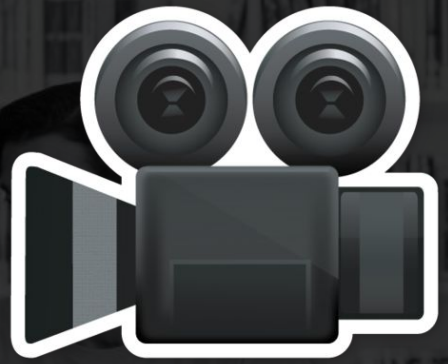
# Curso

## PROGRAMADOR COBOL – Introducción a CICS

Codeki™

# 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

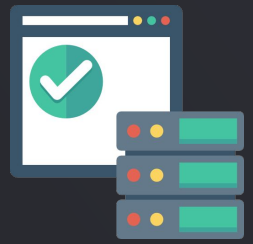
# INTRODUCCIÓN AL CICS - TEMARIO

## 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

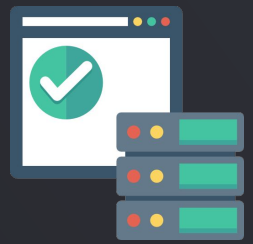
# INTRODUCCIÓN AL CICS - TEMARIO

## 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

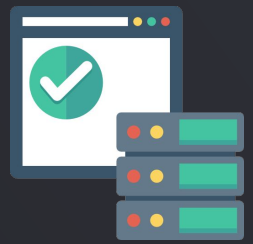
# INTRODUCCIÓN AL CICS - TEMARIO

## 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

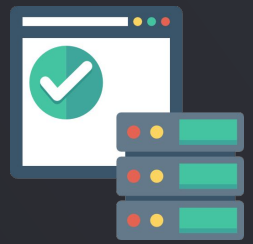
# INTRODUCCIÓN AL CICS - TEMARIO

## UNIDAD LÓGICA DE TRABAJO

- CONCEPTOS
- SYNCPOINT
- SYNCPOINT ROLLBACK
- ABEND

## DEMOSTRACIÓN DE CEDF

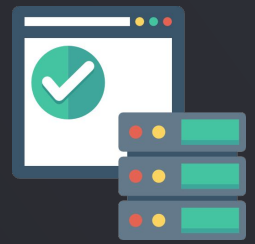
## DEMOSTRACIÓN DE CEMT CECI CEDA



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



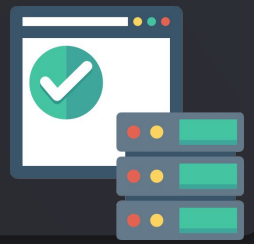
01

# 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

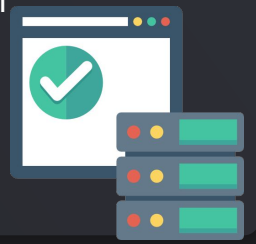
# ESTRUCTURA DEL CICS

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:

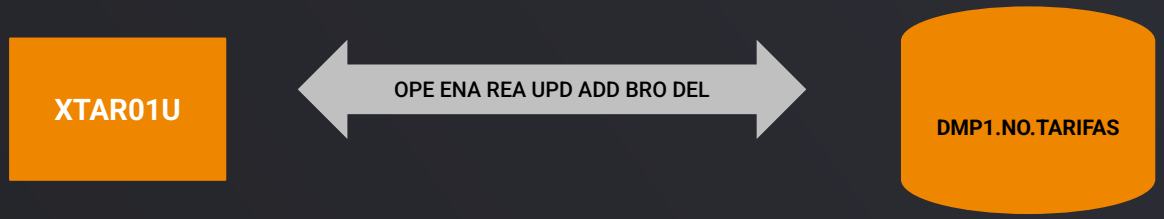
- **FCT (File Control Table):** Todos los archivos que deban ser accedidos por nuestras aplicaciones deberán estar declarados con una entrada en esta tabla. La relación consiste en el nombre del DATASET para el CICS, que consta de 8(ocho) caracteres, y el nombre real o label que este posee en el disco.

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.

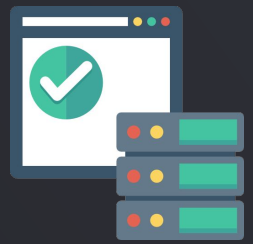


01

# ESTRUCTURA DEL CICS



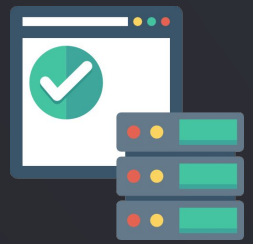

TASK COMANDOS ARCHIVO



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.

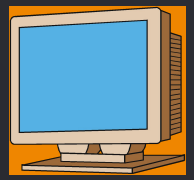


01

# ESTRUCTURA DEL CICS

ACLI

Ter(S205) Tra(CEMT) Pri( 000 ) Pag Ins Ati Tti

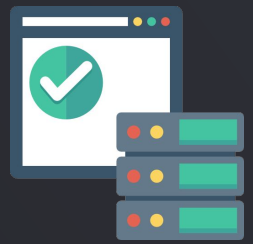


TRANSACCIONES

TERMINALES

APR1

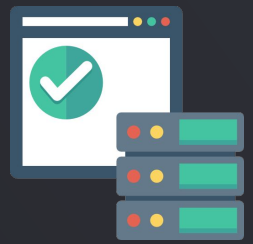
Net(IGKS205 ) Acq



01

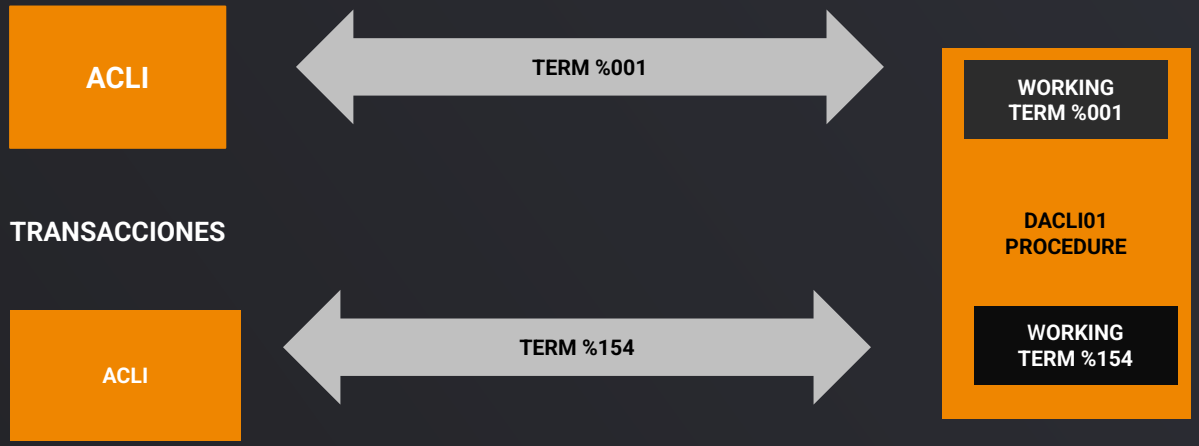
# ESTRUCTURA DEL CICS

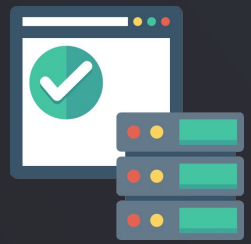
**-PCT (Program Control Table):** Establece la relación transacción/programa. Tiene una entrada por cada transacción a invocarse en el ambiente de CICS. Al ser solicitada la ejecución de una transacción, el CICS automáticamente le asigna un número único de tarea. Esta administración ejercida por el CICS 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

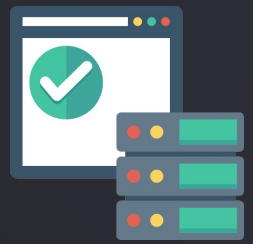


01

# 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

DACLI01
PROGRAMA

TERM %001 Y TERM %154

DACLI01
MODULO
EJECUTABLE

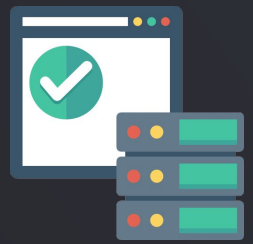
PROGRAMAS

DIRECCIÓN DEL
EJECUTABLE

MACLI01
MAPA

TERM %154 Y TERM %001

MACLI01
MODULO
EJECUTABLE

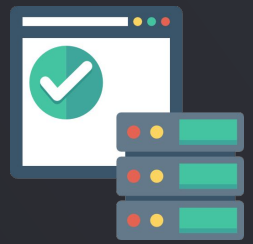


01

# 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.

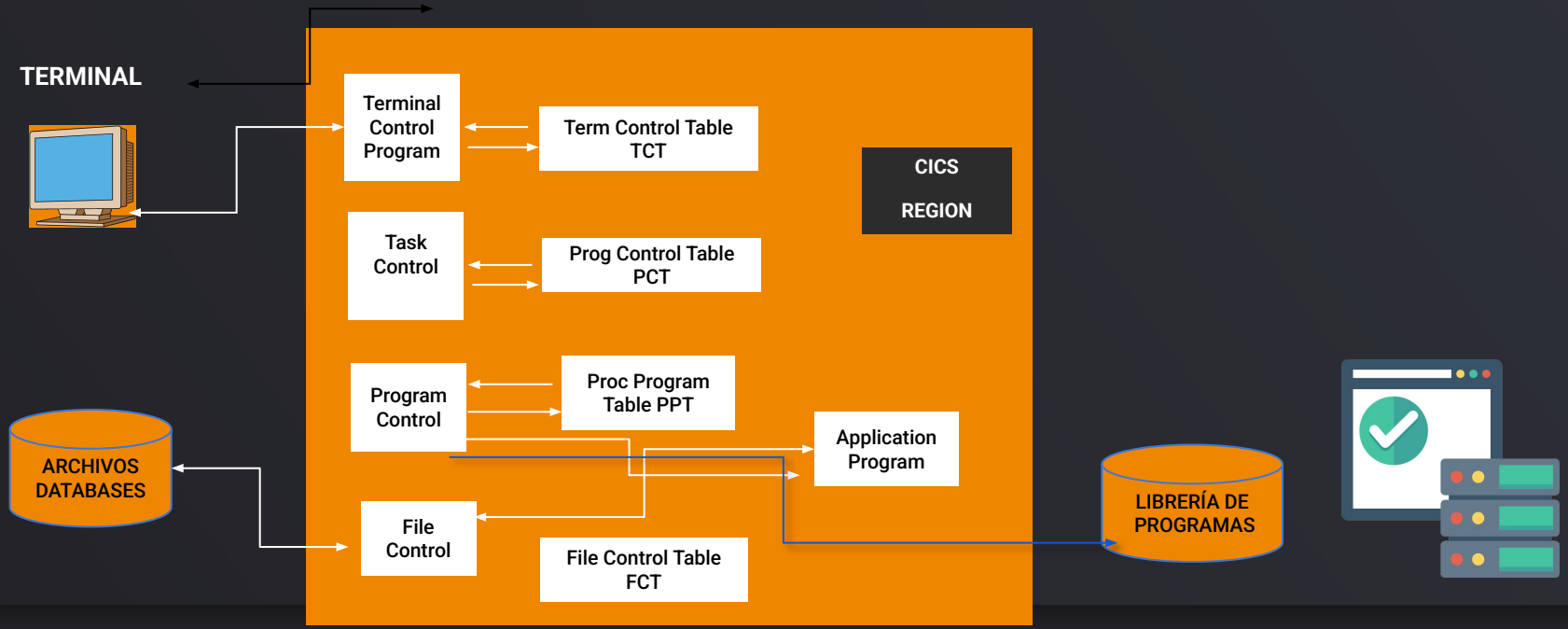
Esto nos permite disponer rápidamente de los programas, mapas, etc., sin tener que esperar a que se recarguen las tablas por la bajada y subida del CICS.



01

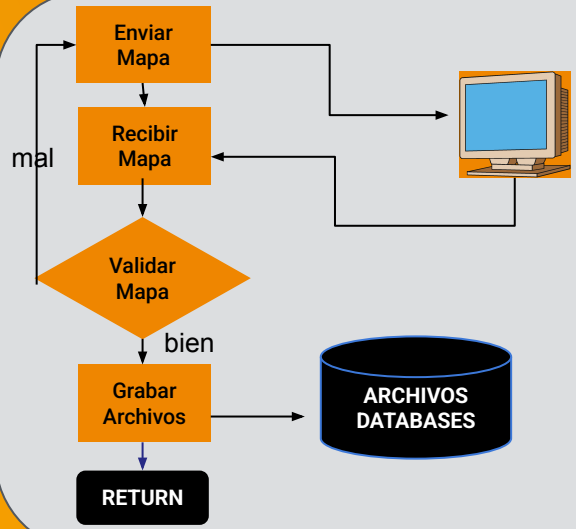
# ESTRUCTURA DEL CICS

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



# PROGRAMACIÓN CONVERSACIONAL

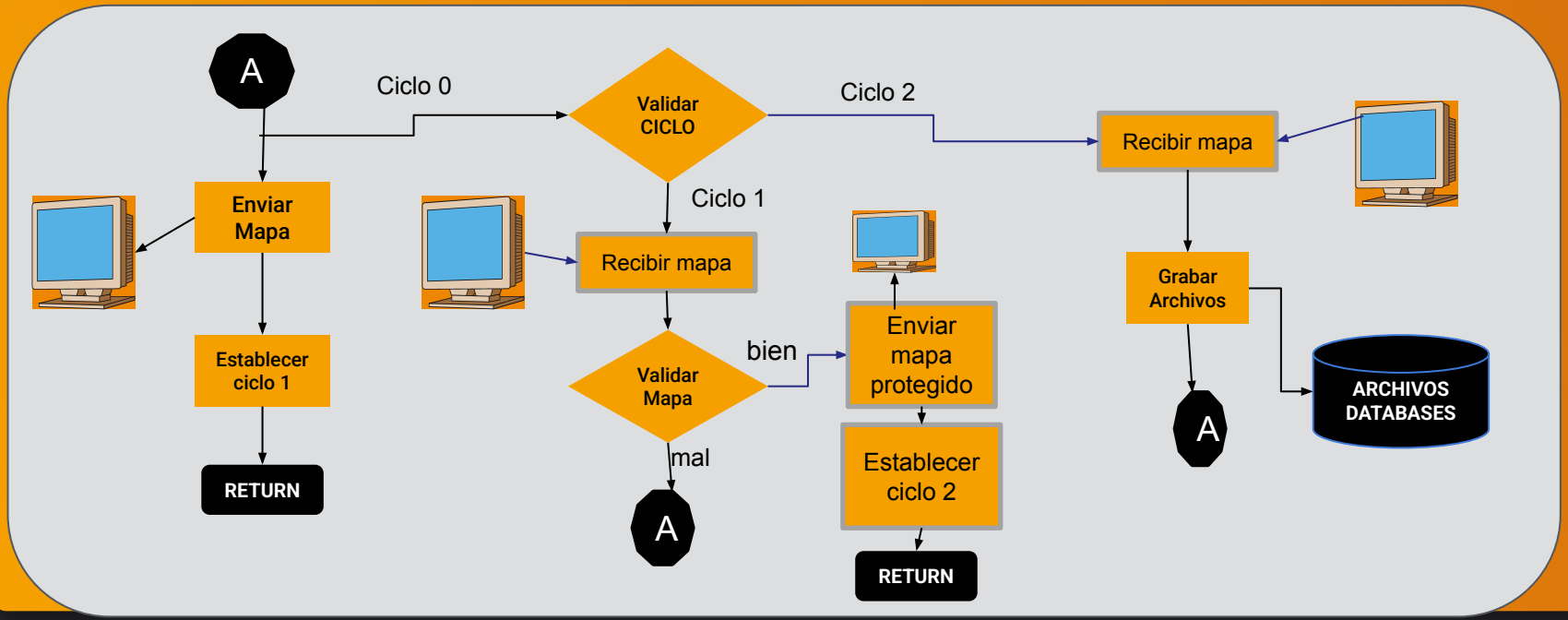
Este método de programación bajo el CICS no es el recomendable, ya que los recursos asignados a la transacción quedan tomados hasta la finalización de la misma. El tiempo de respuesta de la transacción es dependiente de la respuesta del usuario de la misma.




**Nuestra aplicación queda a la espera que el operador pulse ENTER / PF / ATENCIÓN para recibir el MAPA**

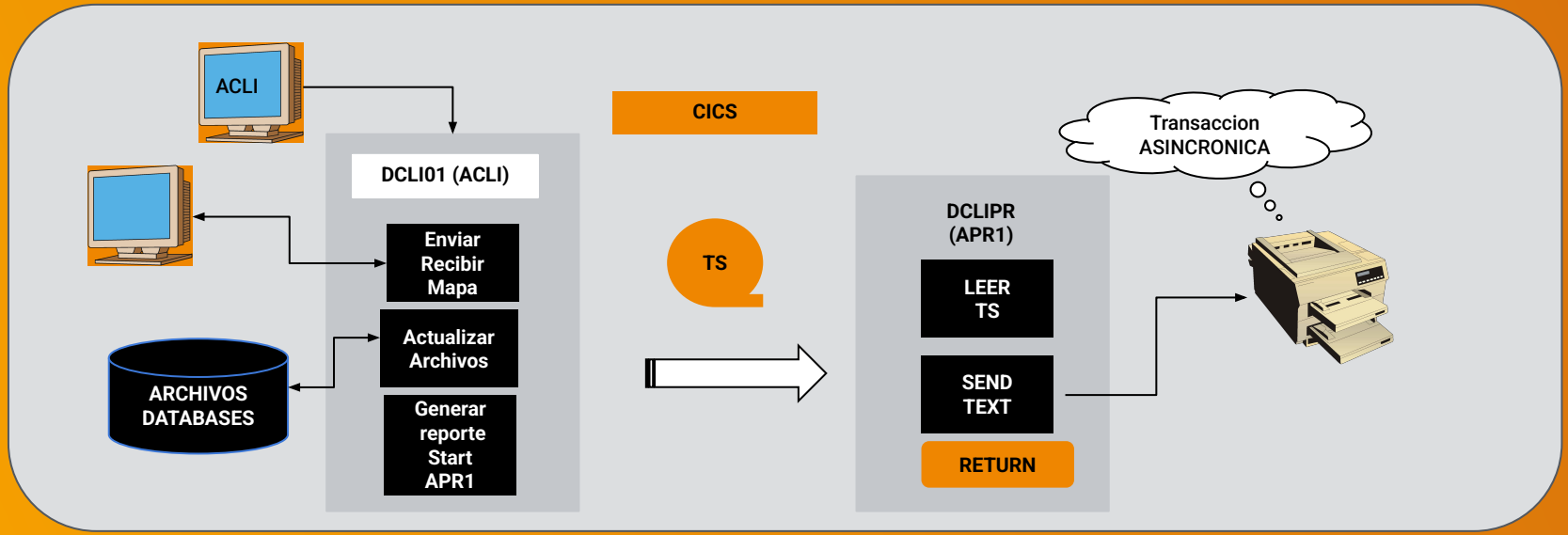
# PROGRAMACIÓN PSEUDO-CONVERSACIONAL

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)

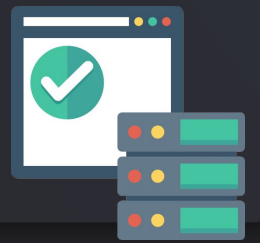
02

# 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.



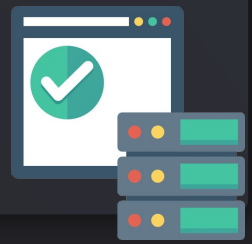
02

# 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**:

COMMAND

ARG

OPTION

```
EXEC CICS READ
     DATASET ('filename')          ('XCLI01U') (WS-FILE)
     [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 ' '.
```

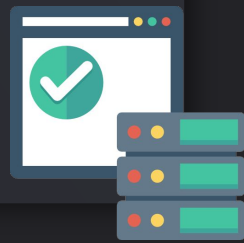
# CODIFICACIÓN PROGRAMA FUENTE

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.
```

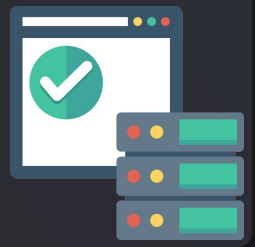


02

# 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 WAPTCCOM.
```



02

# CODIFICACIÓN PROGRAMA FUENTE

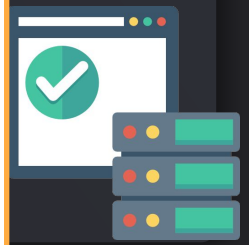
Inclusión en nuestra WORKING STORAGE SECTION de los copys que se encuentran disponibles para desarrollar programas bajo CICS:

```
      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 WAPTCCOM.
```

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 WAPTCCOM.
```

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

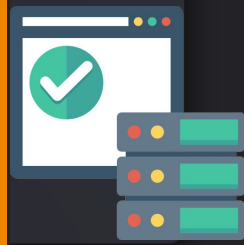
**Incorporaremos tantos copys de mapas como sean necesarios para completar el objetivo del programa y que hemos diseñado bajo BMS o algún utilitario para generación de mapas (SDF).**

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

**Programa: DDTC001**
**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:

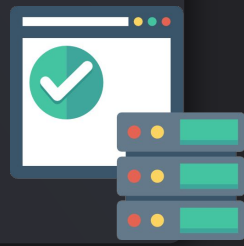
```
      WORKING-STORAGE SECTION.
*-----------------------*
 01  FILLER             PIC X(20)  VALUE '*-> WORKING'.

*-------------------------------------------------------------*
*--> 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 WAPTCCOM.
```

**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 WAPTCCOM.
```

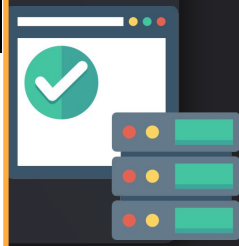
Esta Copy contiene los valores para hacer posible la asignación de atributos a los campos del mapa con todas las combinaciones posibles. (para mapas de pantallas como de impresión).

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

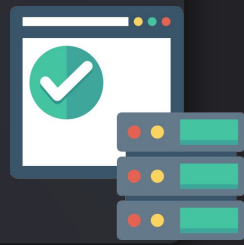
Inclusión del área de comunicación COMMAREA (COMMON AREA) en la LINKAGE SECTION:

```
      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.**

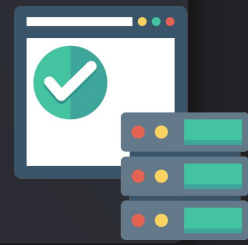
**El tamaño mínimo será el requerido por nuestra aplicación para efectuar el proceso en la forma esperada.**



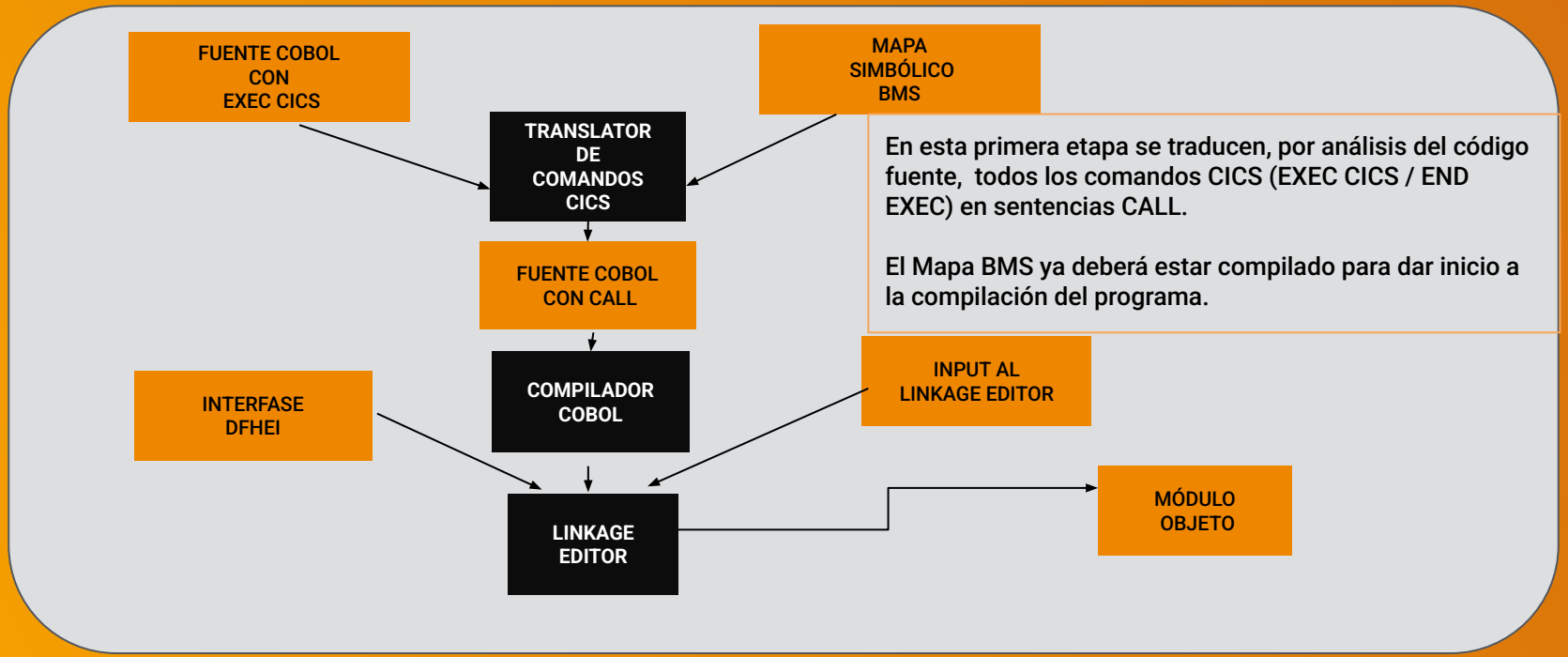
# 02 CODIFICACIÓN PROGRAMA FUENTE

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-OPCION.
  MOVE LENGTH OF COMMAREA-DSPT020  TO COM-LEN.

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



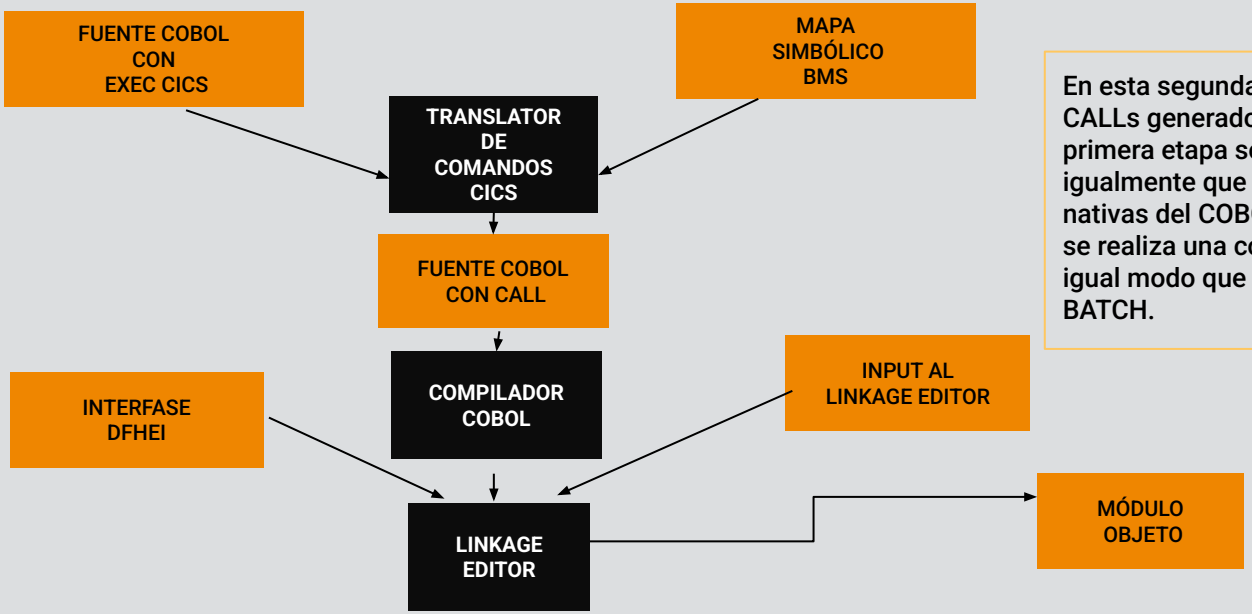
# PROCESO DE COMPILACIÓN




En esta primera etapa se traducen, por análisis del código fuente, todos los comandos CICS (EXEC CICS / END EXEC) en sentencias CALL.

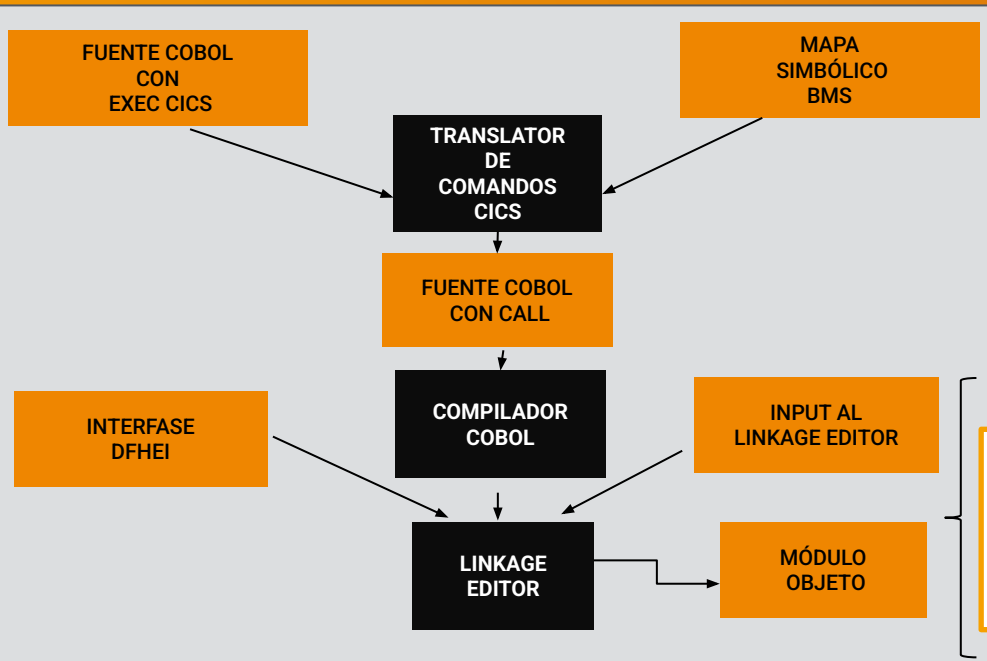
El Mapa BMS ya deberá estar compilado para dar inicio a la compilación del programa.

# PROCESO DE COMPILACIÓN




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 COMPILACIÓN




El módulo objeto es LINK EDITADO de la forma habitual, vinculando al programa con el módulo de interfase DFHECI.
El módulo ejecutable obtenido es almacenado en la librería de carga del CICS

02

# ALTA DEL PROGRAMA CICS

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

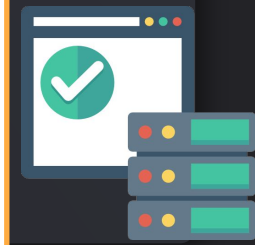
Para ello, nos valdremos de la aplicación **CEDA**, que nos permitirá dar de alta el programa, el mapa, archivos nuevos que requiere nuestra aplicación, etc.

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

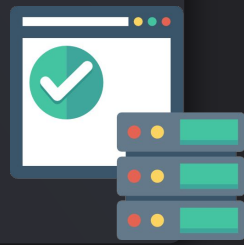
03

# BMS (Basic Mapping Support)

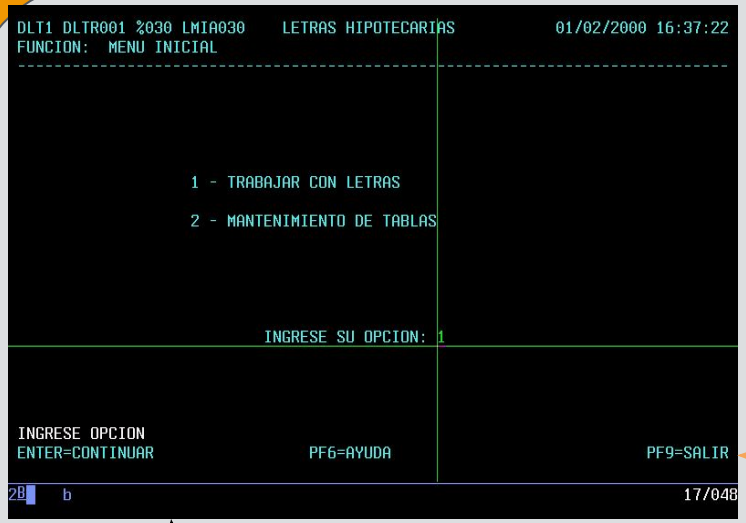
El **BMS** es una interfase entre el TCP y nuestros programas de aplicación. Nos permite aislarnos de las características de una terminal a la hora de enviar o recibir la información.
En la operación de enviar datos desde el programa hacia la terminal (de SALIDA), el BMS toma los datos del área de trabajo 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  MLTR001I.
   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:

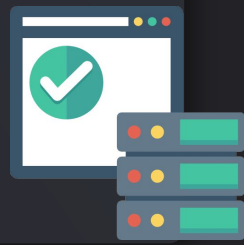
- Longitud y ubicación de los campos de datos
- Atributos de los campos de datos (Protegidos, brillantes, dark, etc.)
- Constantes (encabezamiento, descripciones)
- Características de las terminales

**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)
- NOMBRE**A**  Byte de atributo con las características de NOMBRE (Brillante)
- NOMBRE**L** Byte de atributo que indica el largo en bytes de la variable NOMBRE
- NOMBRE**F**  Flag Byte. Generalmente en X'00'. Si NOMBRE fue borrado asume X'80'
- NOMBRE**I**  Campo de Input. Si no se ingresó NOMBRE contendrá Low-Values.
- NOMBRE**O** Campo de Output. Si contiene Low-Values no se transmite a la terminal.

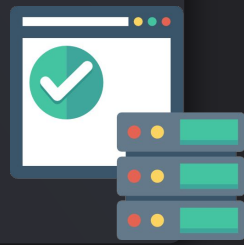


03

# Código fuente BMS

**EDITANDO EL FUENTE DE UN MAPA BMS**

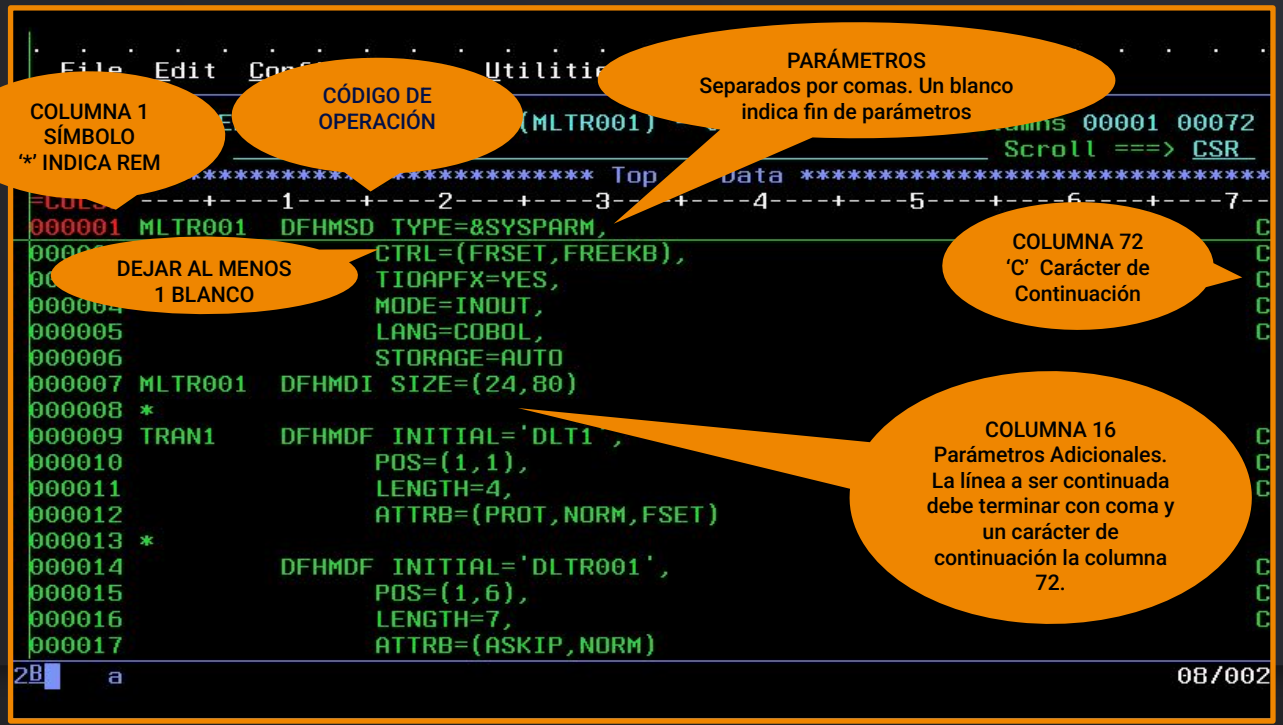
```
 File  Edit  Confirm  Menu  Utilities  Compilers  Iest  Help
EDIT       AEND.ADMINPRD.SOURCE(MLTR001) - 01.00          Columns 00001 00072
Command ===> _____________________________________________ Scroll ===> CSR
****** ***************************** Top of Data ********************************
=COLS>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



03

# Código fuente BMS

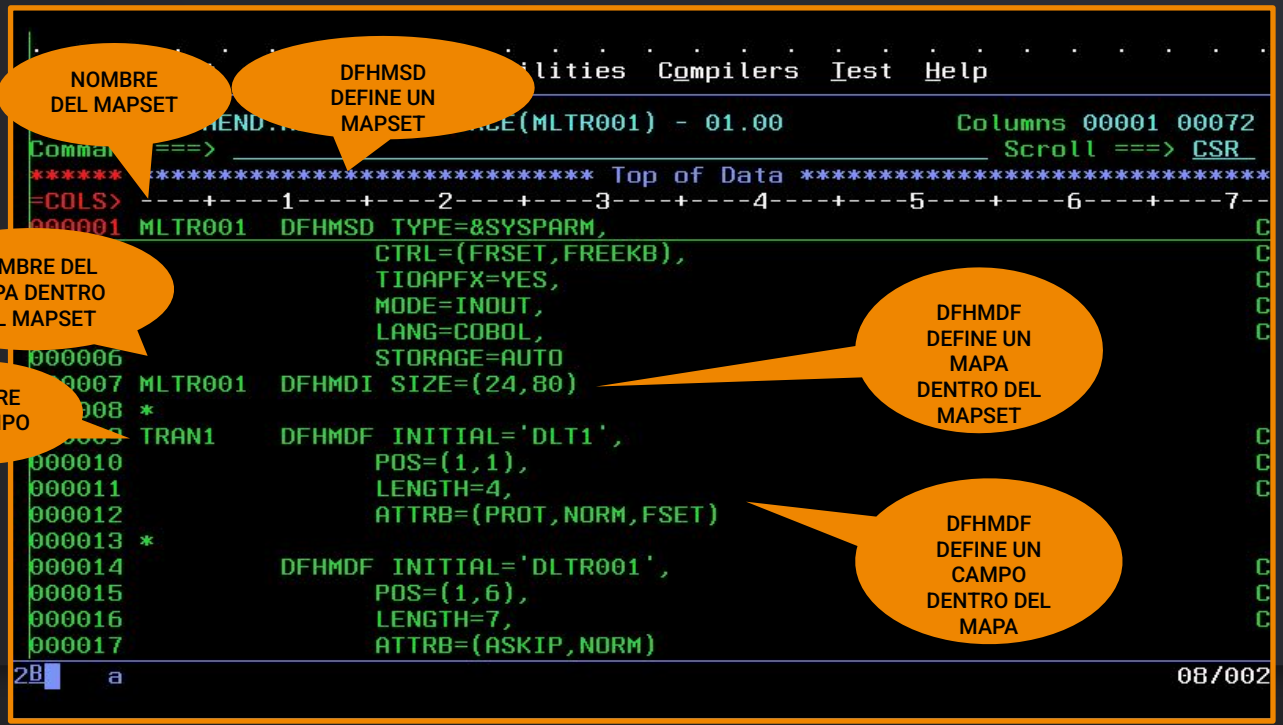
## EDITANDO EL FUENTE DE UN MAPA BMS



03

# Código fuente BMS

## EDITANDO EL FUENTE DE UN MAPA BMS



03

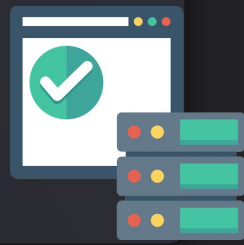
# MACROS BMS

**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.



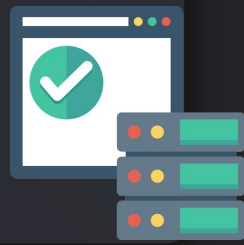
03

# 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

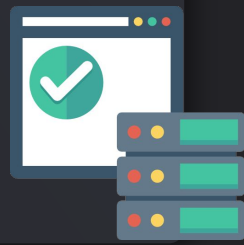
# MACROS BMS

**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).



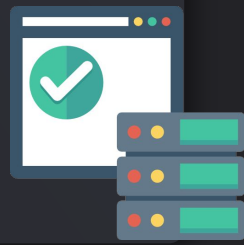
03

# 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.

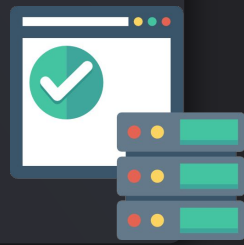


03

# 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

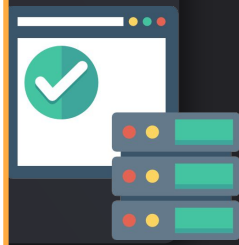
# MACROS BMS

**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])],
[GRPNAME=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.

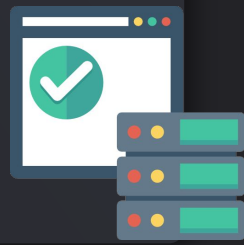


# MACROS BMS

**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],
                 [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.

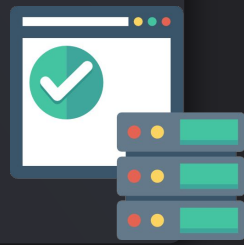


# MACROS BMS

**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],
                 [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'.

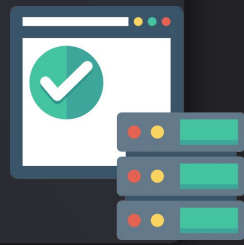


# MACROS BMS

**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],
                 [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

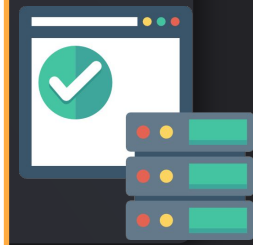
# MACROS BMS

**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).
```

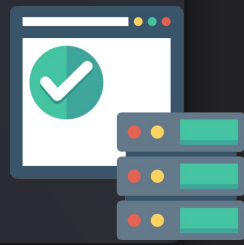


# MACROS BMS

**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],
                 [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 GRPNAME 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' .

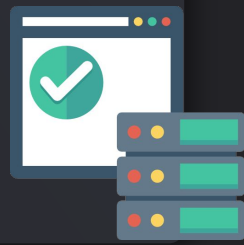


03

# 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

FUENTE ASSEMBLER CON MAPSET

OBTENCIÓN DEL MAPA FÍSICO

INPUT AL LINKAGE EDITOR

LINKAGE EDITOR

OBTENCIÓN DEL MAPA SIMBÓLICO

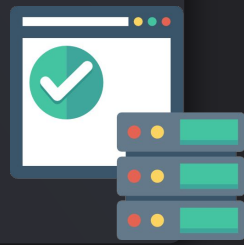
MAPA FÍSICO DE CARGA

COPY COBOL MAPA SIMBÓLICO

Lo probamos en el CICS con la transacción CECI

CECI SEND MAP('MLTR001') ERASE

Lo encontramos en la WORKING STORAGE SECTION o en la LINKAGE SECTION



## 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 INTERFASE BLOCK

04

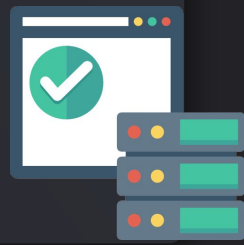
# COMMAREA

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 llego 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

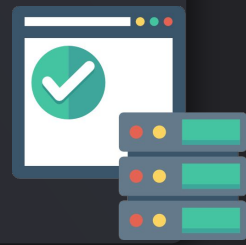
# COMMAREA

```
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.
```

PROGRAMA EMISOR

```
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 RECEPTOR



04

# COMANDO LINK

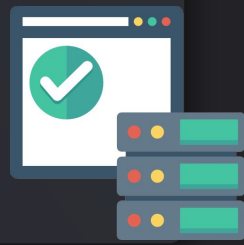
```
EXEC CICS LINK
          PROGRAM    (‘name’)
         [COMMAREA (‘data-area’)
          LENGTH       (‘data-value’) ]
END-EXEC.
```

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

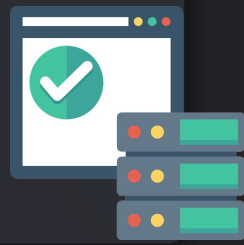
**PROGRAMA PRINCIPAL**

```
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.
```



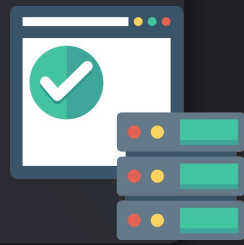
04

# 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**

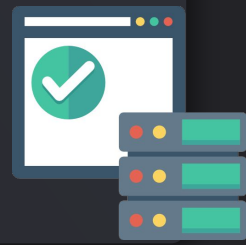
```
MOVE LENGTH OF COMMAREA-DSPT020 TO COM-LEN.
EXEC CICS XCTL PROGRAM  ('DSPT020')
          COMMAREA (COMMAREA-DSPT020)
          LENGTH   (COM-LEN)
END-EXEC.
```

**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

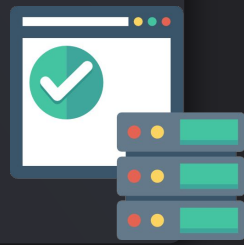
# COMANDO RETURN

```
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.



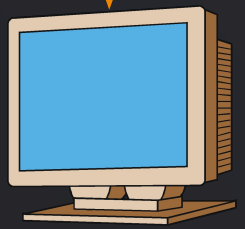
04

# COMANDO RETURN

CICS

**TRANSACCIÓN AAAA**

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



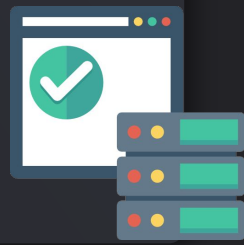
ENTER

PF1 – PF24

ATTN

**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

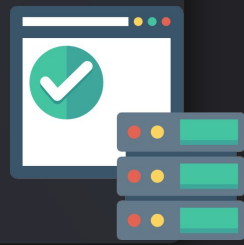
```
EXEC CICS START
          TRANSID      (‘name’)
         [INTERVAL     (hhmmss) : TIME(hhmmss)]
         [FROM         (‘data-area’)
          LENGTH       (‘data-value’) ]
         [TERMID       (‘name’)]
         [REQID        (‘name’)]
END-EXEC.
```

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.

**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**

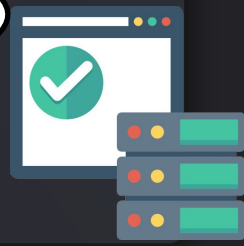
```
MOVE LENGTH OF REG-START TO COM-LEN.
EXEC CICS RETRIEVE
        INTO (REG-START)
        LENGTH (COM-LEN)
END.EXEC.
```

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 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**

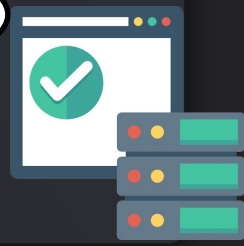
```
MOVE LENGTH OF REG-START TO COM-LEN.
EXEC CICS RETRIEVE
      INTO (REG-START)
      LENGTH (COM-LEN)
END.EXEC.
```

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'



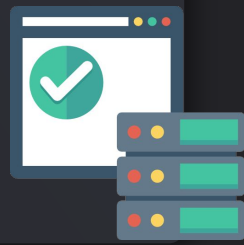
04

# 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.**

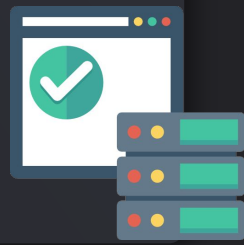


04

# 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!