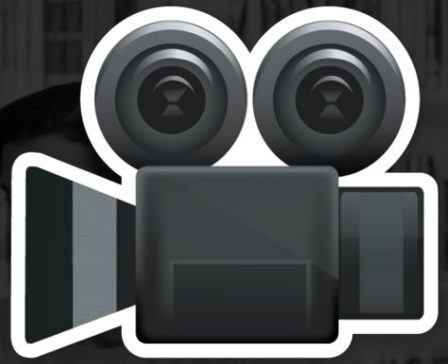
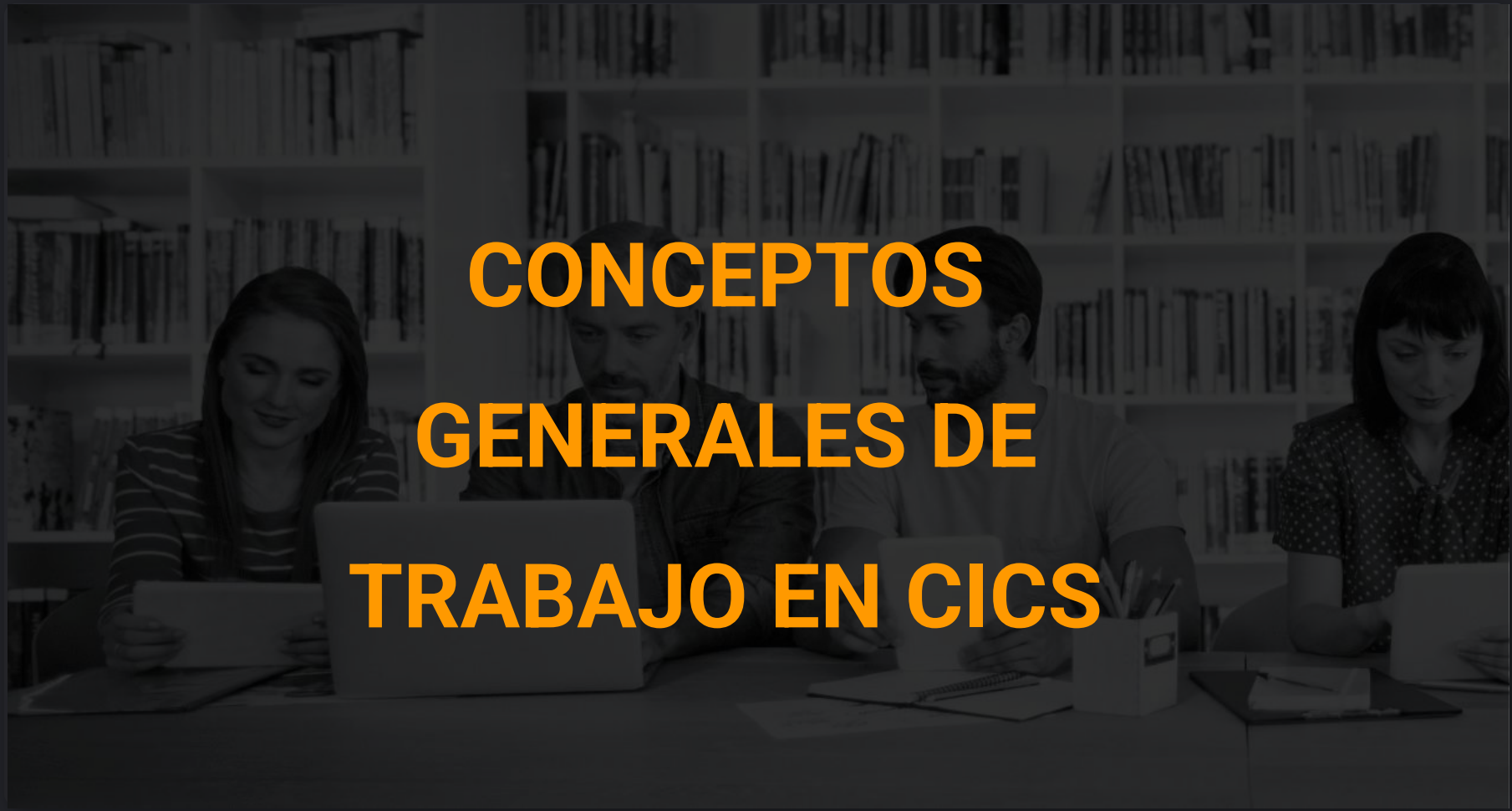
Curso

PROGRAMA COBOL CICS

Codeki™

Recuerda poner a grabar la clase.




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

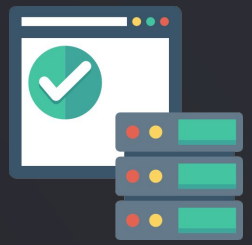
01

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 pasarle 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.

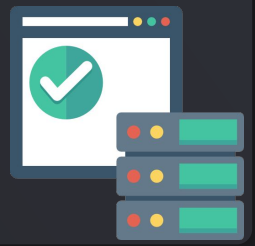


01

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.

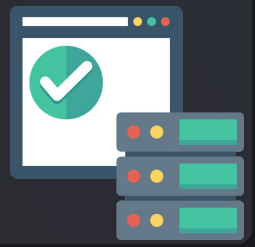
Por supuesto que nuestro programa deberá tener incorporado el código pertinente para llevar a cabo esos chequeos.

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

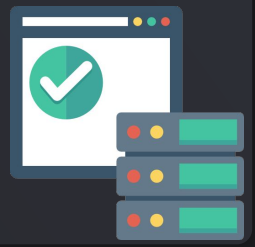
03

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**.



03

HANDLE CONDITION

OBSERVACIONES sobre el HANDLE CONDITION

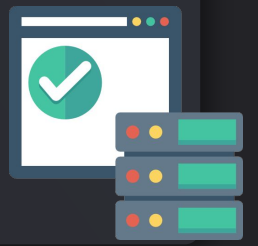
Se pueden especificar hasta 16 condiciones en un solo comando.

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:

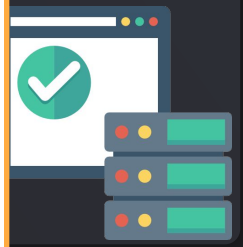
```
EXEC CICS HANDLE CONDITION
        ERROR (error-cics)
END-EXEC.
```

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.

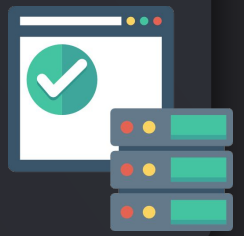


03

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.
```



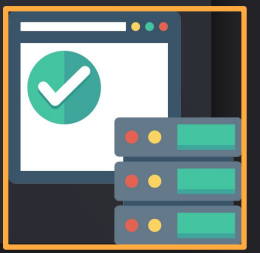
03

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

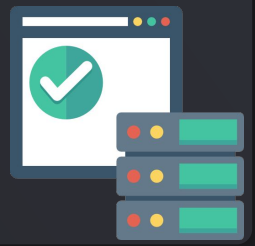
HANDLE AID vs EIBAID

```
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

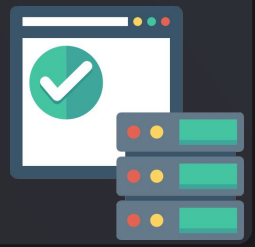


03

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



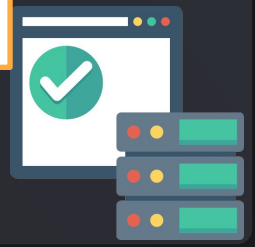
03

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.



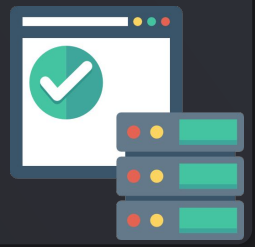
03

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.
```

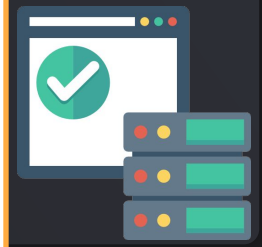
- **ABSTIME data-value** especifica el nombre de la data-area definida en el comando ASKTIME. Este valor de tiempo absoluto podrá ser formateado de acuerdo a las opciones que se indiquen.
- **YYDDD Data-area** especifica un campo de 5 caracteres donde se retorna la fecha en el formato indicado.
- **YYMMDD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



03

FORMATTIME

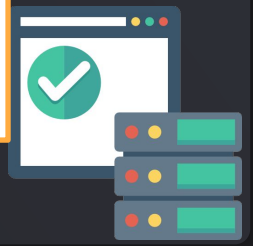
- **DATE data-area** especifica un campo de 8 caracteres en donde se retorna la fecha en el formato default seteado en la Instalación.
- **DATESEP Data-value** especifica el carácter que será insertado como separador entre los componentes de una fecha. Si se omite data-value, asume '/' como separador.
  - **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

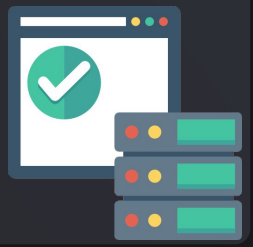
ASKTIME y FORMATTIME

```
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



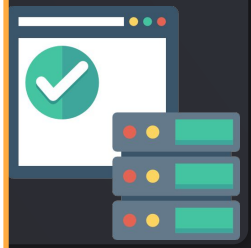
03

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.
```

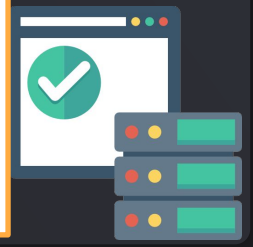
- **MAP** especifica el nombre del **MAPA** a ser enviado a la terminal (1 a 7 caracteres). Deberá coincidir con el nombre definido en la macro **BMS DFHMDI.**
- **MAPSET** especifica el nombre del **MAPSET,** conjunto de mapas al que pertenece el mapa a ser enviado a la terminal (1 a 7 caracteres). 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.



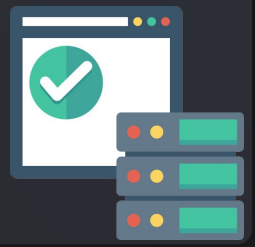
03

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

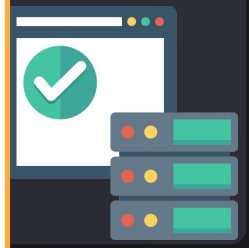
- **FROM** esta opción es usada cuando el mapa ya ha sido recibido en un área de almacenamiento, pero los datos no fueron formateados como mapa. Este comando realmente los formatea.
- **LENGTH** especifica La longitud del dato a ser formateado; debe definirse como un campo binario de media palabra **PIC 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.

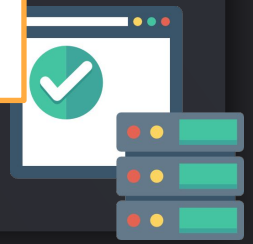


03

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

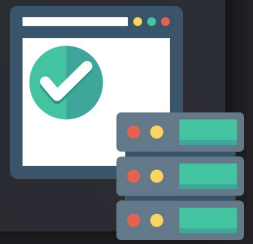
03

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

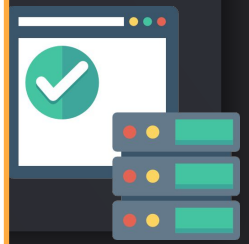
MANEJO DE ARCHIVOS - READ

- **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.



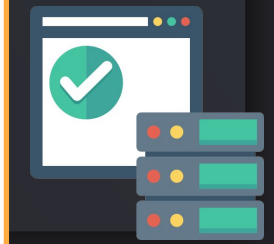
03 MANEJO DE ARCHIVOS - READ

- **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.

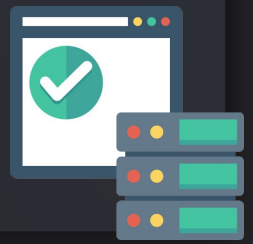
- **GENERIC** identifica a la clave usada para buscar un registro en un archivo **VSAM KSDS** como una **CLAVE PARCIAL,** cuya longitud se especifica en **KEYLENGTH.**
- **KEYLENGTH** es un campo binario de dos bytes que identifica la longitud de la clave especificada en **RIDFLD.** Esta opción debe 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

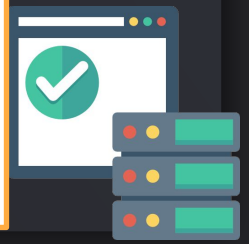
CONDICIONES DE EXCEPCIÓN EN COMANDO READ

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.



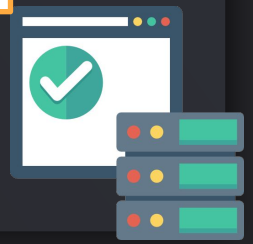
03

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.



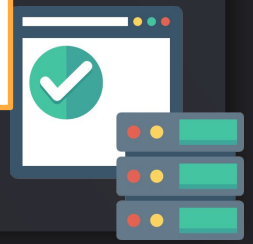
03

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

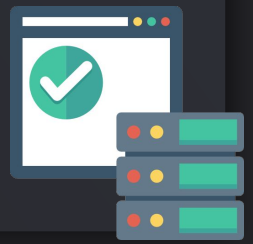
MANEJO DE ARCHIVOS - READ

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.



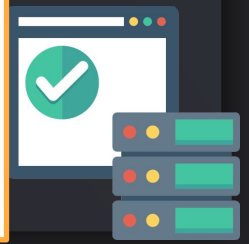
03

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.



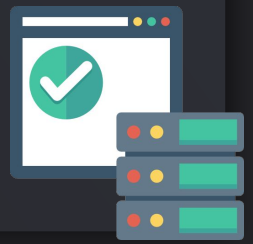
03

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

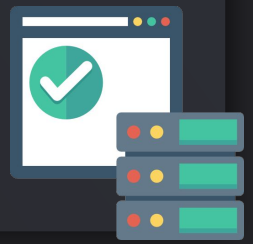
MANEJO DE ARCHIVOS - REWRITE

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

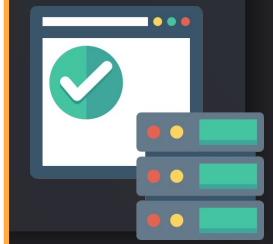
MANEJO DE ARCHIVOS - DELETE

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** específica 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

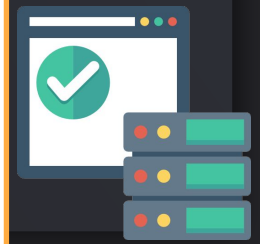
El comando **STARTBR** permite posicionar a la próxima operación de lectura en la clave que cumpla con determinada condición de búsqueda (Equivale a una sentencia **START** de **COBOL** sobre archivos **VSAM**).

```
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

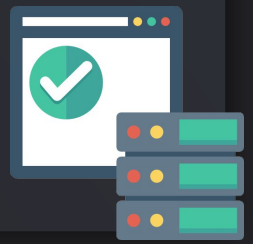
MANEJO DE ARCHIVOS - READNEXT

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 saltear la lectura de los registros existentes entre el valor de la última clave accedida y el nuevo valor del **RIDFLD**.



03

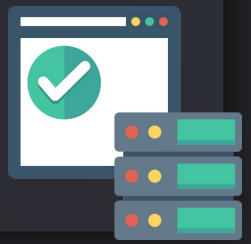
MANEJO DE ARCHIVOS -READPREV

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 I**NVREQ.**



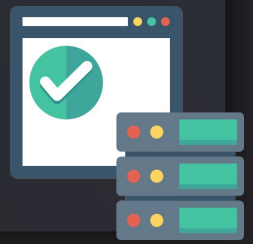
03

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.

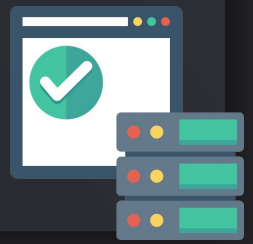


03

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!