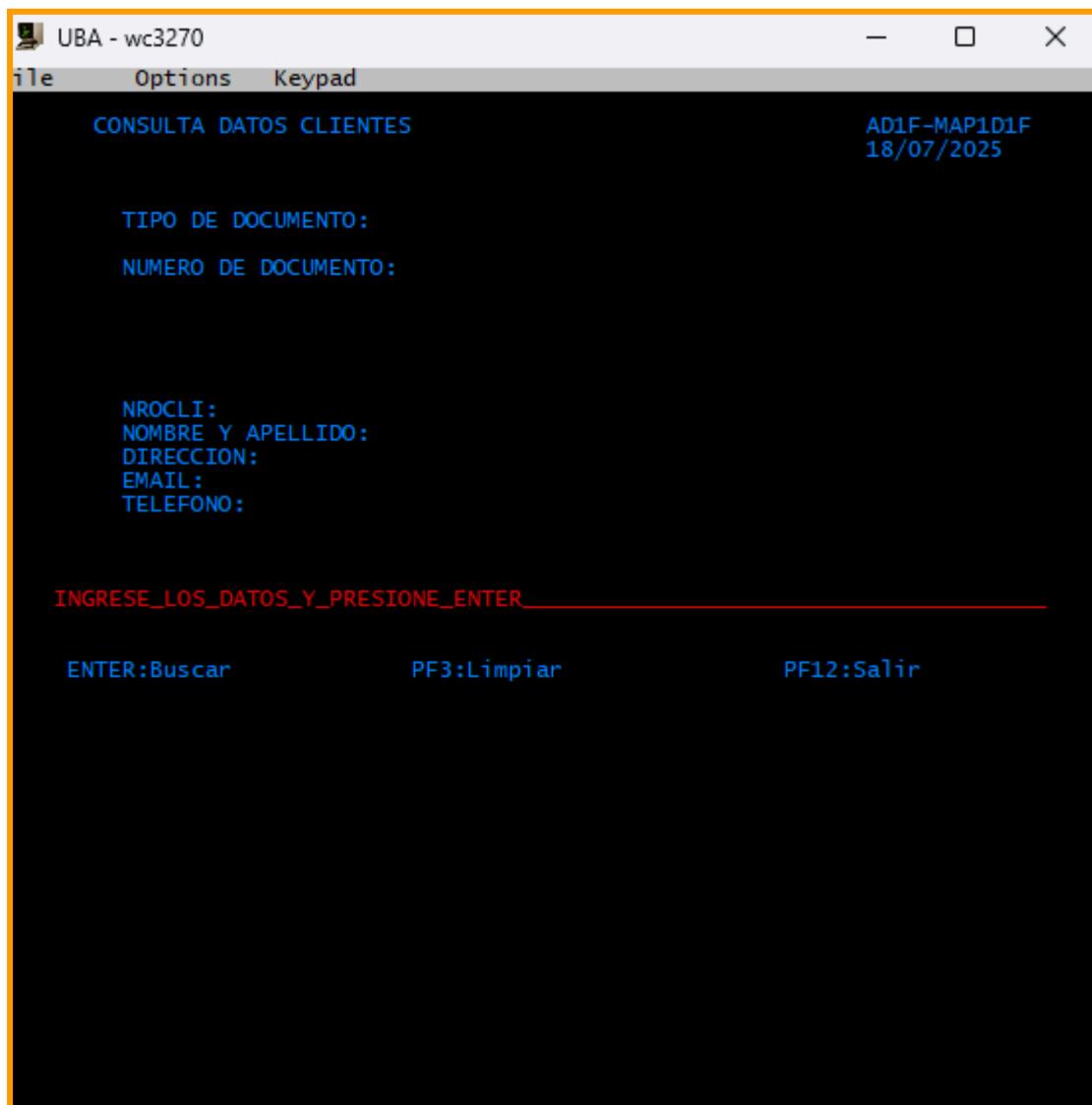
UBA - wc3270

File Options Keypad

TRANSACTION: AD1F PROGRAM: PGMPRD1F TASK: 0000200 APPLID: CICSTS61 DISPLAY: 00
STATUS: ABOUT TO EXECUTE COMMAND
EXEC CICS SEND MAP
MAP ('MAP1D1F')
FROM ('.....18/07/2025.....')
LENGTH (242)
MAPSET ('MAP1D1F')
TERMINAL
FREEKB
ERASE

OFFSET:X'00035C' LINE:00086 EIBFN=X'1804'

ENTER: CONTINUE
PF1 : UNDEFINED PF2 : SWITCH HEX/CHAR PF3 : UNDEFINED
PF4 : SUPPRESS DISPLAYS PF5 : WORKING STORAGE PF6 : USER DISPLAY
PF7 : SCROLL BACK PF8 : SCROLL FORWARD PF9 : STOP CONDITIONS
PF10: PREVIOUS DISPLAY PF11: EIB DISPLAY PF12: ABEND USER TASK



De todas estas transacciones; la más importante indudablemente, resulta ser **CECI**.

A continuación se muestra cómo usar la transacción **CECI** en CICS para simular comandos y probar su sintaxis sin afectar el entorno real.

Es muy útil para practicar COBOL + CICS y entender cómo se comportan las sentencias.

Ejemplo de uso de **CECI**

Supongamos que estás diseñando una pantalla para consultar clientes. Podés simular el envío de un texto como si fuera parte de una transacción real:

```
CECI SEND TEXT FROM('Consulta Clientes - Ingrese Número') ERASE
```

 Este comando le pediría a CICS que simule el envío del texto a la pantalla. No se ejecuta realmente, pero te muestra cómo se vería y cómo se comportaría el comando.

Otro ejemplo útil: recuperar datos de una tabla

```
CECI EXEC CICS RECEIVE INTO(CLIENTE-DATOS) LENGTH(100)
```

Esto simula la recepción de datos del usuario en el área **CLIENTE-DATOS**. Excelente para probar estructuras de datos y validaciones antes de programar definitivamente.

¿Qué más podés hacer?

Podés probar:

- **SEND MAP** para ver cómo se renderiza tu CICS Map en pantalla
 - **EXEC SQL** para simular sentencias DB2 desde COBOL dentro de CICS (si tenés la integración activa)
-

SEND MAP — Mostrar un mapa en pantalla

```
CECI SEND MAP('CONSCLIE') MAPSET('MAPSET01') FROM(CLIENTE-DATOS) ERASE
```

- 🔍 Simula el envío de tu mapa “Consulta Clientes” con los datos cargados desde la estructura **CLIENTE-DATOS**. Muy útil para ver si el mapa se despliega como esperás.
-

RECEIVE MAP — Recibir datos desde el usuario

```
CECI RECEIVE MAP('CONSCLIE') MAPSET('MAPSET01') INTO(CLIENTE-DATOS)
```

- 📌 Te permite verificar cómo se recibirán los datos ingresados en campos del mapa. Así podés ajustar los nombres de los campos o la estructura **CLIENTE-DATOS**.
-

EXEC SQL — Simulación de acceso a DB2

```
CECI EXEC SQL SELECT NOMBRE FROM CLIENTES WHERE ID = '123' END-EXEC
```

- ⚠ Esto depende de que el entorno tenga configurada la integración con DB2. Pero podés usarlo para comprobar la sintaxis de tus queries embebidas.
-

EXEC CICS DELAY — Simular espera

```
CECI EXEC CICS DELAY FOR(5)
```

- ⌚ Simula una espera de 5 segundos en tu programa. Genial para probar cómo se comportan las pausas en la lógica de negocio.
-

WRITEQ TD — Escribir en una cola transitoria

```
CECI EXEC CICS WRITEQ TD QUEUE('SALIDA') FROM(MENSAJE) LENGTH(80)
```

- 🖨 Simula el envío de datos a una cola TD, útil para registros, logs o impresiones.
-

¿CÓMO INTERPRETAR LOS CICLOS EN COBOL CICS?

OBJETIVO

- Los ciclos en COBOL dentro de un entorno CICS no se manejan como los típicos bucles **PERFORM** que se arman en COBOL puro.
- En CICS, el ciclo más importante es el **flujo de ejecución de una transacción**, que involucra varios pasos desde que el usuario interactúa con una pantalla hasta que se devuelve una respuesta.
- Cada **CICLO** representa una **UNIDAD DE TRABAJO** en CICS
- En este documento; vamos a desglosarlo de forma clara y estructurada

ESPECIFICACIONES



Ciclo (Unidad de trabajo) típico de una transacción COBOL-CICS

Este ciclo representa cómo se ejecuta una transacción en CICS usando un programa COBOL; lo desarrollaremos paso a paso:

1. Interacción con el usuario (pantalla/mapa)

- El usuario ingresa datos en una pantalla diseñada con **BMS (Basic Mapping Support)**.
- El mapa se define en assembler o con herramientas específicas.

2. Recepción de datos (EXEC CICS RECEIVE MAP**)**

- El programa COBOL usa esta sentencia para recibir los datos del mapa.

```
EXEC CICS  
  RECEIVE MAP (MAPCLI1)  MAPSET (MAPSET1)  
  END-EXEC.
```

3. Validación y lógica de negocio

- Se analiza lo ingresado, se validan campos, y se decide qué hacer (consultar en base de datos DB2 o archivo VSAM, mostrar error, etc.).

- Aquí puede haber un bucle COBOL tal como:

```
PERFORM UNTIL END-OF-FILE

EXEC CICS
    READ DATASET ('PERSONA')
        RIDFLD (WS-USER-DATA)
        INTO (REG-PERSONA)
        LENGTH (CT-DATASET-LEN)
        EQUAL
        RESP (WS-RESP)
    END-EXEC
```

```
MOVE datos TO campos-de-salida
END-PERFORM.
```

4. 📦 Envío de datos a la pantalla (**EXEC CICS SEND MAP**)

- Se devuelven los datos al usuario en el mapa:

```
EXEC CICS
    SEND (MAPCLI1) MAPSET (MAPSET1)
        FROM(WS-DATOS) ERASE
    END-EXEC.
```

5. 🔄 FINAL CICLO o UNIDAD DE TRABAJO - (**EXEC CICS RETURN TRANSID**)

- El programa termina su ejecución (liberando recursos tomados) y espera la próxima acción del usuario (es lo que llamamos PSEUDO-CONVERSACIONAL):

```
EXEC CICS
    RETURN TRANSID('BD1F')
    COMMAREA (WS-COMMAREA)
END-EXEC.
```



¿Dónde aparecen los ciclos COBOL?

Deberemos diferenciar un **bucle** de un **ciclo** en COBOL CICS.

Los bucles COBOL como `PERFORM UNTIL`, `PERFORM VARYING`, etc., se usan para:

- Recorrer resultados de una consulta SQL.
 - Procesar múltiples registros.
 - Validar múltiples campos.
 - Generar líneas en una tabla de salida.
-

Los ciclos o UNIDADES DE TRABAJO en COBOL CICS; Se resuelven en un proceso completo de una transacción; por ejemplo: [`consulta clientes`](#) y muestra los resultados en una pantalla.

Este ejemplo incluye:

- Un mapa llamado **MAPCLI1**
 - Acceso a datos en archivo VSAM KSDS
 - Uso de bucles COBOL
 - Flujo CICS completo; por CICLO o UNIDADES DE TRABAJO
-



Estructura general del programa

IDENTIFICATION DIVISION.

PROGRAM-ID. PGMPRD1F.

ENVIRONMENT DIVISION.

INPUT-OUTPUT SECTION.

DATA DIVISION.

WORKING-STORAGE SECTION.

* **Variables para mapa**

COPY MAP1D1F.

* **Variables generales**

01 CT-DATASET PIC X(08) VALUE 'PERSONA'.

01 CT-DATASET-LEN PIC S9(04) COMP VALUE 160.

PROCEDURE DIVISION.

MAIN-PROCESS.

LUEGO DE ESTABLECER CICLO > 0 - LA DFHCOMMAREA CONTIENE DATOS

```
EXEC CICS  
    RECEIVE MAP(MAPCLI1) MAPSET(MAPSET1)  
END-EXEC.
```

```
MOVE TIPDO1I TO WS-USER-TIPDOC  
MOVE NUMDO1I TO WS-USER-NRODOC
```

```
MOVE WS-USER-TIPDOC    TO TIPDO1O  
MOVE WS-USER-NRODOC   TO NUMDO1O
```

```
EXEC CICS  
    READ DATASET ('PERSONA')  
        RIDFLD (WS-USER-DATA)  
        INTO (REG-PERSONA)  
        LENGTH (CT-DATASET-LEN)  
        EQUAL  
        RESP (WS-RESP)  
END-EXEC
```

EVALUATE WS-RESP

WHEN DFHRESP(NORMAL)

```
MOVE WS-USER-TIPDOC    TO TIPDO1O  
MOVE WS-USER-NRODOC   TO NUMDO1O  
MOVE PER-CLI-NRO      TO NROCLIO  
MOVE PER-NOMAPE       TO NOMAPEO  
MOVE PER-DIRECCION    TO DIREO  
MOVE PER-TELEFONO     TO TELEO  
MOVE PER-EMAIL         TO EMAILO  
MOVE CT-MNS-06         TO MSG1O  
MOVE LENGTH OF MAP1D1FO TO WS-LONG
```

WHEN DFHRESP(NOTFND)

```
MOVE WS-USER-TIPDOC    TO TIPDO1O  
MOVE WS-USER-NRODOC   TO NUMDO1O  
MOVE CT-MNS-03         TO MSG1O
```

WHEN OTHER

```

MOVE WS-USER-TIPDOC      TO TIPDO1O
MOVE WS-USER-NRODOC      TO NUMDO1O
MOVE CT-MNS-08           TO MSG1O

END-EVALUATE.

EXEC CICS
    SEND MAP ('MAPCLI1')  MAPSET ('MAPSET1')
        FROM (MAP-DATA) ERASE
END-EXEC.

EXEC CICS
    RETURN TRANSID ('AD1F')
    COMMAREA (WS-COMMAREA)
END-EXEC.

```

Mapa **MAPCLI1** (simplificado)

Este mapa tendría campos como:

- **ID-CLIENTE** para ingresar el ID
 - **NOMBRE y DIRECCION** para mostrar resultados
 - **MENSAJE** para mostrar acciones a seguir por parte del usuario o errores detectados
-

¿Qué conceptos se integran aquí?

Componente	Función
RECEIVE MAP	Captura datos ingresados por el usuario
SQL DECLARE/FETCH	Consulta DB2 usando cursor para múltiples filas como resultado de la ejecución de una query
PERFORM UNTIL	Bucle COBOL para recorrer resultados
SEND MAP	Devuelve datos al usuario en pantalla
RETURN TRANSID	Finaliza el ciclo y espera nueva acción

Todos estos conceptos forman parte de un mismo CICLO en COBOL CICS.



EXPRESIÓN ‘CICLO 0’ EN CICS (PRIMERA UNIDAD DE TRABAJO en una transacción)

- Se refiere al primer ingreso del usuario a una transacción, antes de que haya interacción o datos disponibles recibidos a través de **LINKAGE SECTION**.
 - Es un concepto práctico que se usa para diferenciar el estado inicial de una transacción de los ciclos o unidades de trabajo posteriores, donde ya hay datos ingresados a través de **LINKAGE SECTION**.
-



Diferencias entre el ciclo 0 y los demás ciclos en CICS

Acción	Ciclo 0 (Inicial)	Ciclos posteriores (Iterativos)
💡 Interacción del usuario	No hay datos ingresados aún	El usuario ya ha ingresado datos en el mapa
📝 RECEIVE MAP	Puede no ejecutarse, o se usa para inicializar campos a través de DFHRESP = MAPFAIL	Se usa para capturar datos ingresados
🧠 Lógica de negocio	Se muestra pantalla vacía o con valores por defecto	Se ejecuta lógica según los datos ingresados
🗄 Acceso a datos almacenados en disco	No se realiza consulta aún	Se consulta la base de datos o archivo VSAM con los datos del usuario
📤 SEND MAP	Se envía mapa vacío o con mensaje de bienvenida	Se envía mapa con resultados, errores o confirmaciones
🔄 Flujo de control	Se prepara el entorno para la interacción	Se procesa y se puede repetir según la lógica del negocio; hasta que el usuario indique el FIN a través de la tecla de función correspondiente



¿Por qué es importante distinguir el ciclo 0?

- Permite inicializar variables y estructuras antes de procesar datos.
 - Evita errores por campos vacíos o no definidos.
 - Mejora la experiencia del usuario al mostrar una pantalla limpia o con datos.
 - Facilita el diseño de mapas y programas más robustos.
-



Ejemplo de detección del ciclo 0 en COBOL-CICS

```
MOVE DFHCOMMAREA TO WS-COMMAREA.
```

```
IF EIBCALEN = 0
```

```
    MOVE LENGTH OF MAPCLI1O TO WS-LONG
```

```
    MOVE LOW-VALUES TO MAPCLI1O
```

```
    MOVE 'Ingrese la opción deseada:' TO MENSAJE
```

```
    EXEC CICS
```

```
        SEND MAP (MAPCLI1)      MAPSET (MAPSET1)
```

```
            FROM (MAPCLI1O) ERASE
```

```
    END-EXEC
```

```
    EXEC CICS
```

```
        RETURN
```

```
        TRANSID ('BD1F')
```

```
        COMMAREA (WS-COMMAREA)
```

```
END-EXEC
```

```
ELSE
```

```
    EXEC CICS
```

```
        RECEIVE MAP (MAPCLI1)
```

```
            MAPSET (MAPSET1)
```

```
            INTO (MAPCLI1I)
```

```
            RESP(WS-RESP)
```

```
    END-EXEC
```

```
END-IF.
```

🔍 ¿Cómo se interpreta esta porción de código?

- **EIBCALEN = 0** Verifica el largo de la **DFHCOMMAREA**
 - Si se cumple la condición; **estamos en el ciclo 0 o primera unidad de trabajo**: no hay datos ingresados aún.
 - Se envía el mapa con un mensaje con instrucción al usuario, y se retorna el control a CICS incluyendo la **DFHCOMMAREA**.

```
EXEC CICS  
RETURN  
TRANSID ('AD1F')  
COMMAREA (WS-COMMAREA)  
END-EXEC.
```

De esta manera resulta PSEUDO-CONVERSACIONAL; libera los recursos tomados.

- Si en cambio NO cumple la condición; **estamos en cualquier otro ciclo > 0**: ya hubo interacción con el usuario; entonces se reciben los datos a través de la **DFHCOMMAREA** y se arman los caminos funcionales que responden a la solución.
- Finalizando con:

```
EXEC CICS  
SEND MAP (MAPCLI1)    MAPSET (MAPSET1)  
      FROM (MAPCLI1O) ERASE  
END-EXEC  
  
EXEC CICS
```

```
RETURN  
TRANSID ('BD1F')  
COMMAREA (WS-COMMAREA)  
  
END-EXEC.
```

Excepto que el usuario decida salir de la aplicación (mediante tecla de función específica: **PF12**) y en ese caso se construye el siguiente camino lógico de salida final; **DEVUELVE EL CONTROL AL CICS**; finalizaron **TODAS** las **UNIDADES DE TRABAJO** (a pedido del usuario) para esta transacción:

En **WORKING STORAGE SECTION**.

```
01 CT-MNS-EXIT    PIC X(72) VALUE      'FIN TRANSACCION BD1F'.
```

En **PROCEDURE DIVISION**.

```
EXEC CICS  
    SEND CONTROL ERASE  
END-EXEC
```

```
EXEC CICS  
    SEND TEXT  
        FROM (CT-MNS-EXIT)  
END-EXEC
```

```
EXEC CICS  
    RETURN  
END-EXEC.
```

Clase 22 asincrónica - práctica con CICS

OBJETIVO

- Construir un código COBOL CICS para mostrar un Menú de opciones al usuario
- Al finalizar esta práctica; el estudiante habrá estado practicando la codificación de un programa pseudo conversacional COBOL CICS

ESPECIFICACIONES

PROGRAMA CICS - MODIFICACIONES

TRANSACCIÓN FXXX - PROGRAMA PGMMOXXX - MAPA MAP5XXX

(donde xxx = nro. alumno)

LAYOUT MAPA (MAP5XXX) : en página siguiente

<p>PGMMOC12-MODIFICACIÓN DE CLIENTES</p> <p>Cargar los datos necesarios y presionar 'ENTER'</p> <p>Tipo de documento: XX Nro.de documento: ZZZZZZZZZ9</p> <p>Nombre y apellido: XXXXXXXXXXXXXXXXXXXXXXX Fecha nacimiento: 99/99/9999 Sexo: X (F/M)</p> <p>Presione tecla ENTER procesar F3 limpiar F12 salir</p>	<p>FC12-MAP5C12 99-99-9999</p> <p>72 caracteres</p>
--	--

Modificar el programa CICS **PGMPRxss** para que acepte como IO , MAPA (MAP5XXX): MODIFICACIÓN CLIENTES o sea llamado desde el PGMMExxx; en este caso considerar que recibe una COMMAREA con los datos de tipo y nro de documento.

Luego de recibidos y validados los datos ingresados; se buscará el cliente en el archivo de Personas, con los datos validados OK:

- Archivo VSAM KSDS = CLIENTES

LAY OUT ALU9999.CURSOS.COPYLIB(CPPERSON)

datos en: KC03xxx.CURSOS.PERSONA.KSDS.VSAM (**DEFINIDO EN TABLA CICS CON DDNAME: PERSOxxx**)

y se enviará el mapa con los datos del cliente solicitado.

NOTA IMPORTANTE:

Los campos pintados de “amarillo” serán modificables por el usuario

Los campos pintados de “turquesa”; serán NO MODIFICABLES

Se mostrarán los errores al pie de página.

RESUMEN DE NAVEGABILIDAD DEL PROGRAMA

1) ¿Cómo ingresaron los datos?

- a) vía SEND/RECEIVE de MAPA? En este caso; la COMMAREA estará vacía.

Controlar que el usuario PRESIONA ENTER, en ese caso, se validarán los datos ingresados de TIPO Y NRO DE DOCUMENTO; y ante validación exitosa; se enviará el MAPA con los datos del Cliente. Si se encuentra algún problema; indicarlo en la línea de mensajes de 72 caracteres disponible.
Ejemplo: cliente dado de baja.

Para finalizar; lo hará al recibir la solicitud del usuario con PF12 (retornando a CICS)

Si el usuario presiona PF3; significa que hay que limpiar el MAPA y volver a enviarlo sin datos para una nueva consulta (mostrar solamente TIPO y NÚMERO DE DOCUMENTO)

- b) vía COMMAREA? En este caso; la COMMAREA tendrá datos.

En este caso; se validará los datos ingresados en COMMAREA de TIPO Y NRO DE DOCUMENTO; y luego se enviarán todos los datos del cliente y los necesarios para posibilitar la ejecución del programa llamador; también vía COMMAREA:

- Nombre de programa llamador
- Tipo y número de documento
- Nombre y apellido del cliente
- Fecha de nacimiento del cliente
- Sexo del cliente
- Código de retorno

TENER PRESENTE QUE LA COMMAREA EN ambos programas llamador y llamado, deberá ser de la misma estructura de datos y del mismo largo total; considerando un filler con espacios al final.

Para finalizar; lo hará retornando al programa LLAMADOR (RETURN TRANSID)

Cómo se integra el campo CÓDIGO DE RETORNO?

- Si la validación fue exitosa: devolver zeros en este campo
- Si la validación tuvo algún problema; integrar:
 - 01 para tipo documento erróneo
 - 02 para número de documento no informado
 - 03 para cliente inexistente
 - 04 para cliente dado de baj

NUEVA CONSIGNA:

Modificar el programa CICS **PGMMExxx** para que en caso que el USUARIO SOLICITE MODIFICACIÓN; realice XCTL PGMMOxxx con la COMMAREA correspondiente.

OBJETIVO

- Construir código COBOL CICS para realizar un LINK o enlace desde un programa COBOL CICS a otro
- Al finalizar esta práctica, el estudiante habrá practicado la modalidad CONVERSACIONAL de un programa CICS.
- Antes se había construído un Menú que invoca los subprogramas de manera pseudo conversacional (XCTL); ahora se hará con LINK, que provoca que sea conversacional

¿Qué significa CONVERSACIONAL?

Todos los recursos (ambos programas en este caso) permanecen en memoria

¿Qué significa PSEUDO CONVERSACIONAL?

Que solamente permanece en memoria el programa invocado y libera al invocador. Se recomienda el uso de esta opción en CICS, excepto para funcionalidades muy puntuales que NO REQUIEREN intervención del usuario.

ESPECIFICACIONES

PRÁCTICA CICS - LINK

Tomar como ejemplo:

- KC02788.ALU9999.FUENTE(**PGMCON9L**) Programa que realiza LINK al programa PGMLNK99
- KC02788.ALU9999.FUENTE(**PGMLNK99**)

Modificar el programa CICS **PGMMExxx** para que en caso que el USUARIO invoque un programa según funcionalidad con la COMMAREA correspondiente; según el siguiente detalle de las teclas presionadas:

- 1) ALTA: realice LINK PGMALxxx
- 2) BAJA: realice LINK PGMBAxxx
- 3) MODIFICACIÓN: realice LINK PGMMOxxx
- 4) CONSULTA: realice LINK PGMPRxxx

Tener en cuenta de tomar backup de la versión actual de cada programa con XCTL.