

DIPLOMATURA EN PROGRAMACION ABAP  
MÓDULO 11: SCREEN PAINTER y MENU PAINTER

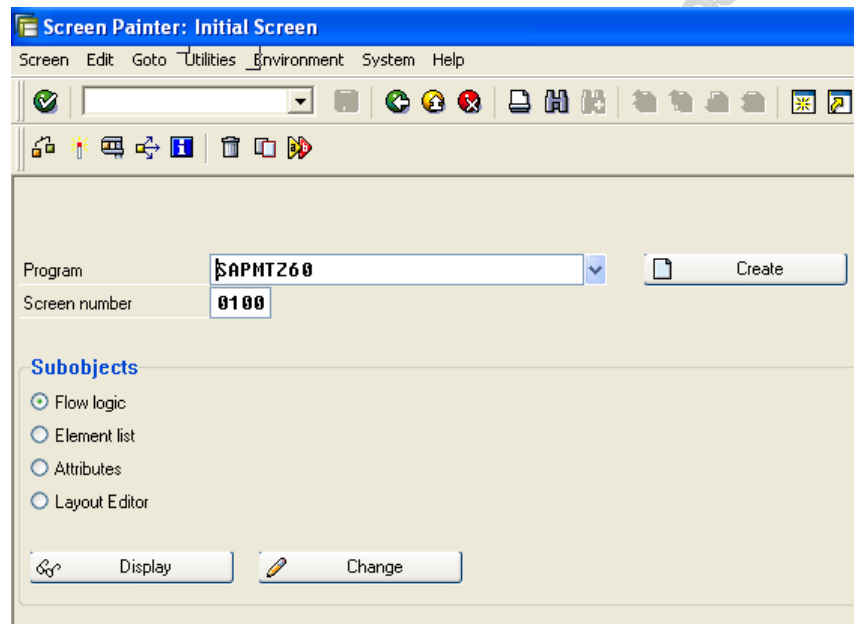
# Utilización del Screen Painter y del Menu Painter para la creación y codificación de programación de diálogo y Module Pools con ABAP

# Screen Painter

A continuación se detallan sus características más salientes.

## Introducción al Screen Painter

El Screen Painter nos permite crear las pantallas de SAP y diseñar su comportamiento (lógica de proceso). Dentro del Editor de Dynpros en SAP, encontramos las siguientes solapas:



- Atributos: en esta ventana se define el tamaño de la dynpro y cuál será la próxima dynpro asociada al module Pool, es decir, que se establecerá el flujo de procesamiento del module pool. También podemos definir el tipo de screen.

**Screen Painter: Display Screen for SAPMTZ60**

Screen number: **100** Active

Attributes | Element list | Flow logic

Short description: Dialog programming: Tables TC (initial screen)

Original language: **EN** English Package: **SDWA**

Last changed on/by: **18.09.2001 10:11:39**

Last generation: **0:00:00**

**Screen type**

- ☒ Normal
- ☐ Subscreen
- ☐ Modal dialog box
- ☐ Selection screen

**Settings**

- ☐ Hold Data
- ☐ Switch off runtime-compress
- ☐ Template - non-executable
- ☐ Hold Scroll Position
- ☐ Without Application Toolbar

**Other attributes**

Next Screen: **200**

Cursor position: [Empty field]

Screen group: [Empty field]

Lines/Columns: Occupied **4 21** Mainten. **21 83**

Context menu FORM ON CTMENU

- Lista de elementos: en esta solapa se definen los componentes que formarán parte de la dynpro. Está directamente asociada con todos los elementos diseñados con el Screen Painter.


**Screen Painter: Display Screen for SAPMTZ60**

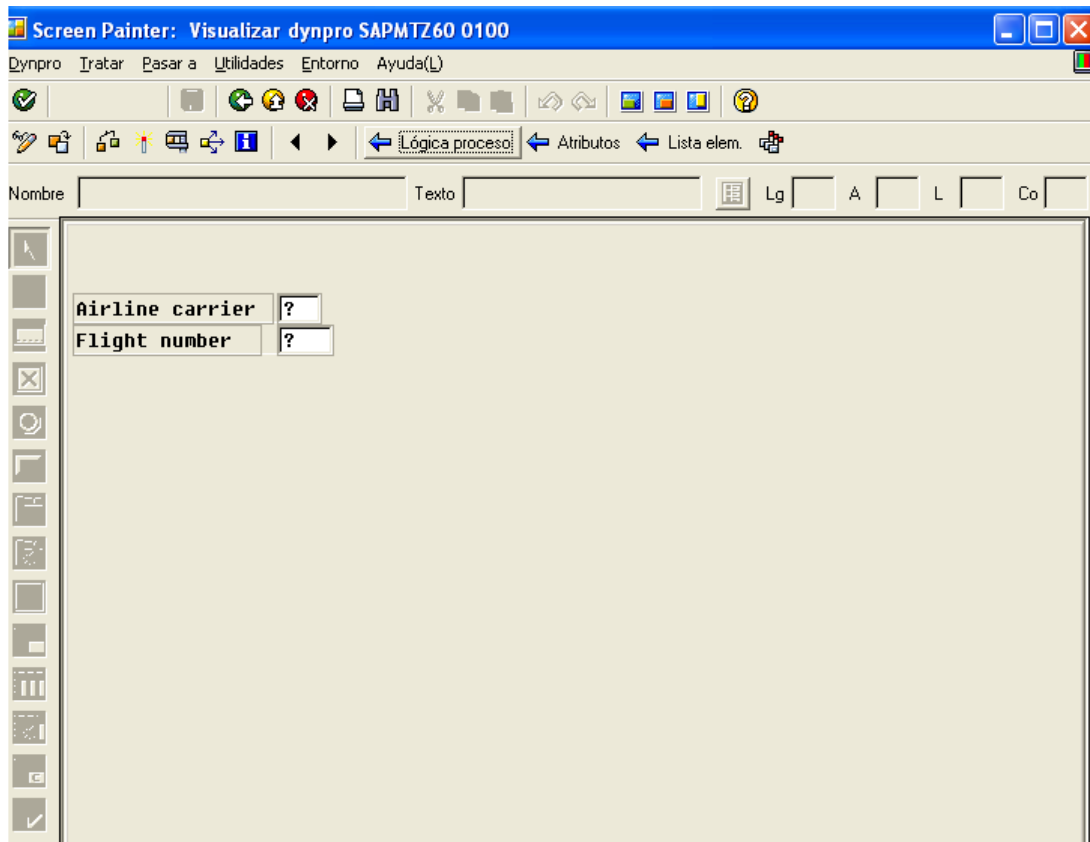
Screen number: **100** Active

Attributes | **Element list** | Flow logic

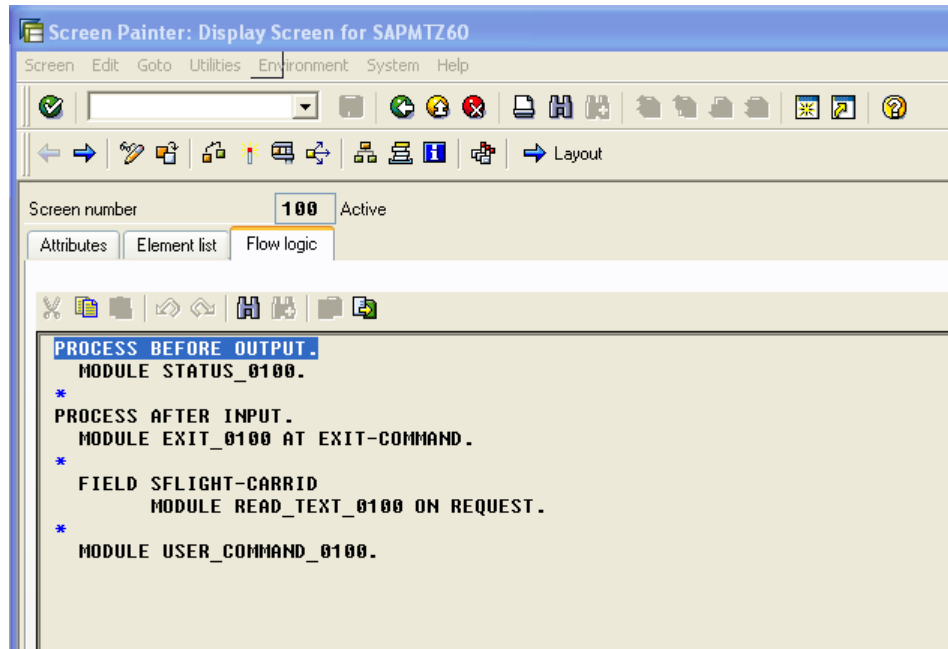
General attr. | Texts / I/O templates | Special attr. | Display attr. | Mod. groups / functions | References

Hi...	Name	Type of...	Line	Col...	DefLg	VisLg	Hei...	Scr...	Format	Input	Out...	Outp...	Dict...	Dict...	Property list
	<u>SFLIGHT-CARRID</u>	Text	3	1	16	16	1			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	F	<a href="#">Properties</a>
	SFLIGHT-CARRID	I/O	3	18	3	3	1		CHAR	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>		<a href="#">Properties</a>
	<u>SFLIGHT-CONNID</u>	Text	4	1	15	15	1			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	F	<a href="#">Properties</a>
	SFLIGHT-CONNID	I/O	4	18	4	4	1		NUMC	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>		<a href="#">Properties</a>
	OK_CODE	OK	0	0	20	20	1		OK	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		

- Para acceder al diseño de la dynpro, seleccionaremos el botón  Layout, ubicado en la barra de tareas del editor:



- **Lógica del proceso:** En esta sección definiremos el código abap y los eventos correspondientes al procesamiento de la aplicación de diálogo, denominados PBO (PROCESS BEFORE OUTPUT- procesos anteriores a la presentación de la dynpro-) y PAI (PROCESS AFTER INPUT- procesos posteriores al ingreso de datos a la dynpro por parte del usuario). Estos eventos serán detallados más adelante. En SAP, al conjunto de pantalla y lógica de proceso de la misma se le denomina Dynpro ('Dynamic Program').



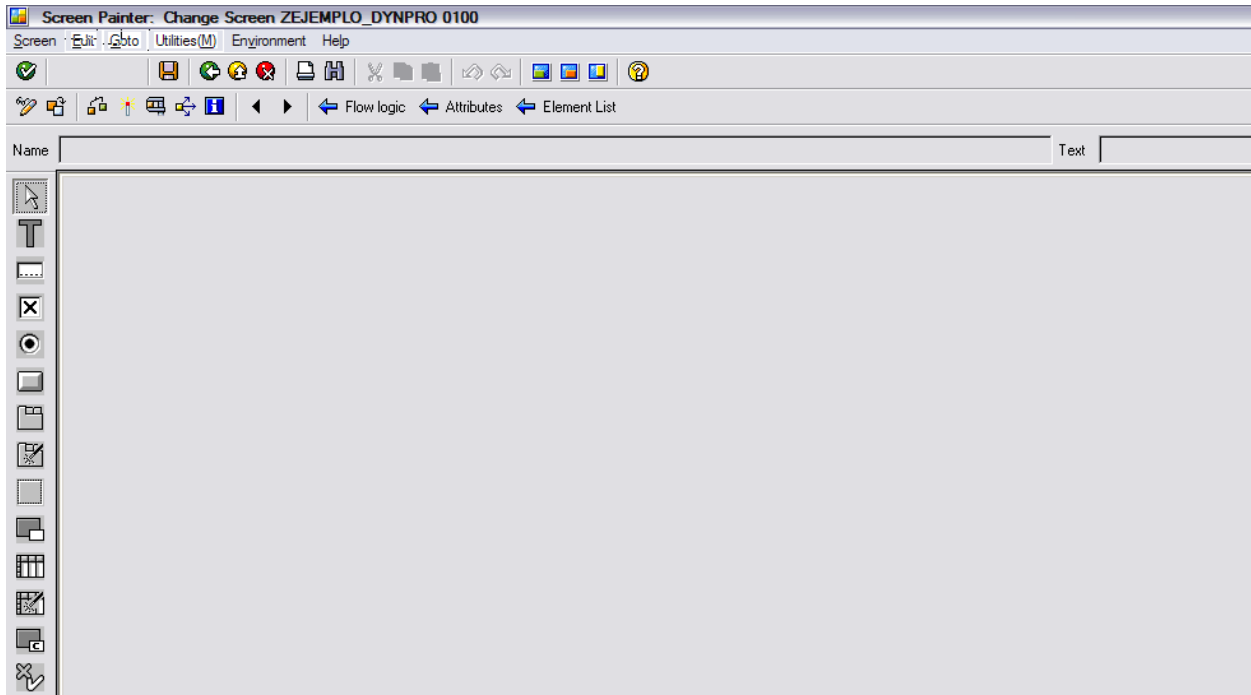
## Utilizando el Screen Painter

Así tras acceder al Screen Painter desde el ABAP/4 Workbench, tendremos que introducir el programa y el número de pantalla que deseamos actualizar o visualizar. Una vez hecho esto aparecerá el Editor de Pantallas 'Fullscreen Editor'.

Si estamos creando el dynpro por primera vez, nos pedirá los atributos de pantalla:

- Descripción
- Tipo de pantalla (normal, de selección, modal, Subscreen).
- Código de la siguiente pantalla.
- Campo donde se situará el cursor.
- Grupo de pantallas
- Tamaño máximo de la pantalla.

En el editor de pantallas podemos observar 3 áreas diferenciadas:



- La cabecera: Con datos sobre la pantalla y el campo que se está manteniendo en ese preciso instante.
- La barra de objetos (columna izquierda): Lista de los objetos que se pueden crear en la pantalla: Textos, entrada de datos, 'checkboxboxes', 'frames', 'screens', etc.
- El área de dibujo: Es el área donde se dibuja la pantalla que estemos diseñando.

## Creando objetos en la pantalla

Para dibujar un objeto en la pantalla tendremos que colocarlo en el área de trabajo y posteriormente definir sus características (atributos). Para ello tendremos que pulsar el botón correspondiente en la barra de objetos y marcar el área donde vamos a situar el objeto.

Si queremos cancelar la creación de un objeto pulsaremos el botón Reset de la misma barra de objetos.

## Objetos disponibles

Tenemos diversos objetos disponibles en pantalla para el diseño:

- Textos: Textos, literales, que son fijos en pantalla y no pueden ser manipulados por el usuario. Para considerar varias palabras como un mismo texto tendremos

que colocar un símbolo '-' entre ellas, ya que de otro modo las considerará como objetos de texto completamente distintos.

- **Objetos de entrada/ salida ('Templates'):** Son campos para introducir o visualizar datos. Pueden hacerse opcionales o obligatorios. Los caracteres de entrada se especifican con el símbolo '\_', pudiendo utilizar otros caracteres para dar formato a la salida. Por ejemplo una hora podemos definirla como: \_\_:\_\_:\_\_.
- **Radio-Buttons:** Son pequeños botones redondos, que permiten una entrada de dos valores sobre una variable (marcado o no marcado). Los podemos agrupar, de forma que la selección de uno implique que no se pueda seleccionar ningún otro.
- **Check-Boxes:** Es como un radio-button pero de apariencia cuadrada en vez de redonda. La diferencia respecto los radio-buttons deriva en su utilización en grupos, ya que se pueden seleccionar tantos check-boxes como se quiera dentro de un grupo.
- **Pushbuttons:** Es un botón de tipo pulsador. Se le asocia a un código de función, de forma que en el momento que se pulsa el Pushbutton se ejecute la función.
- **Frames (Cajas):** Son cajas que agrupan grupos de objetos dentro de una pantalla como por ejemplo un conjunto de radio-buttons.

Una vez situados los objetos sobre el área de trabajo, podremos modificar su tamaño, moverlos o borrarlos.

Todos los textos, pushbuttons pueden tener asociados iconos en su salida por pantalla. Los iconos tienen una longitud de dos caracteres y están representados por símbolos estándares. El icono será un atributo más de los campos y por lo tanto se definirán junto al resto de atributos del objeto.

## **Creando objetos desde el diccionario de datos.**

En la pantalla que estamos diseñando, podemos utilizar campos que ya existen en el diccionario de datos o declarados en el module pool- Para ello tendremos que seleccionar: Goto -> Dict / Program fields.

Aparecerá una pantalla de selección de datos en la que indicaremos el campo o la tabla de la cual queremos obtener datos. Además se deberá especificar, si queremos ver la descripción de cada campo (indicando la longitud) y si queremos realizar una entrada de datos ('template') de dicho campo por pantalla. Finalmente pulsaremos el botón correspondiente a crear desde el diccionario de datos o desde un programa.

Marcaremos el campo que queremos incorporar a nuestra pantalla y los copiaremos sobre el área de trabajo, situándolos en la posición que creamos más conveniente.

## **Definiendo los atributos individuales de cada campo**

Los atributos de los objetos definen sus características. Podemos mantener los atributos desde el mantenimiento de atributos de campo o desde listas de campos. Existen los atributos generales, de diccionario, de programa y de visualización

### **Atributos Generales:**

- Matchcode: Permite especificar un matchcode para la entrada de un campo.
- References: Especificamos la clave de la moneda en caso de que el campo sea de tipo cantidad (CURR o QUAN).
- Field type: Tipo de campo.
- Field Name: Nombre del campo. Con este nombre se identificarán desde el programa.
- Field text: Texto del campo. Si queremos utilizar un icono en vez de texto dejaremos este valor en blanco.
- With icon: si queremos utilizar iconos en entrada de datos ('templates').
- Icon name: Identifica el nombre de un icono para un campo de pantalla.
- Scrolling: Convierte un campo en desplegable, cuando su longitud real es mayor que su longitud de visualización.
- Quick Info Es el texto explicativo que aparece cuando pasarnos por encima de un icono con el mouse.
- Line: Especifica la línea donde el elemento aparecerá. El sistema completa este valor automáticamente.
- CI: Especifica la columna donde el elemento aparecerá. El sistema completa este valor automáticamente.
- Ht: Altura en líneas. El sistema completa este valor automáticamente.
- Dlg : Longitud del campo.
- Vlg: longitud de visualización.
- FctCode: Código de función (código de 4 dígitos). Atributo sólo para pushbuttons.



- FctType: Especifica el tipo de función asociado al código de función.
- Groups: Identifica grupos de modificación para poder modificar varios campos simultáneamente. Podemos asignar un campo a varios (4) grupos de modificación.

### **Atributos de Diccionario:**

- Format: Identifica el tipo de datos del campo. Determina el chequeo que realiza el sistema en la entrada de los datos.
- Frm DICT.: El sistema completa este atributo en el caso de que el campo lo hayamos creado a partir de un campo del diccionario de datos.
- Modific.: El sistema completa este campo si detecta alguna diferencia entre la definición del campo en el diccionario de datos y su utilización en pantalla.
- Conv. Exit: Se define el código de identificación de una rutina de conversión de datos no estándar.
- Param. ID: Código del parámetro SET/GET. (Ver siguiente atributo).
- SET param/GET param : Los parámetros SPA (Set Parameter) y GPA (Get Parameter), nos permiten visualizar valores por defecto en campos. Si marcamos el atributo SET param, el sistema guardará en un parámetro ID lo que entremos en este campo. Si marcamos el atributo GET param, el sistema inicializa el campo, con el valor del parámetro ID que tenga asignado en el atributo anterior.
- Up./lower: El sistema no convierte la entrada a mayúsculas.
- Foreign key: Si queremos que sobre el campo el sistema realice un chequeo de clave externa. (Hay que definir previamente las claves externas en el diccionario de datos).

### **Atributos de programa:**

- Input field: Campo de entrada.
- Output field: Permite visualización. Se puede utilizar en combinación con el anterior.
- Output only: Sólo visualización.
- Required field: Atributo para campos obligatorios. Se distinguen con un '?'.

- **Poss. Entry:** El sistema marca este atributo si hay un conjunto de valores para el campo. No es posible modificar el contenido del atributo.
- **Poss. Entries:** Indica cómo podemos ver la flecha de entradas posibles en un campo.
- **Right-justif:** Justifica cualquier salida del campo a la derecha.
- **Leading zero:** Rellena con ceros por la izquierda en el caso de salidas numéricas.
- **\*-entry:** Permite la entrada de un asterisco en la primera posición de un campo. Si se introduce un \* se podrá hacer un tratamiento en un módulo: FIELD MODULE ON \*-INPUT.
- **No reset:** Cuando activamos este atributo, la entrada de datos no podrá ser cancelada mediante el carácter !.

### **Atributos de visualización:**

- **Fixed font :** Visualizar un campo de salida en un tamaño fijo (no proporcional). Sólo se puede utilizar en campos Output only.
- **Briglit :** Visualiza un campo en color intenso.
- **Invisible:** Oculta un campo.
- **2-dimens:** Visualiza un campo en dos divisiones en vez de en tres.

### **Lógica de proceso de una pantalla**

### **Introducción a la lógica de proceso**

Una vez hemos definido gráficamente las pantallas, será preciso escribir una lógica de proceso para cada una de ellas, pasándose a denominar dynpros.

Para introducir la lógica de proceso de las pantallas, utilizaremos una versión especial del editor de ABAP/4. Goto -> Flow logic.

La lógica de proceso de las pantallas tiene una estructura determinada, y utilizan comandos y eventos propios de manejo de pantallas, similares a los utilizados en ABAP/4.

Consistirá en dos eventos fundamentales:

- PROCESS BEFORE OUTPUT (PBO).
- PROCESS AFTER INPUT (PAI).

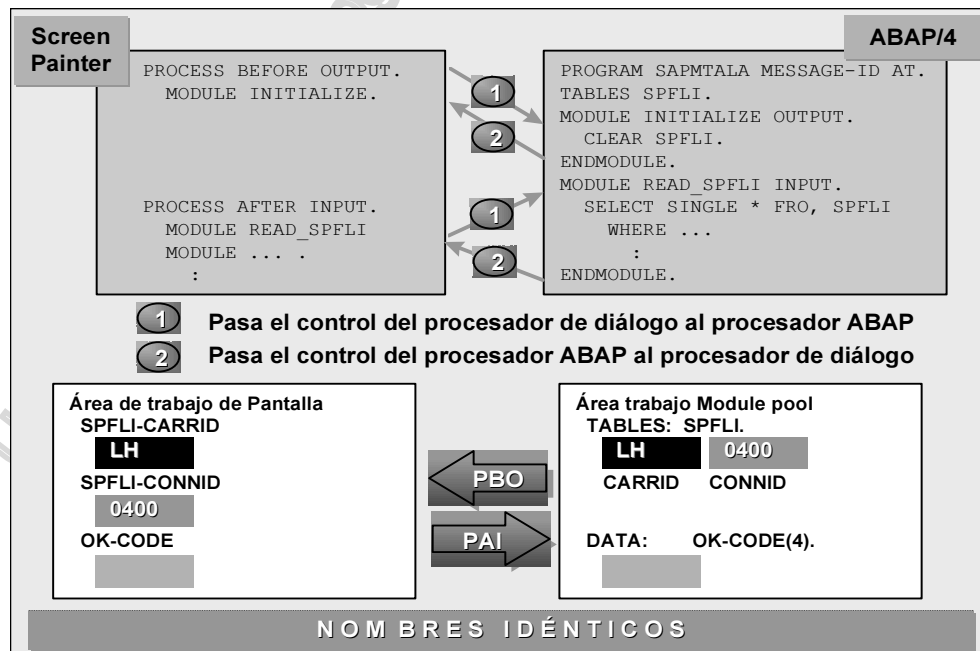
El flujo de control de una transacción, se procesa en dos eventos principales: PROCESS BEFORE OUTPUT (PBO) y PROCESS AFTER INPUT (PAI). Dentro de cada uno de ellos, existen subeventos que son invocados por la acción del usuario.

El evento PBO se dispara antes de que la pantalla sea presentada al usuario (por ejemplo cuando invocamos por primera vez a una transacción). Durante el mismo, el programador puede controlar qué datos se presentarán al usuario.

El evento PAI, se acciona cuando el usuario completa el ingreso de datos y subsecuentemente lanza una acción. Acciones como presionar ENTER, clickear en un Push Button de la pantalla o barra de tareas, o elegir un comando del menú activan el evento PAI. Durante este evento, el programador puede procesar los datos ingresados por el usuario y prepararlos para el evento PBO, que siempre se ejecuta inmediatamente.

## Procesamiento de Eventos

El sistema procesa los módulos de un evento en forma secuencial:



En cada módulo el control pasa del procesador de diálogo al procesador ABAP/4. Después del procesamiento el control es regresado al procesador de diálogo.

El comando PBO indica el comienzo de la sección PBO en el flujo de control del Module Pool, todas las sentencias bajo este comando son ejecutadas antes de que la pantalla se presente al usuario. Aunque el evento PBO siempre se ejecuta primero cuando una transacción es invocada, cualquier código de inicialización que se debe ejecutar cada vez que se muestra una pantalla debe situarse aquí. Se puede utilizar código ABAP para detectar el primer ingreso a una transacción y no a subsecuentes visualizaciones de la misma pantalla.

Cuando todos los módulos de PBO han sido procesados, el contenido de los campos del área de trabajo ABAP/4 es copiado a un campo con nombre idéntico en el área de trabajo de pantalla.

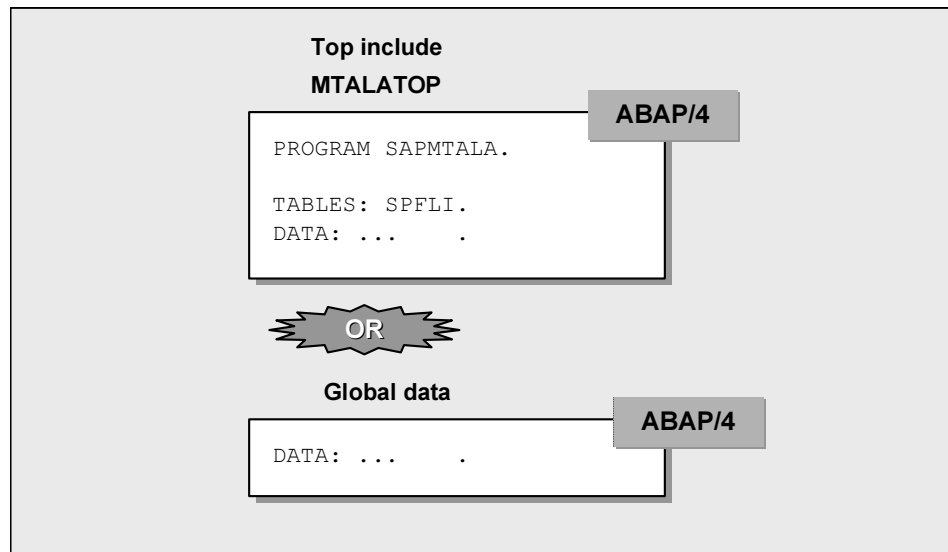
El comando PAI indica el comienzo de la sección. Este evento se dispara cuando el usuario presiona ENTER- aunque se puede ignorar y no procesar ningún dato cada vez que esto ocurre. Un hábito común es presionar ENTER para desplazarse por los campos, por lo que es importante ignorar el evento PAI en estos casos. A través de este evento se puede detectar qué acciones el usuario puede estar realizando dentro de la aplicación. Decidir qué tipo de tecla o acción se elegirá para procesar el evento PAI, depende del tipo de transacción que se esté desarrollando. Las pantallas simples pueden ser accionadas por el uso de la tecla ENTER, pero para algunas más complejas es mejor evitarlo. En su lugar se pueden utilizar botones de pantalla que le dirían al usuario exactamente qué hacer, como los botones “Copiar”, “Borrar”, “Siguiente”, etc..

Antes de que el módulo PAI sea procesado, el contenido de los campos del área de trabajo de pantalla es copiado a los campos nombrados idénticamente en el área de trabajo de ABAP/4.

Los subeventos invocados dentro del PBO y PAI que se utilizarán en una transacción de diálogo son invocados a través de código ABAP por el comando MODULE. Este bloque de código ABAP se codifica de la misma forma que un reporte ABAP, con todas las sentencias inherentes al lenguaje de programación.

La lógica e invocación de estos eventos determinan el comportamiento de un transacción durante el procesamiento de los datos, el código asociado a cada acción se escribe en ABAP y es propio de cada pantalla / dynpro del module Pool.

## Definición de campos en el Module Pool

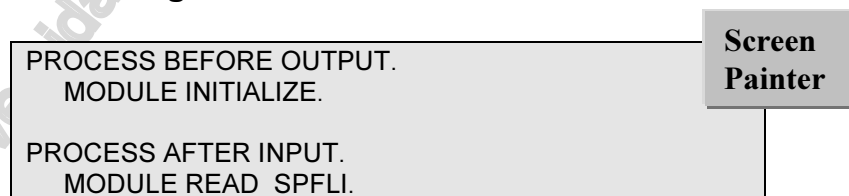


En el procesamiento en diálogo, los datos son transferidos entre pantallas y programas ABAP/4. El sistema ejecuta esta comunicación automáticamente, pero es necesario usar nombres idénticos en las pantallas y en el module pool.

Se definen los campos relevantes como datos globales en el programa Top Include. Al hacer esto, se procesa el programa Top Include, ó las secciones concernientes a las estructuras de datos globales, ó las estructuras de Diccionario.

## Definición del Flujo de Control

### Introducción a la Lógica de Proceso



Una vez hemos definido gráficamente las pantallas, será preciso escribir una **lógica de proceso** para cada una de ellas, pasándose a denominar Dynpros.

Para introducir la lógica de proceso de las pantallas, utilizaremos una versión especial del editor de ABAP/4. **Goto -> Flow Logic.**

La lógica de proceso de las pantallas tienen una estructura determinada, y utilizan comandos y eventos propios de manejo de pantallas, similares a los utilizados en ABAP/4.

Consistirá en dos eventos fundamentales:

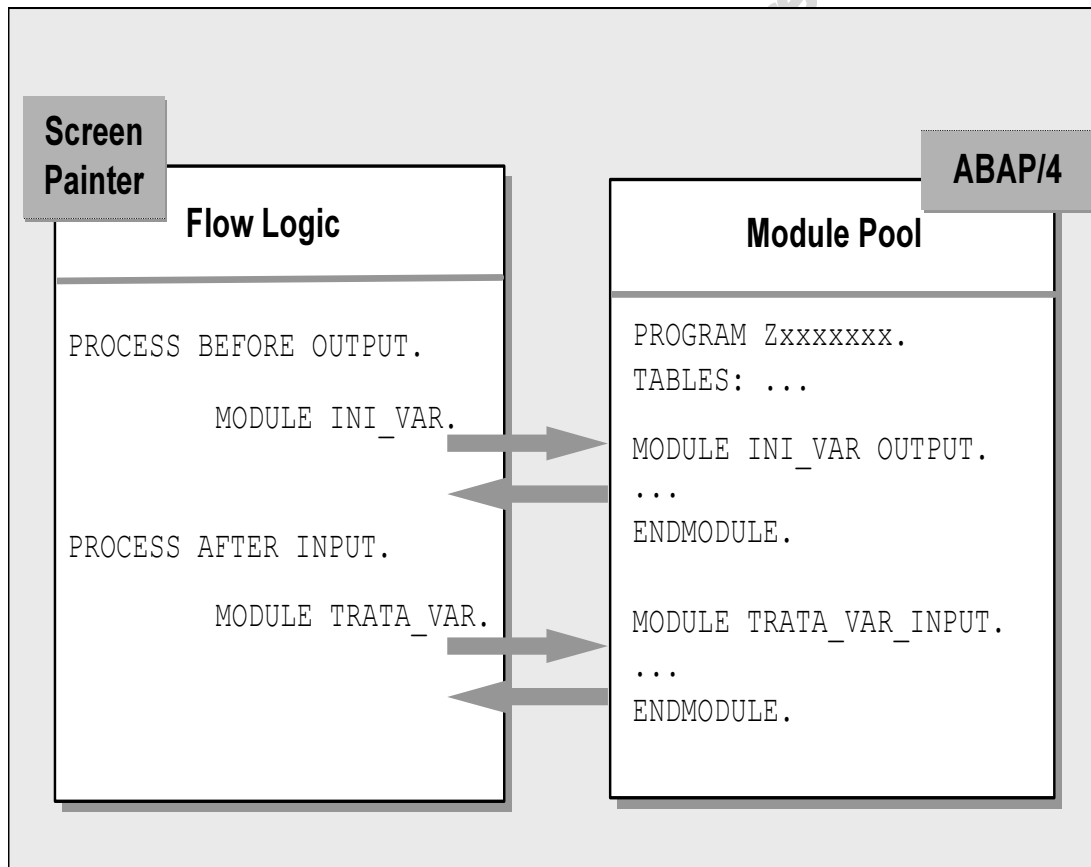
**PROCESS BEFORE OUTPUT (PBO).**

**PROCESS AFTER INPUT (PAI).**

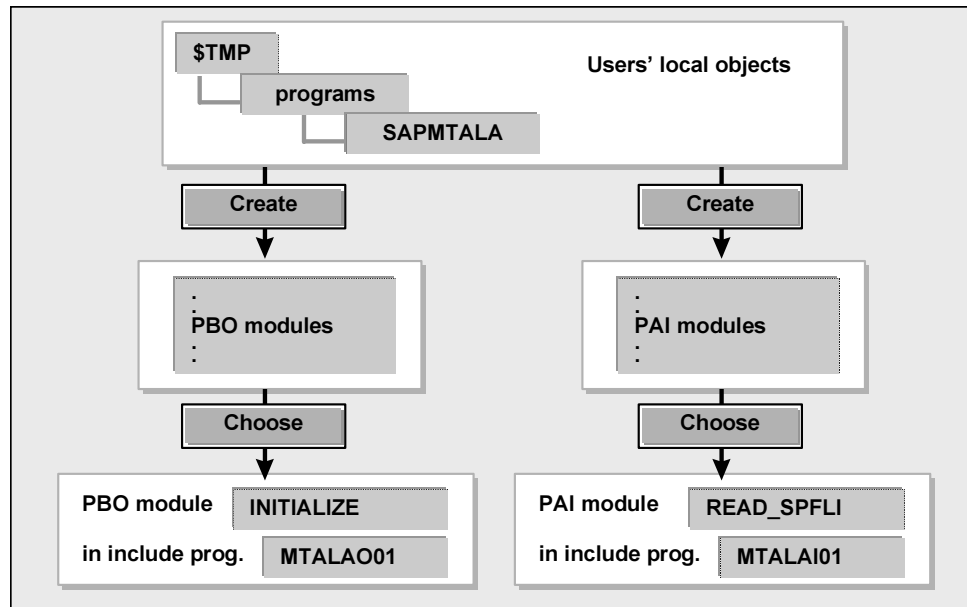
El **PBO** será el evento que se ejecutará previamente a la visualización de la pantalla, mientras que el **PAI**, se ejecutará después de la entrada de datos del usuario en la pantalla.

Además de estos dos eventos obligatorios existen eventos para controlar las ayudas, **PROCESS ON HELP-REQUEST (POH)**, y para controlar los valores posibles de un campo **PROCESS ON VALUE REQUEST (POV)**.

Desde la lógica de proceso de las pantallas no se actualizan datos, únicamente se llaman a los módulos del Module Pool que se encarga de esta tarea.



## Creación de Módulos ABAP/4



Para llamar a un módulo utilizaremos la sentencia **MODULE**.

**MODULE <nombre\_módulo>.**

Si se selecciona un módulo con doble click, el sistema crea las instrucciones **MODULE ... END MODULE** en el programa incluido correspondiente.

Si el include no existe, el sistema creará uno automáticamente si así se desea. Esto también inserta una instrucción **INCLUDE** en el programa principal, para hacer referencia a este programa incluido

En el Module Pool el código del módulo empezará con la sentencia:

**MODULE <nombre\_módulo> OUTPUT.**

En el caso de ser un módulo llamado desde el PAI será:

**MODULE <nombre\_módulo> INPUT.**

```

MODULE INITIALIZE OUTPUT.
  CLEAR SPFLI.
ENDMODULE.

MODULE READ_SPFLI INPUT.
  SELECT SINGLE * FROM SPFLI
  WHERE CARRID = SPFLI-CARRID
  AND   CONIID = SPFLI-CONNID.
  .
  .
ENDMODULE.
  
```

## Process Before Output (PBO)

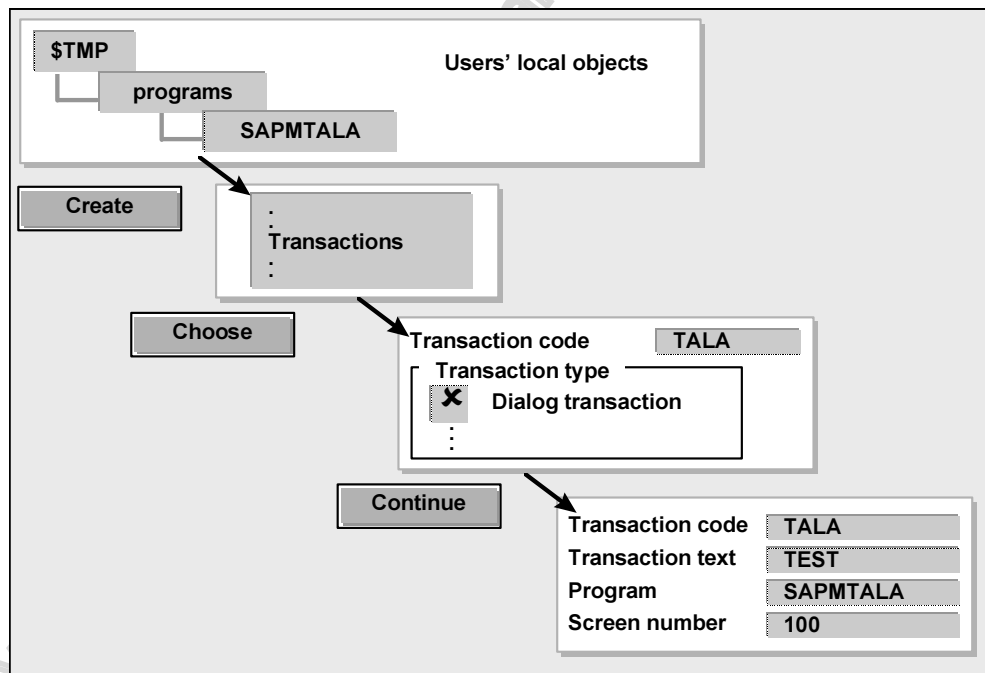
En el módulo del PBO los inicializaremos al valor que queramos. Si no inicializamos explícitamente, el sistema le asignará su valor inicial por defecto, al no ser que lo hayamos definido como parámetro SPA/GPA.

En el módulo PBO, indicaremos todos los pasos que queremos realizar antes de que la pantalla sea visualizada, como por ejemplo inicializar los campos de salida. Esta inicialización se realizará en un módulo independiente dentro del Module Pool.

## Process After Input (PAI)

El PROCESS AFTER INPUT se activa cuando el usuario selecciona algún punto de menú, pulsa alguna tecla de función o pulsa ENTER. Si alguno de estos eventos ocurre, el PAI de la pantalla necesitará responder apropiadamente a la función seleccionada por el usuario.

## Definiendo la llamada (Códigos de Transacción)



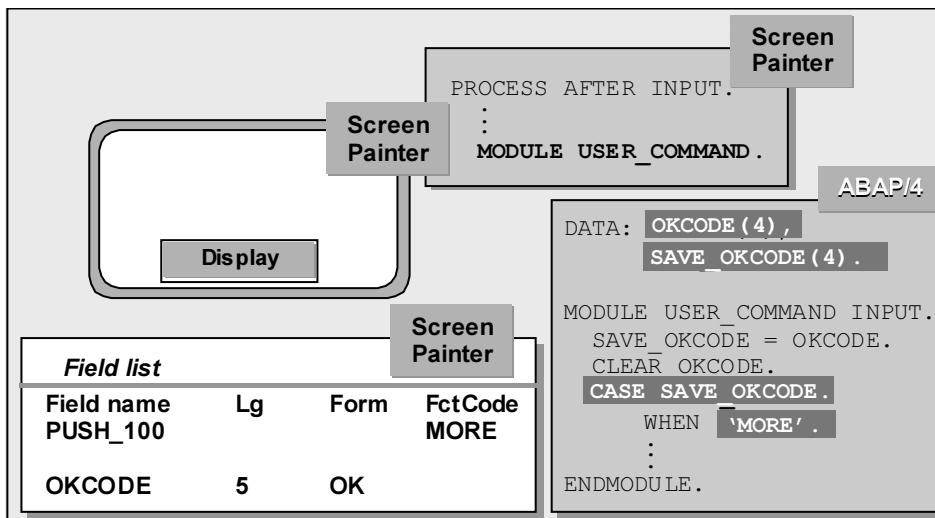
Se puede iniciar un programa de diálogo ABAP/4 ("transacción"), especificando un código de transacción, el cual podremos definirlo desde la transacción SE93.

Las transacciones de cliente deben iniciar con "Z" ó "Y".



El sistema almacena las especificaciones en la tabla TSTC. (En vez de crear una transacción desde la lista de objetos, se puede modificar la tabla directamente).

## Leyendo códigos de función en programas



Cuando el usuario de una transacción, pulsa una tecla de función, un punto de menú, un pushbutton, un icono o simplemente la tecla ENTER, los datos introducidos en la pantalla se pasan a los módulos del PAI para ser procesados junto a un código de función que indicará qué función ha solicitado el usuario.

En el Screen Painter, será necesario crear un campo de tipo código de función, de longitud 4 bytes, que normalmente aparece al final de la lista de campos de cada pantalla. Tradicionalmente a este campo se le denomina OK\_CODE, y será declarado en nuestro module Pool como cadena de caracteres de 4 posiciones:

En la lógica de proceso de cada pantalla, tendremos que realizar el tratamiento del OK\_CODE. Para ello utilizaremos un modulo que deberá ser el último del evento PAI, es decir que se ejecutará una vez que todos los datos de entrada han sido validados correctamente.

Una vez procesado el módulo de función, borraremos el contenido del OK\_CODE, inicializándolo para la próxima pantalla. Podemos guardar el contenido del OK\_CODE en una variable intermedia e inicializarlo inmediatamente

## Process Before Output (PBO)

En el Módulo PBO, indicaremos todos los pasos que queremos realizar antes de que la pantalla sea visualizada, como por ejemplo inicializar los campos de salida, Esta inicialización se realizará en un módulo independiente dentro del module pool-

Para llamar a un módulo utilizaremos la sentencia **MODULE**.

**MODULE <nombre modulo>.**

Ejemplo:

**PROCESS BEFORE OUTPUT.**

**\***

**MODULE inicializar\_campos.**

**\***

**PROCESS AFTER INPUT.**

En el Module pool el código del módulo empezará con la sentencia:

**MODULE <nombre\_modulo> OUTPUT.**

En el caso de ser un módulo llamado desde el PAI será:

**MODULE <nombre-inodulo> INPUT.**

Dentro del module pool declararemos los campos de pantalla que vayamos a utilizar y en el módulos del PBO los inicializamos al valor que queramos. Si no inicializamos explícitamente, el sistema le asignará su valor inicial por defecto, a no ser que lo hayamos definido como parámetro SPA / GPA.

Además de inicializar los campos de entrada de datos, el evento PBO, puede ser utilizado para:

- Seleccionar el Status (menú) o el título de los menús con:

**SET PF-STATUS <GUI\_status>.**

**SET TITLEBAR <título> WITH <p1> <p2> <p3> <p4>.**

- Desactivar funciones de un menú con :

**SET PF-STATUS <GUI-status> EXCLUDING <function-code>.**

Si se quiere desactivar más de una función, se le indicará en el EXCLUDING, una tabla interna con los códigos de función que queremos desactivar. La estructura de la tabla interna será:

**DATA: BEGIN OF tabint OCCURS 20,  
                    FUNCTION (4),  
                    END OF tabint.**

- Modificar los atributos de la pantalla en tiempo de ejecución. Para ello disponemos de la tabla **SCREEN**, donde cada entrada se corresponde a un campo de la pantalla que puede ser leída y modificada con LOOP's y MODIFY'S.

La estructura de la tabla SCREEN es :

NAME	30	Nombre del campo de pantalla.
GROUP1	3	Campo pertenece al grupo de campo 1.
GROUP2	3	Campo pertenece al grupo de campo 2.
GROUP3	3	Campo pertenece al grupo de campo 3.
GROUP4	3	Campo pertenece al grupo de campo 4.
ACTIVE	1	Campo visible y listo para entrada de datos.
REQUIRED	1	Entrada de datos obligatoria.
INPUT	1	Campo apto para entrada de datos.
OUTPUT	1	Campo sólo visualización.
INTENSIFIED	1	Campo en color intensificado
INVISIBLE	1	Campo invisible.
LENGTH	1	Longitud de salida reducida.
REQUEST	1	Campo con Valores posibles disponibles
VALUE_HELP	1	Campo con Help disponible.
DISPLAY_3D	1	Campo tridimensional

- Inicializar radio-buttons y check-boxes. Los declararemos como caracteres de longitud 1 y los inicializamos como:

Radio1 = 'X'  
Check1 = 'X'.

En el caso de los radio-buttons, que pueden funcionar en grupo, de forma que la activación de un botón desactive otro, no será necesario codificar este comportamiento manualmente, ya que lo realizará el sistema automáticamente.

Para definir un grupo de radio-buttons, seleccionamos los botones del grupo y marcar: **Edit -> Radio Button Group**. En caso de utilizar el editor Alfanumérico: Seleccionamos el primer objeto de grupo, escoger **.Edit -> Graphical element**, Seleccionar el último objeto del grupo y pulsar: **Define Graph group**.

- Para utilizar subscreens será necesario realizar las llamadas a las mismas en tiempo de PBO y de PAI.

PROCESS BEFORE OUTPUT.

CALL SUBSCREEN <área> INCLUDING <programa> <screen>.

PROCESS AFTER INPUT.

CALL SUBSCREEN <área>.

Donde: <área>: es el nombre que le hemos dado al área de subscreen en el Screen Painter.

<Programa>: Es el nombre del programa que controla el subscreen.

**<screen>** : Es el número de pantalla.

La subscreen se ejecutará como una pantalla normal y corriente con su PBO y PAI correspondiente que son llamados desde el PBO y el PAI de la pantalla principal.

- Para manipular la posición del cursor.

Por defecto el sistema sitúa el cursor en el primer campo de entrada de la pantalla, pero es posible que queramos situarlo en otro campo:

**SET CURSOR FIELD** <nombre\_campo>.

También es posible que en algún momento sea interesante conocer en qué campo está situado el cursor. Para hacer esto utilizamos:

**GET CURSOR FIELD** <var\_campo>.

## **Process After Input (PAI)**

El PROCESS AFTER INPUT se activa cuando el usuario selecciona algún punto de menú, pulsa alguna tecla de función o pulsa ENTER. Si alguno de estos eventos ocurre, el PAI de la pantalla necesitará responder apropiadamente a la función seleccionada por el usuario.

## **La validación de los datos de entrada**

Una de las funciones más importantes de Process After Input, es la de **validar los datos de entrada** de la pantalla antes de ser usados. Existen dos tipos de validación de los datos de entrada: Un **chequeo automático** realizado por el sistema y un **chequeo manual** programado con el comando **FIELD** de la lógica de proceso de Dynpros.

### **Verificación automática**

El sistema realiza automáticamente una serie de chequeos de los datos de entrada, antes de procesar el evento PAI.

## Verificación de formato

Field list	
Field name	Format
DATE	DATS
...	
Amount	DEC

Date: 30.02.1996  
Amount:   
E: Invalid date

Date:   
Amount: 12A3  
E: Please enter numeric value

El procesador de diálogo valida las entradas de acuerdo a los atributos de cada campo. Si el sistema detecta un valor incorrecto, despliega un mensaje de error y vuelve a mostrar los campos para su nueva entrada.

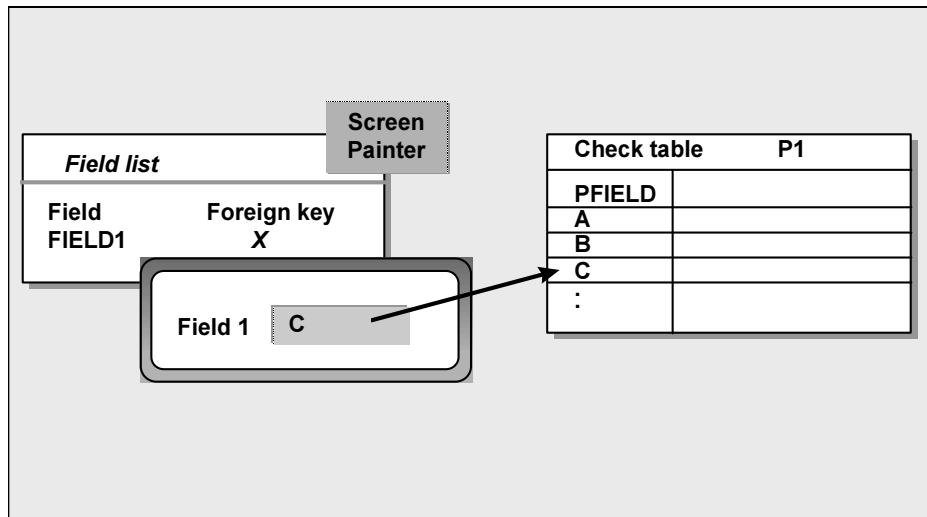
## Verificación de campos obligatorios

Field list	
Field name	OBLIGATORY
PLANETYPE	X

?

Al momento de que algún campos de la pantalla se le asigna el atributo de que es obligatorio, el procesador de diálogo no continua con el proceso, al menos que todos los campos obligatorios tengan algún valor.

## Verificación de llaves foráneas



Una verificación de clave foránea es procesada solo si un campo de pantalla se refiere a un campo del Diccionario para el cual se ha definido una tabla de verificación.

Al mismo tiempo, la funcionalidad de la tecla F4 es activada. Esto significa que las posibles entradas para un campo son desplegadas.

## Verificación de valores fijos

En el Diccionario ABAP/4, se pueden definir los valores fijos para los dominios.

Si se define un campo de pantalla con referencia a un dominio con valores fijos, ocurre lo siguiente:

- Los valores fijos son desplegados si el usuario presiona la tecla F4 para ver los posibles valores para el campo de entrada.
- El procesador de diálogo verifica los valores introducidos en el campo contra el conjunto de valores fijos del Dominio correspondiente.

## Verificación manual en Module Pool.

Además del chequeo automático es posible realizar una validación más extensa de los valores de entrada a los campos valores de entrada a los campos con las instrucciones **FIELD** y **CHAIN** de la lógica de proceso del Screen Painter.

Con **FIELD** podemos validar individualmente cada campo de forma que en caso de error, la siguiente entrada de datos sólo permitirá introducir el campo erróneo sobre el que estamos utilizando la instrucción **FIELD**.

Dependiendo del tipo de sentencia **FIELD** que utilicemos, el mecanismo de chequeo se realizará en la lógica de proceso del Screen Painter o en un módulo ABAP/4.



Es posible realizar distintas validaciones de un campo de entrada dependiendo de la fuente con la que contrastamos los valores posibles. Así podemos verificar el contenido de un campo comparándolo con una tabla de la base de datos, con una lista de valores, o realizando la validación en un módulo del Module Pool.

Para verificar un campo contra la base de datos utilizamos:

```
FIELD <campo_pantalla> SELECT FROM <tabla>  
WHERE <campo_tabla> = <entrada_campo_pantalla>.
```

Si no se encuentran registros en el Diccionario de Datos el sistema emite un mensaje de error estándar. Existe una versión ampliada de la instrucción anterior que permite enviar mensajes o warnings en caso de que encuentre o no registros:

```
FIELD <campo_pantalla> SELECT * FROM <tabla>  
WHERE <condición>  
WHENEVER (NOT) FOUND SEND  
ERRORMESSAGE / WARNING <número>  
WITH <campo-texto>.
```

Para verificar un campo respecto a una lista de valores utilizamos:

```
FIELD <campo_pantalla> VALUES (<lista_valores>).
```

Donde <lista\_valores> puede ser:

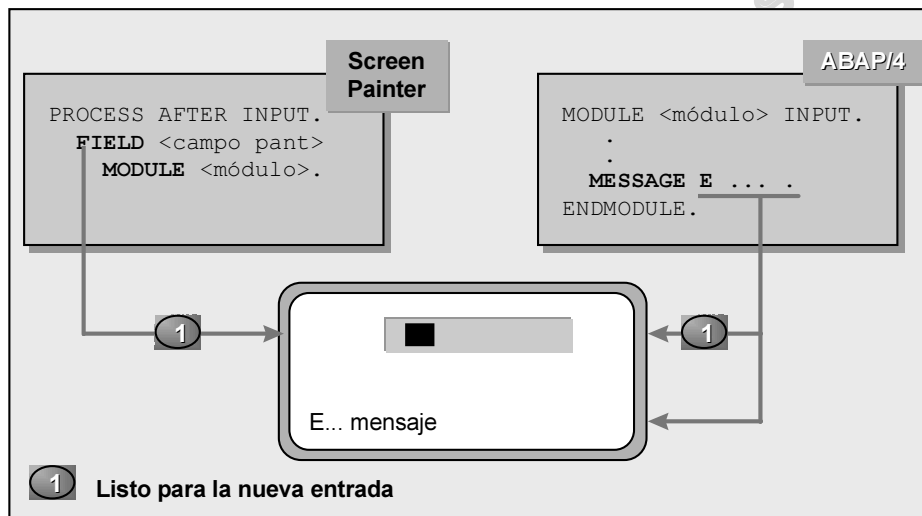
(‘<valor>’)

```
(not'<valor>')  
( '<valor 1>', '<valor 2>', ...NOT'<valor n>')  
(BETWEEN '<valor 1>' AND '<valor 2>')  
(NOT BETWEEN '<valor 1>' AND '<valor 2>')
```

Si el valor entrado por el usuario no corresponde a ningún valor de la lista el sistema emite un mensaje de error.

Para verificar un campo en un módulo de ABAP/4 utilizamos:

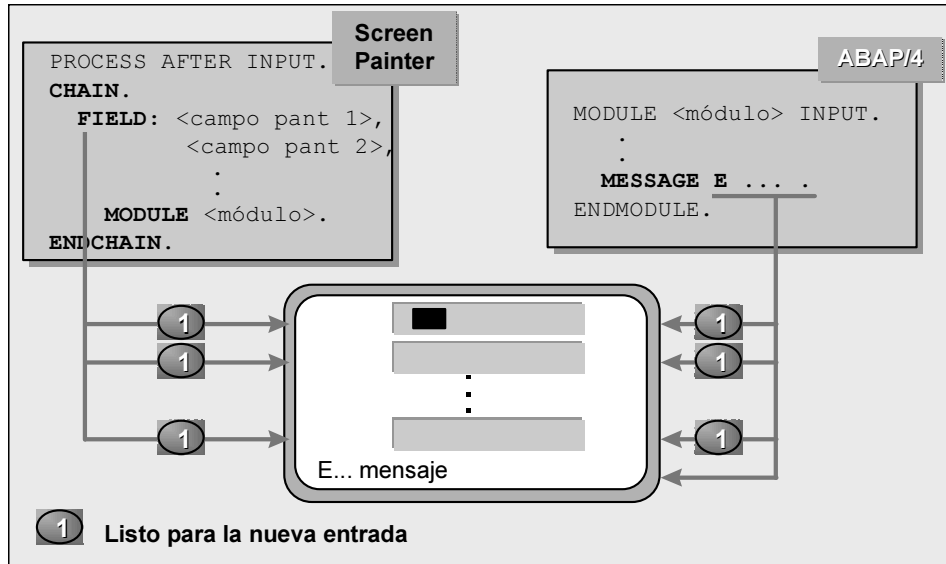
```
FIELD <campo_pantalla> MODULE <módulo_ABAP/4>.
```



Si el módulo resulta con un error (E) o un mensaje de advertencia (W), la pantalla es desplegada nuevamente pero sin procesar los módulos **PBO**. El texto del mensaje es mostrado, y solo el campo que ocasionó el error estará disponible para introducir datos nuevamente.

La instrucción **CHAIN...ENDCHAIN** encierra un conjunto de instrucciones **FIELD**, en caso de error en la entrada de alguno de ellos, todos los campos del **CHAIN** se podrán modificar, mientras que los que no pertenezcan al **CHAIN** estarán bloqueados para la entrada de datos.





```

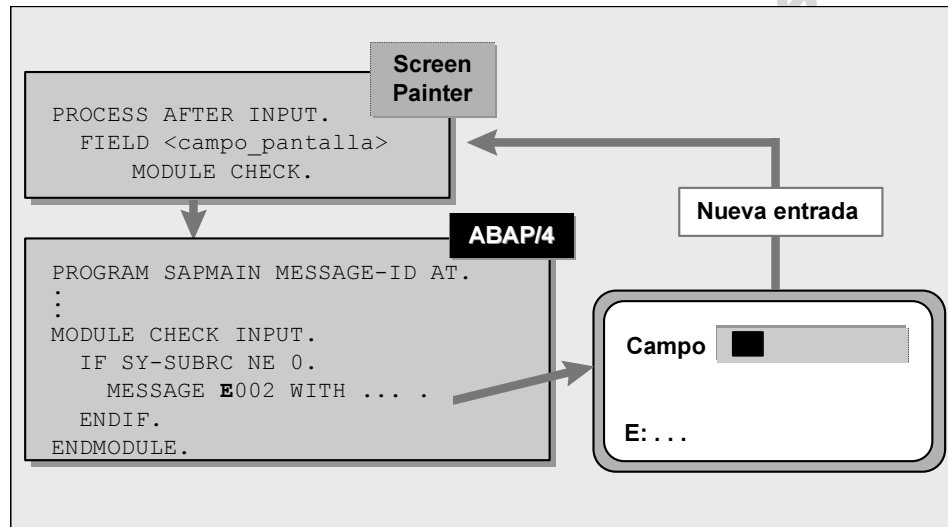
CHAIN.
  FIELD <campo 1>, <campo 2>, <campo 3>.
  MODULE <mod1>.
  MODULE <mod2>.
ENDCHAIN.
    
```

```

CHAIN.
  FIELD <campo 1>, <campo 2>.
  MODULE <mod1>.
  FIELD <campo 3> MODULE <mod2> ON CHAIN INPUT.
ENDCHAIN.
    
```

## Mensajes en pantalla

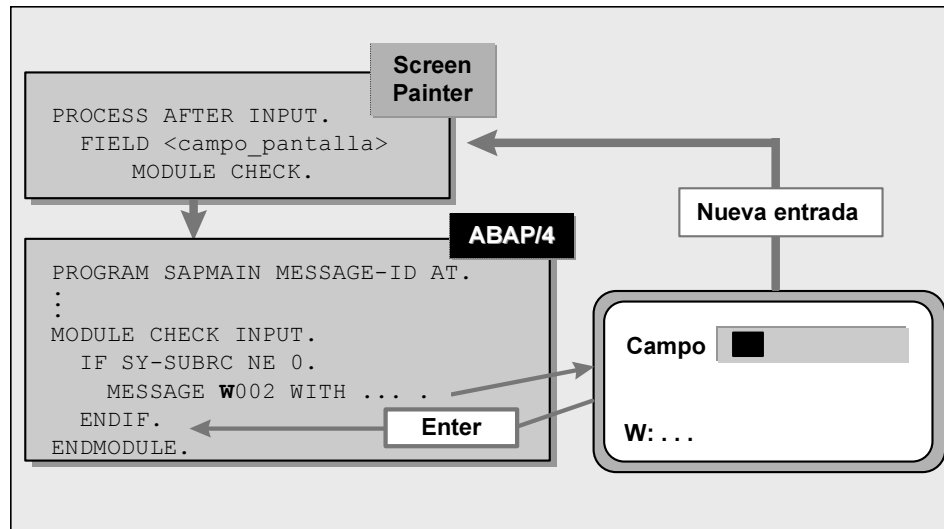
### Mensaje de Error



El texto de un mensaje de error ('E') es desplegado en la pantalla actual.

Todos los campos de pantalla asignados al módulo correspondiente (instrucción FIELD) se vuelven disponibles para introducir información de nuevo. El sistema obliga al usuario a reintroducir datos.

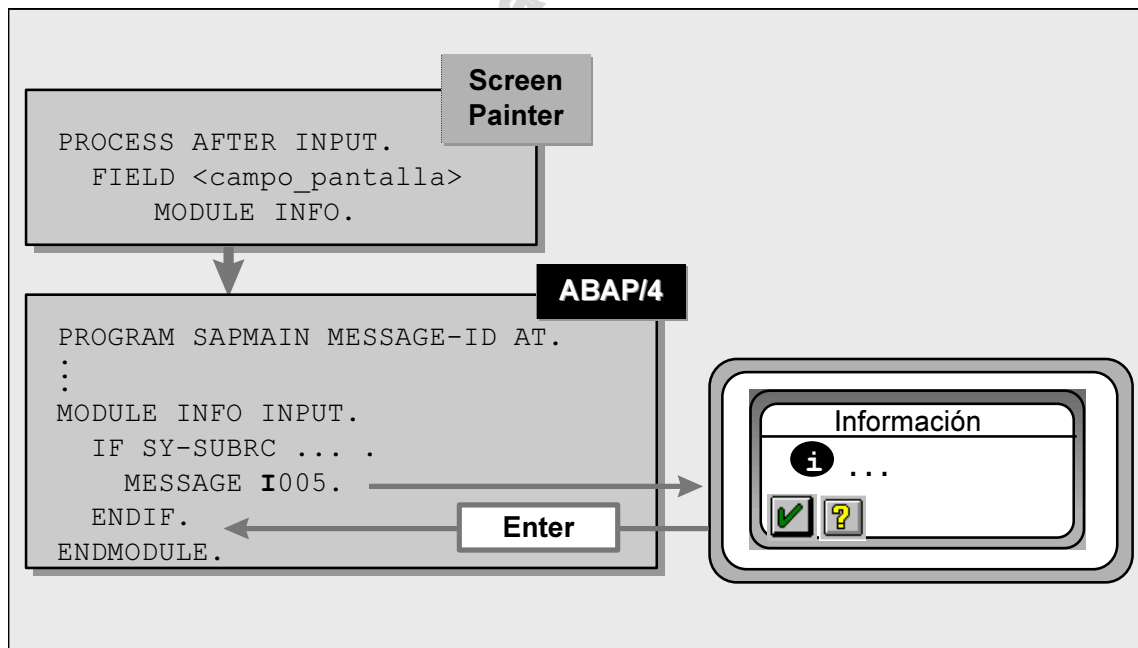
## Mensaje de Advertencia (Warning)



El texto del mensaje de advertencia ('W') es desplegado en la pantalla actual.

Todos los campos de pantalla asignados al módulo correspondiente (instrucción FIELD) se vuelven disponibles para introducir información de nuevo. El usuario puede reintroducir los datos o ignorar el mensaje de advertencia presionando la tecla ENTER.

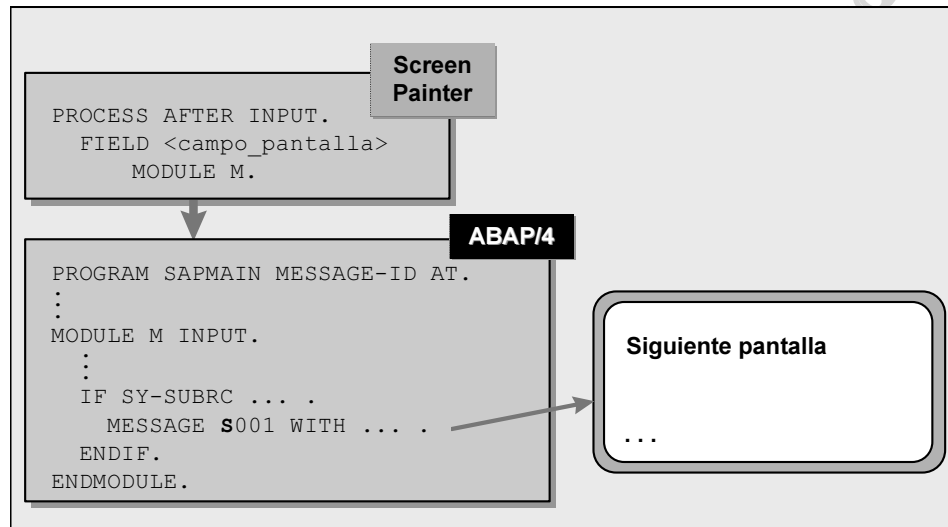
## Mensaje de Información



El texto de un mensaje de información ('I') es desplegado en la pantalla actual.

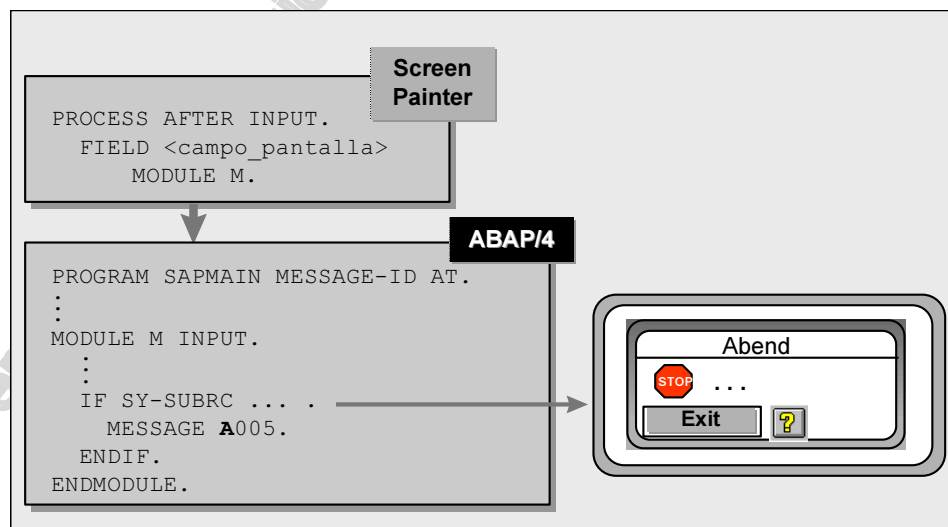
El proceso de la pantalla actual es suspendido. Después de que el usuario presione la tecla ENTER, el programa continúa con su ejecución normal desde el punto donde fue suspendido.

### Mensaje de Buen resultado



Un mensaje de texto de buen resultado ('S') es desplegado en la pantalla siguiente a la actual.

### Mensaje de Interrupción (Abend)



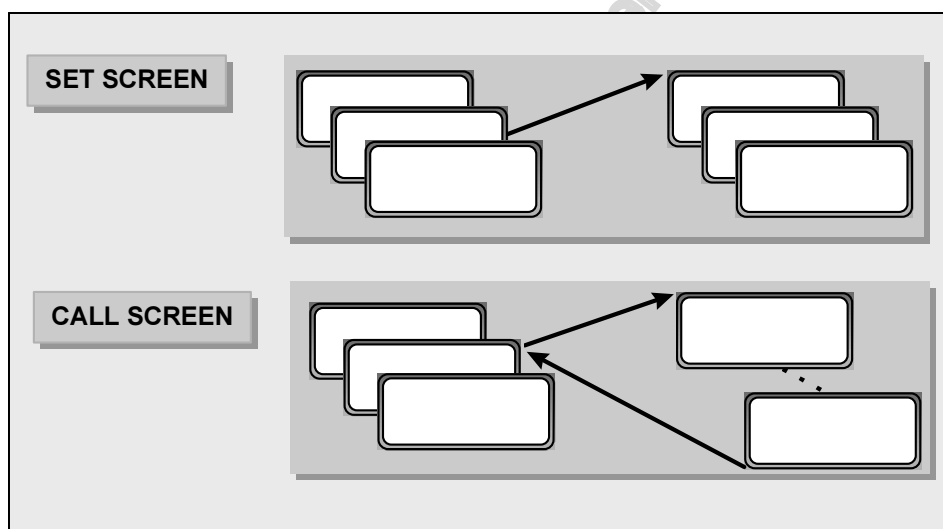
El texto de un mensaje de Interrupción ('A') es desplegado en la pantalla actual.

Después de que el usuario presione la tecla ENTER, el proceso actual es terminado y el proceso regresa a la pantalla inicial.

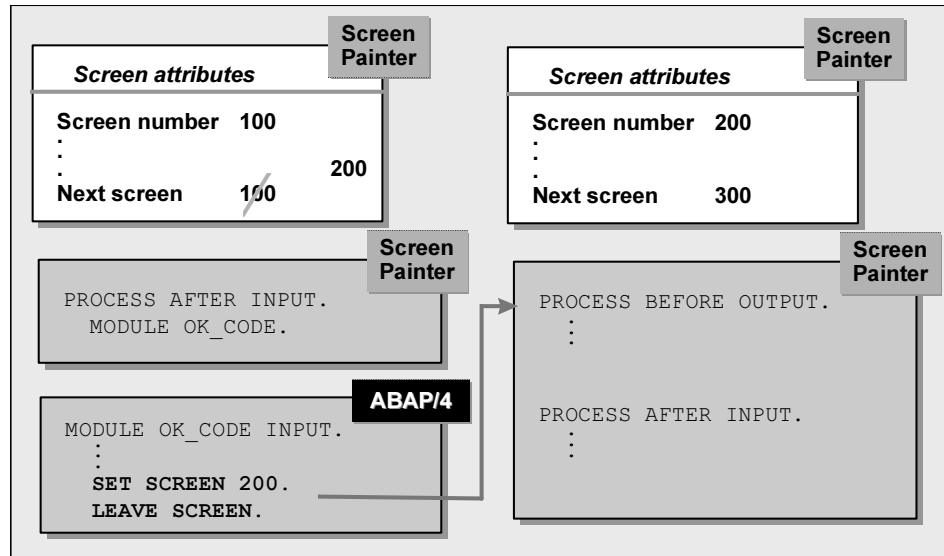
## Secuencia dinámica de pantallas

### Introducción

Desde una transacción podemos ir controlando el flujo de pantallas de la misma, llamar a otras transacciones o reportes.



### Configuración dinámica de la siguiente pantalla



Por defecto, cuando acaben los módulos del evento PAI, el sistema saltará a la pantalla que indique el atributo **Next Screen** de la pantalla en ejecución. Es posible modificar el atributo de la próxima pantalla con la instrucción **SET**.

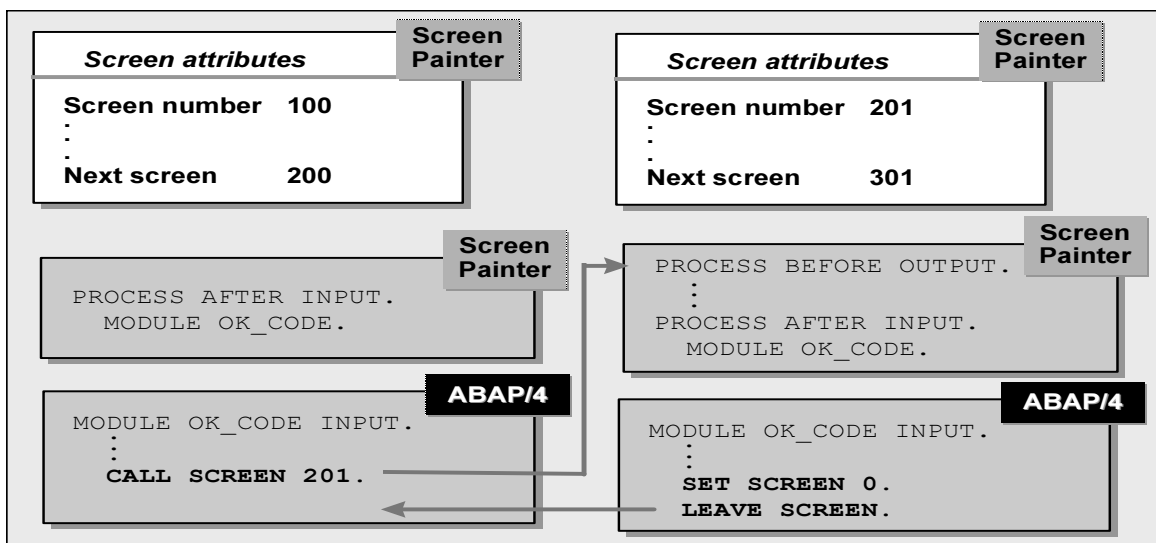
**SET SCREEN <no.\_pantalla>.**

La instrucción **SET SCREEN nnnn** reescribe temporalmente la siguiente pantalla a procesar. La pantalla **nnnn** debe ser una pantalla del mismo "module pool".

La pantalla siguiente es procesada después de procesar la pantalla actual, o al menos que se termina la ejecución de la pantalla actual con la instrucción **LEAVE SCREEN**. Al encontrar esta instrucción, se ejecuta la pantalla siguiente en forma inmediata.

Si se desea terminar el procesamiento de la pantalla actual e ir directamente a la pantalla siguiente en una sola instrucción, se puede usar el estatuto **LEAVE TO SCREEN nnn**.

## Inserción de una o más pantallas



La instrucción **CALL SCREEN nnnn** interrumpe el procesamiento de la pantalla actual para procesar la pantalla nnnn y las pantallas subsecuentes.

La pantalla llamada con esta instrucción deberá ser una pantalla del mismo "module pool".

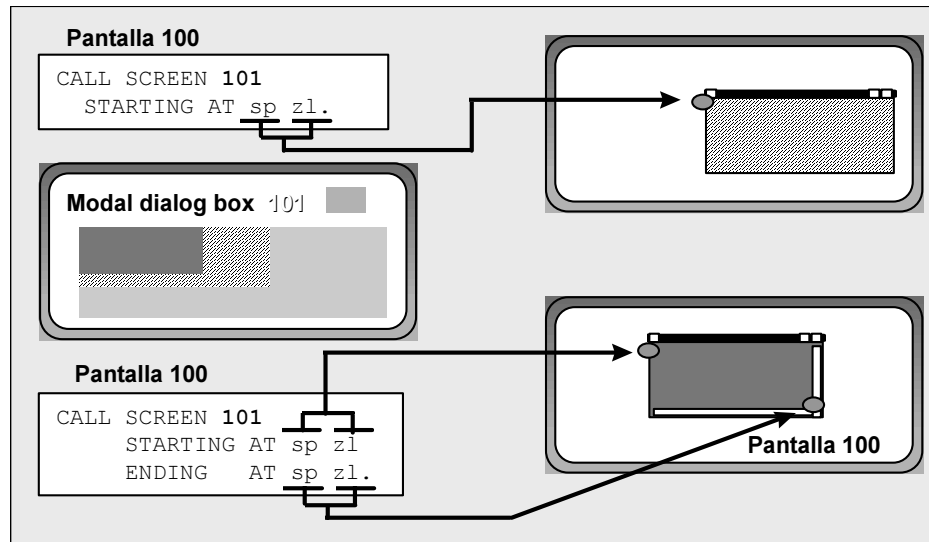
Cualquiera de las instrucciones: **SET SCREEN 0**, **LEAVE SCREEN**, **LEAVE TO SCREEN 0**, regresa el control a la locación donde fue ejecutada la instrucción **CALL SCREEN nnnn**.

Si se usa cualquiera de estas instrucciones cuando no se está en el modo de llamada, el programa termina. Por ejemplo el regresar al lugar donde este programa fue llamado (ver también la instrucción **LEAVE PROGRAM**).

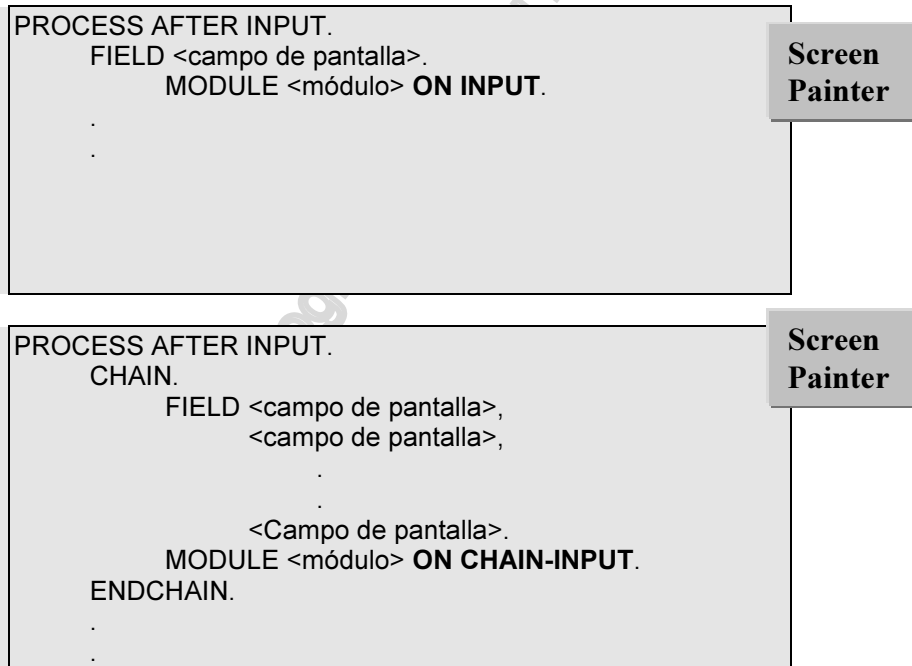
Usando las adiciones **STARTING AT** y **ENDING AT** en la instrucción **CALL SCREEN**, se puede especificar la posición y el tamaño de la pantalla a llamar. En estos casos, los estándares ergonómicos de SAP establecen que la pantalla debe estar definida como de diálogo tipo modal.

Se puede usar la adición **STARTING AT** sin la adición **ENDING AT**. Aquí, el sistema determina el tamaño de la pantalla de diálogo según el tamaño definido en el atributo de pantalla conocido como "Used size". El punto de comienzo de esta pantalla será la esquina superior izquierda. Si también se usa **ENDING AT**, el sistema incluye lo más posible de la pantalla de diálogo dentro del área definida por las coordenadas, iniciando en la esquina superior izquierda.

Si la pantalla aparece incompleta, se incluye en la misma una barra de desplazamiento.



## Ejecución condicionada de módulos



Si se especifica la adición **ON INPUT** después de **MODULE** en una instrucción **FIELD**, el módulo es ejecutado solamente si el campo relevante contiene un valor diferente al valor inicial.

En un estatuto **CHAIN** se debe usar la instrucción **ON CHAIN-INPUT**. Entonces, el módulo concerniente es procesado solamente si al menos uno de los campos de pantalla del estatuto **CHAIN** contiene un valor diferente al valor inicial.



Se puede usar la adición **ON INPUT** solamente si la instrucción **MODULE** es especificada dentro de una instrucción **FIELD**.

Si se especifica la adición **ON REQUEST** después de **MODULE** en una instrucción **FIELD**, el módulo es ejecutado únicamente si el campo relevante tiene una nueva entrada.

En un estatuto **CHAIN**, se debe usar la instrucción **ON CHAIN-REQUEST**. Entonces, el módulo concerniente es procesado solamente si al menos uno de los campos de pantalla del estatuto **CHAIN** tiene una nueva entrada.

Se puede usar la adición **ON REQUEST** solamente si la instrucción **MODULE** es especificada dentro de una instrucción **FIELD**.

```
PROCESS AFTER INPUT.  
  FIELD <campo de pantalla>.  
    MODULE <módulo> ON REQUEST.
```

**Screen  
Painter**

```
PROCESS AFTER INPUT.  
  CHAIN.  
    FIELD <campo de pantalla>,  
      <campo de pantalla>,  
      .  
      <Campo de pantalla>.  
    MODULE <módulo> ON CHAIN-REQUEST.  
  ENDCHAIN.
```

**Screen  
Painter**

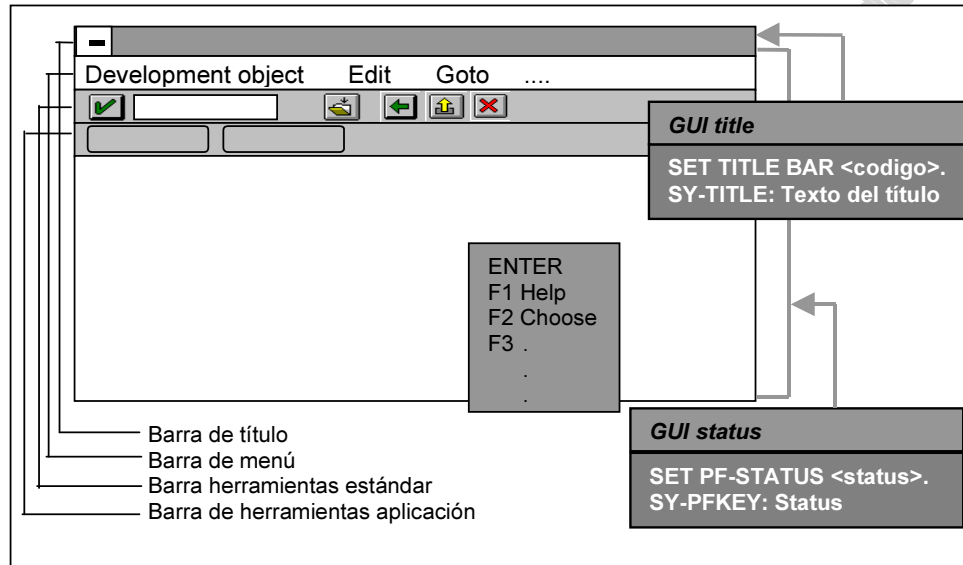
Es posible que en alguna ocasión el usuario quiera salir de la pantalla sin necesidad de pasar las validaciones automáticas (Por ejemplo utilizando las funciones estándares **BACK**, **EXIT** o **CANCEL**). En este caso utilizaremos la cláusula **AT EXIT COMMAND** de la instrucción **MODULE**.

**MODULE <módulo\_ABAP> AT EXIT-COMMAND.**



## Diseño de menús con el Menu Painter

### Introducción



Con el **Menú Painter** diseñaremos las superficies **GUI**, (Graphical User Interface), sobre las que correrán las transacciones SAP.

Una GUI contiene todos los menús, teclas de función, pushbuttons, etc.... disponibles para el usuario, durante la ejecución de una transacción.

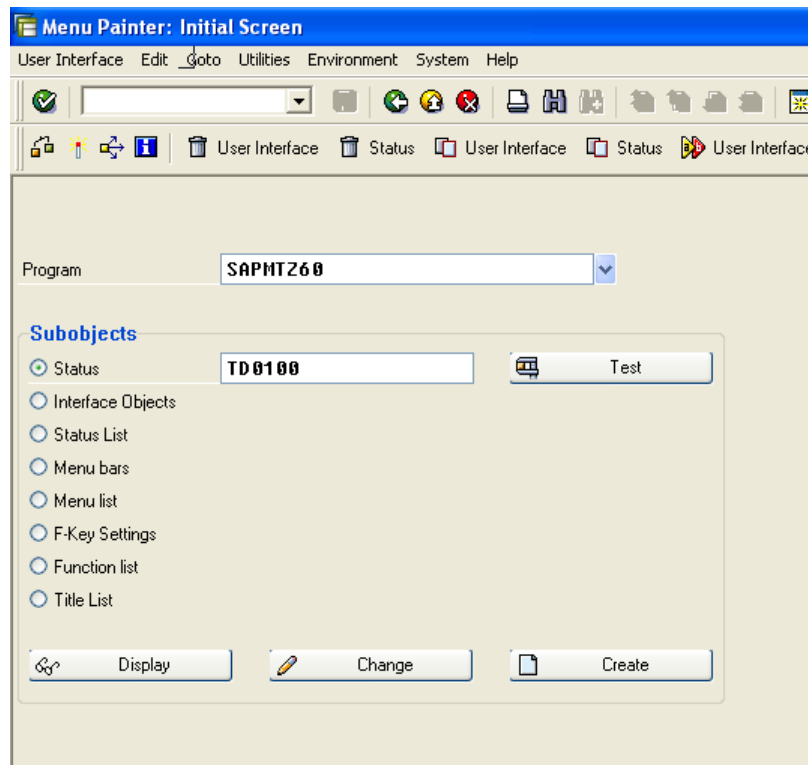
Podremos indicar el status que utilizaremos en una pantalla o el título en un módulo PBO de la pantalla con las instrucciones:

```
SET PF-STATUS <cod_status>.
SET TITLEBAR <cod_título>.
```

Indicaremos las diferentes interfaces GUI de una transacción mediante los **status**. Una transacción tendrá muchos status diferentes. No será necesario redefinir todos los objetos de los status, ya que muchos objetos definidos en un status podrán ser utilizados en otro. Por ejemplo es posible crear una barra de menús igual para toda una transacción.

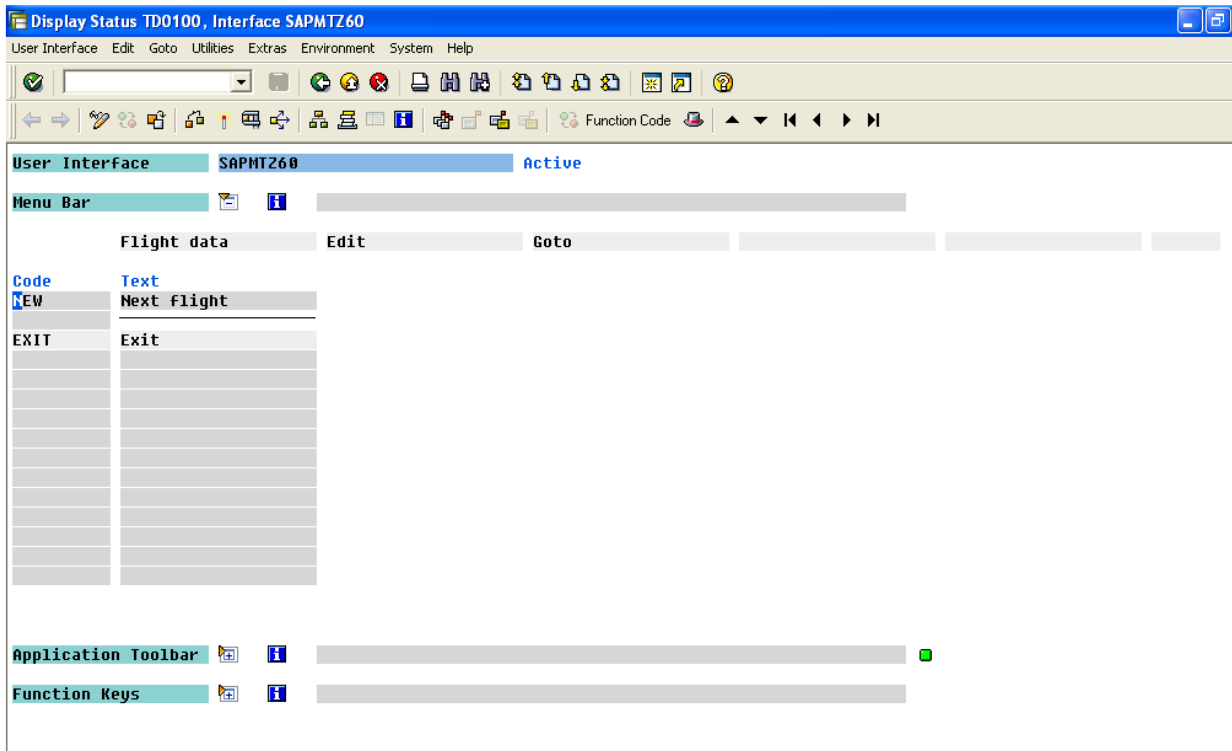
Para iniciar el Menú Painter, puedes usar cualquiera de los siguientes caminos:

- Seleccionando: **Tools -> ABAP/4 Workbench -> Desarrollo -> Menú Painter.**
- Dando el código de transacción **SE41.**
- Dentro del editor de ABAP/4, posicionar el cursor en el nombre del status en la sentencia **SET PFSTATUS <status>** dando doble clic y/o presionar **F2.**



Es posible mantener tanto un estatus de un determinado programa, como los diferentes objetos de un GUI que forman parte de los status (Barras de Menús, teclas de función, títulos de menú...).

## Tecclas de Función:



Como es recomendable que todas las teclas de función que se definan, estén incluidas en la barra de menús y si se desea en la barra de aplicación, comenzaremos por definir estas teclas de función primeramente.

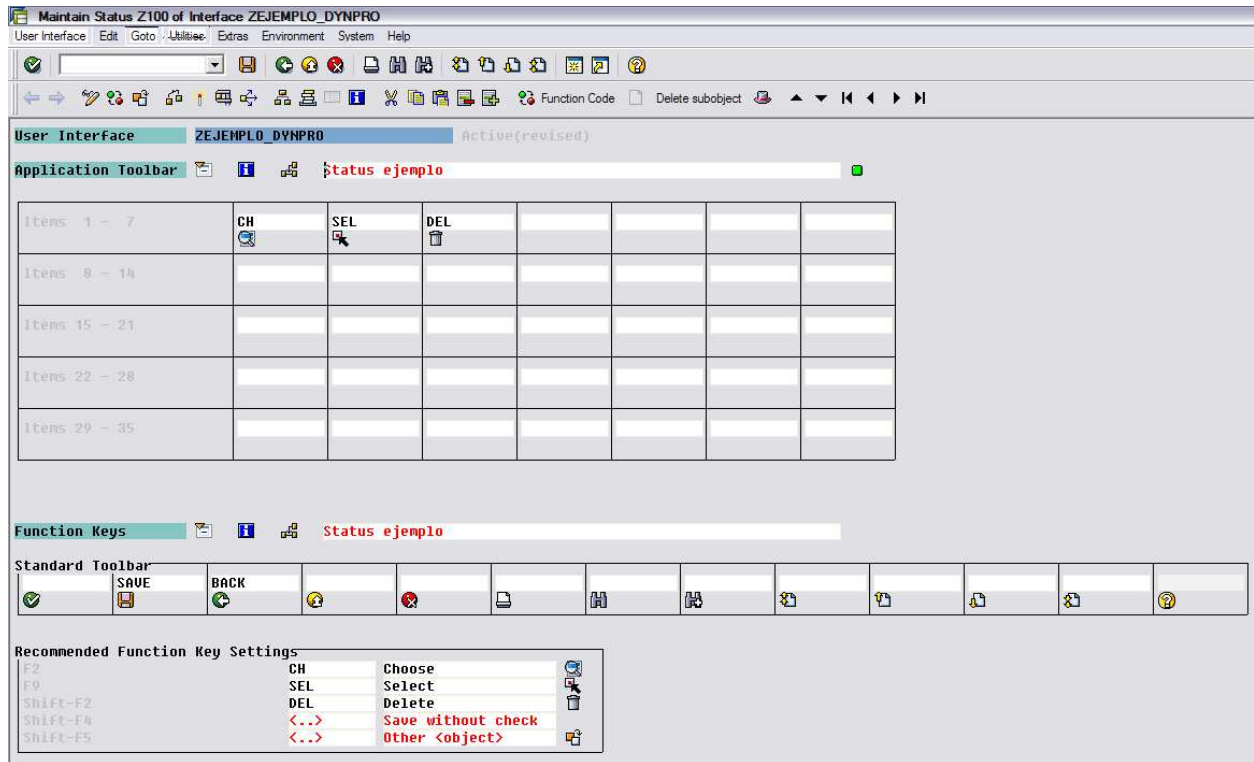
Para definir las teclas de función utilizamos el espacio destinado para ello. Indicando el código de la función en la línea correspondiente a la tecla que deseamos utilizar. El texto de la tecla de función aparecerá automáticamente, pero podrá ser modificada en caso de desearlo.

SAP no recomienda definir nuevas teclas de función en el espacio reservado para teclas de función estándar.

Las teclas de función, son teclas especiales del teclado que permite a los usuarios acceder fácilmente a las opciones indicadas. En SAP existen tres tipos de códigos de función: Reservadas, Recomendadas y de Libre Asignación. Las siguientes teclas están reservadas por SAP y no pueden ser asignadas por el programador:

F1	Ayuda
F3	Volver
F4	Entradas posibles (Match Code)
F12	Cancel

## Los 'Pushbuttons'



Los pushbuttons son botones tridimensionales que aparecen debajo de la barra de herramientas estándar (barra de aplicación).

Previamente a definir un botón será necesario definir la función deseada como una tecla de función.

Para ver que funciones se pueden utilizar nos situaremos sobre 'Application Toolbar' y pulsaremos F4.

Indicaremos el código de función que deseamos que aparezca en la barra de herramientas de aplicación. Podemos especificar si queremos que aparezca un texto corto o únicamente un icono que identifique la función.

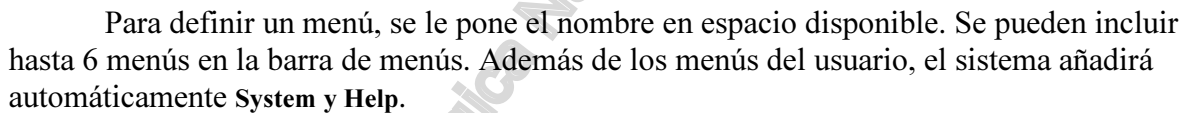
No será necesario definir las funciones de la barra de herramientas estándar, 'Standard Toolbar'.

Para definir iconos para visualizarlos en la barra de herramientas de aplicación será necesario:

Seleccionar: **Edit -> Insert -> Function with icon.**

Entrar el código de función.

Introducir el nombre del icono y el texto del menú.



Para abrir un menú o submenú, hacer un Doble-Click sobre el nombre. Cada entrada estará compuesta de un código de función y un texto de función o un texto de menú. Con **F4** podemos ver una lista de las funciones que podemos utilizar.

En el caso de un menú en cascada, no será necesario indicar el código, y con Doble-Click podemos desarrollar las opciones del submenú.

## Otras utilidades del Menú Painter

### Activación de Funciones

Podemos hacer que las funciones de la barra de menús estén en modo activo o inactivo. En caso de estar inactivas, se visualizarán en la barra de menús en baja intensidad y su selección no implicará efecto alguno, en cambio las funciones activas serán completamente ejecutables.

Para activar o desactivar funciones seleccionar **'Function Activation'**.

### 'FastPaths'

Un 'FastPath' (Camino rápido), es una manera rápida de escoger funciones en un menú, asignando a cada función una tecla.

Podemos mantener 'FastPaths' seleccionando:

Goto -> Further Options -> FastPath.

Indicaremos para las funciones deseadas una letra, normalmente la primera, teniendo cuidado de no especificar la misma letra para distintas funciones.

### Títulos de Menú

Es posible mantener distintos títulos para un menú.

Goto -> Title List.

Cada título se identificará con un código de título de 3 dígitos.

Introduciremos el texto del título, pudiendo utilizar variables de la misma forma que lo hacíamos con los mensajes en ABAP/4, es decir utilizando el símbolo &. Posteriormente será en el programa ABAP/4 donde le indiquemos que título vamos a utilizar con la instrucción:

```
SET TITLEBAR <cod_título> WITH <var1> <var2>...
```

En tiempo de ejecución el título del menú se guardará en la variable del sistema SY-TITLE.



## **Prueba, Chequeo y Generación de Status**

Podemos probar el status simulando la ejecución de la interface con:

User Interface -> Test Status, y introduciendo los datos: Número de pantalla, y Código del título.

Antes de usar la interfase podemos comprobar que la hemos definido correctamente, realizando un proceso de chequeo con: User Interface -> Check Syntax. Posteriormente realizaremos un proceso de generación de la interface que incluye el chequeo y la grabación de la misma.

## **Menús de ámbito o de área**

Un menú de ámbito es una agrupación de transacciones en forma de menú. Es una manera de agrupar las transacciones más frecuentemente utilizadas por un usuario bajo un mismo menú. A diferencia de una transacción de diálogo, el menú de ámbito sólo llama a otras transacciones, no pudiendo incorporar otro tipo de funciones propias.

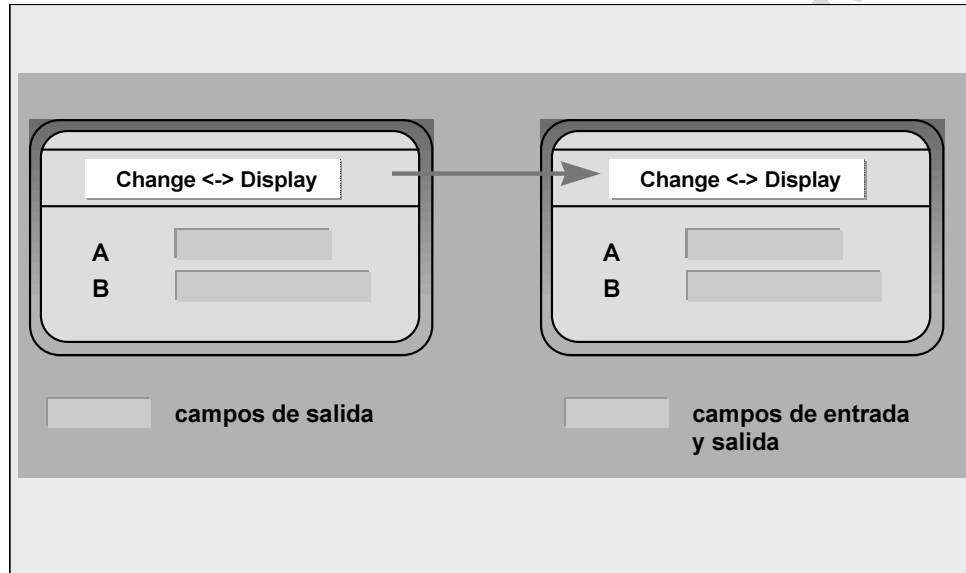
Podemos crear los menús de ámbito con una versión simplificada del Menú Painter.

**Tools ... Development -> Other Tools -> Area Menus.**

Únicamente será necesario introducir los códigos de transacción (tabla TSTC) y el texto del menú.

## Modificación dinámica de pantallas

### Introducción



Se pueden cambiar temporalmente ciertos atributos de campos, por ejemplo cambiar los campos de solo-lectura en campos de entrada/salida.

También se puede usar la modificación dinámica de pantallas para facilitar el ocultar ciertos campos y así evitar secuencias dinámicas de pantallas.

### Atributos de campos modificables

Los campos de pantalla y sus atributos modificables son automáticamente almacenados en la tabla interna SCREEN.

La tabla SCREEN es inicializada con los campos definidos en el Screen Painter y con sus atributos cada vez que el módulo PBO es ejecutado.

Para determinar los campos para los cuales se puede cambiar uno ó más atributos, se lee el campo SCREEN-NAME y del campo SCREEN-GROUP1 al campo SCREEN-GROUP4 en la tabla SCREEN.

El campo SCREEN-REQUEST está reservado para uso interno del sistema.

Tabla SCREEN / Atributos modificables de campos

SCREEN-NAME	<i>Field name</i>
SCREEN-GROUP1	<i>Modification group1</i>
SCREEN-GROUP2	<i>Modification group2</i>
SCREEN-GROUP3	<i>Modification group3</i>
SCREEN-GROUP4	<i>Modification group4</i>
SCREEN-REQUIRED	<i>Required field</i>
SCREEN-INPUT	<i>Input field</i>
SCREEN-OUTPUT	<i>Output field</i>
SCREEN-INTENSIFIED	<i>Highlighted field</i>
SCREEN-INVISIBLE	<i>Invisible field</i>
SCREEN-LENGTH	<i>Field length</i>
SCREEN-ACTIVE	<i>Active field</i>
SCREEN-DISPLAY_3D	<i>3-dimensional field</i>
SCREEN-VALUE_HELP	<i>Field with value help</i>
SCREEN-REQUEST	<i>Input exist (PAI only)</i>

### Atributos: Modificación de grupos

Lista de campos: Modificación de grupos					
Field name	Gr1	Gr2	Gr3	Gr4	...
SPFLI_ITAB-CONNID	SEL				
SPFLI_ITAB-CITYFROM	SEL				
SPFLI_ITAB-CITYTO	SEL				
...					

Screen  
Painter

Se puede asignar un campo a cuatro grupos diferentes. Los nombres de grupos son de tres caracteres de longitud y pueden ser definidos libremente.

## Programa

Para programar la modificación de la pantalla, es necesario un módulo que se ejecuta durante el módulo PROCESS BEFORE OUTPUT.

Se pueden cambiar los atributos de un campo y/o un grupo de campos en la tabla SCREEN. (LOOP AT SCREEN WHERE ... y READ TABLE SCREEN son soportados).

Se activan o desactivan los atributos asignando valores 1 ó 0. Para almacenar los cambios realizados, se utiliza la instrucción MODIFY SCREEN.

No es permitido el usar la instrucción SCREEN-ACTIVE = 1 como una operación dinámica para activar un campo que no se ha definido como oculto en el Screen Painter. Sin embargo, si se puede usar esta instrucción para inactivar un campo definido como visible en el Screen Painter. SCREEN-ACTIVE = 0 tiene el mismo efecto que las tres instrucciones SCREEN-INVISIBLE = 1, SCREEN-INPUT = 0 y SCREEN-OUTPUT = 0.

```
PROCESS BEFORE OUTPUT.
.
.
MODULE MODIFY_SCREEN.
```

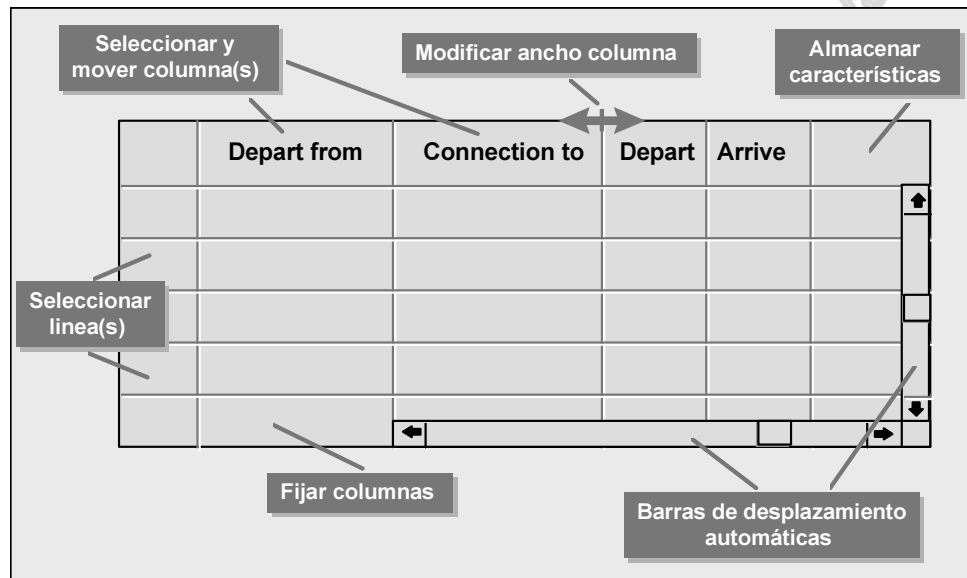
Screen  
Painter

```
MODULE MODIFY_SCREEN OUTPUT.
  LOOP AT SCREEN.
    IF SCREEN-GROUP1 = 'SEL'.
      SCREEN-INPUT = 1.
    ENDIF.
    IF SCREEN-NAME = 'SFLIGHT-CARRID'.
      SCREEN-ACTIVE = 0.
    ENDIF.
    MODIFY SCREEN.
  ENDLOOP.
ENDMODULE.
```

ABAP/4

## Tablas de Control

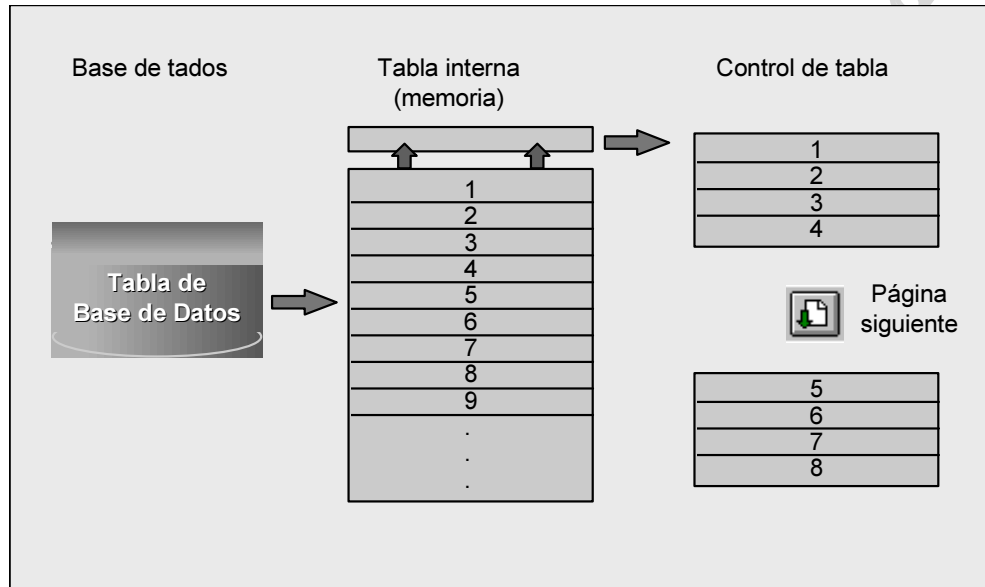
### Características de las tablas de control



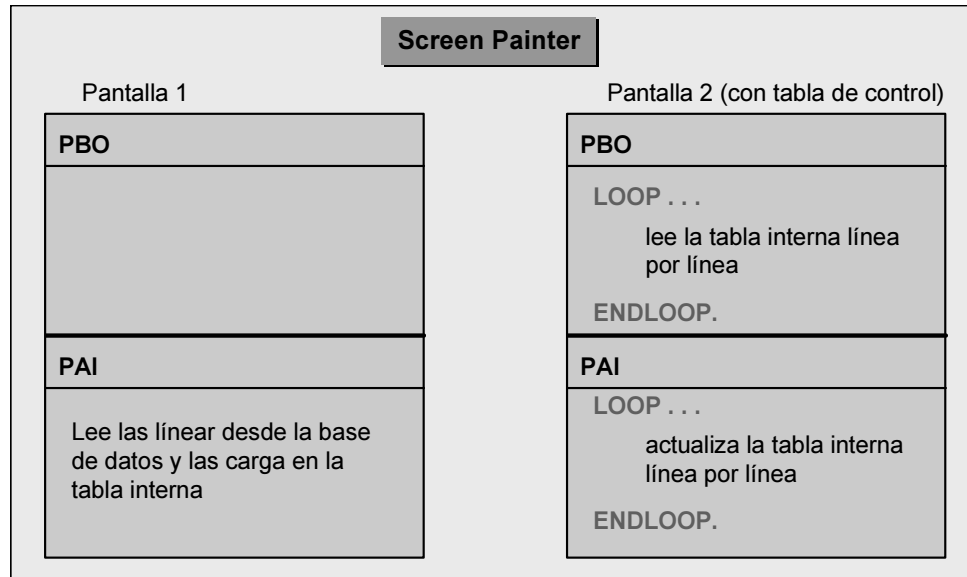
- Con las tablas de control, se pueden mostrar o introducir líneas y/o datos en forma tabular.
- Alcance de Función:
  - Cambiar de tamaño la tabla para desplegar y editar dato.
  - El ancho y posición de columna puede ser modificado por el usuario.
  - La selección de columna y/o línea selección con color-intenso.
  - Selección de línea(s), múltiple, total y de-seleccionar.
  - Encabezados de columna son mostrados como botones para selección de columna.
  - Desplazamiento horizontal y vertical con barras de desplazamiento.
  - Compuesto de algún número de columnas clave (columnas fixed lead).
  - Los atributos de celda son modificables.
- El usuario puede almacenar diferentes características variables y colocar alguna de estas o las características básicas como el actual.

## Principios para las tablas de control

### Llenado

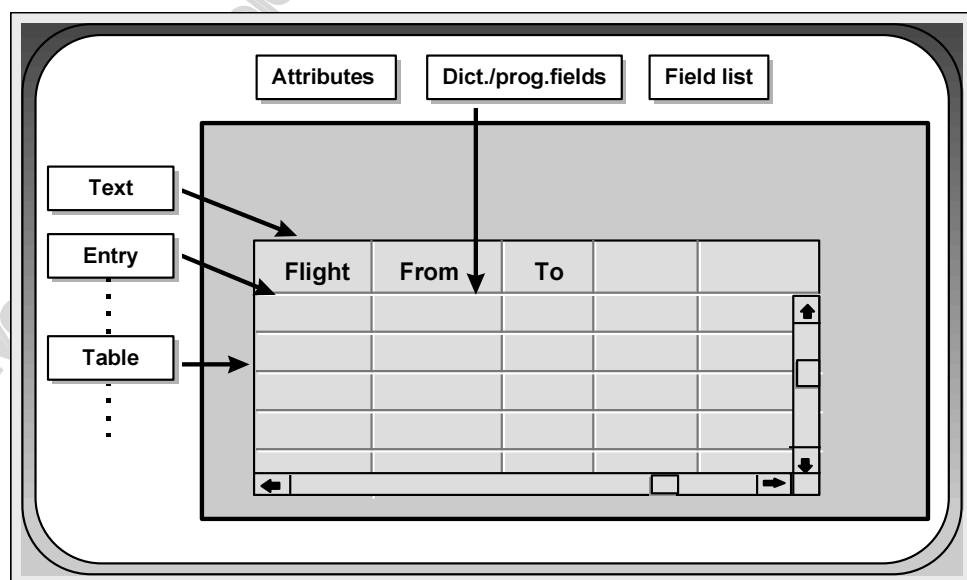


- Por razones de optimización, se leen los datos para la tabla de control una vez y se almacenan en una tabla interna. Las líneas de la tabla de control son entonces tomadas de esta tabla interna.
- Hay solo un área de trabajo para editar líneas dentro de la tabla de control. Por esta razón, se necesita una instrucción LOOP ... ENDLOOP para cada tabla de control en los módulos PBO y PAI del flujo lógico.



- En el módulo PBO, una línea de la tabla de control debe estar llenada con una línea de la tabla interna cada vez que el ciclo es procesado.
- Similarmente, el módulo PAI debe copiar los cambios en una línea de tabla de control dentro de la línea de tabla interna correspondiente.
- Cuando se manejan funciones, se debe diferenciar entre las que se aplican sólo en una línea en una tabla de control y las relacionadas con la pantalla completa.

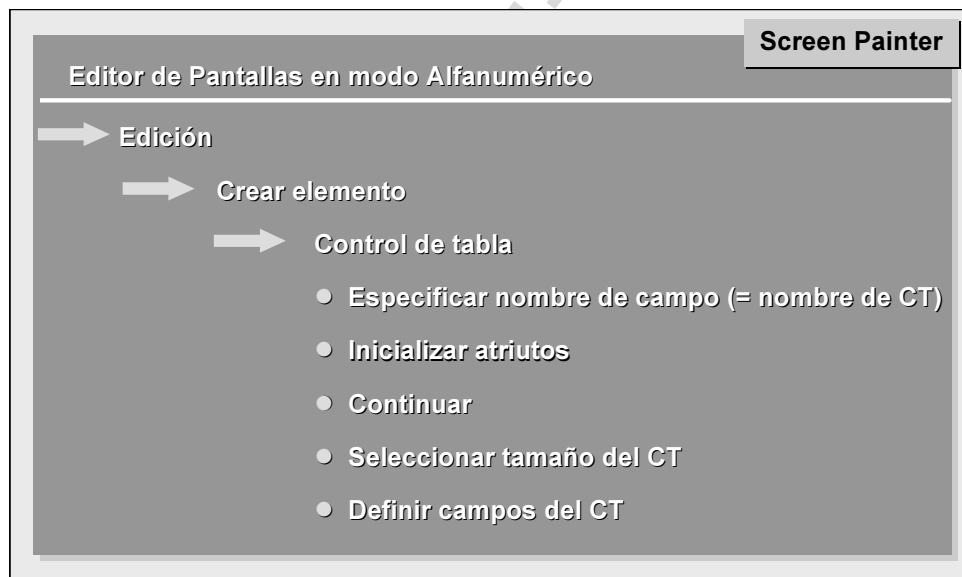
### Creación en modo Gráfico



- En el editor *fullscreen gráfico*, se elige *Table* para crear una tabla de control y se usa el botón izquierdo del ratón para posicionarlo en la pantalla.
- Entonces, se definen los campos en la tabla de control, por ejemplo, usando estos en ABAP/4 Dictionary.

### **Creando Tablas de control (Fullscreen Alfanumérico)**

- Puede definirse una tabla de control en el editor alfanumérico fullscreen.
- En el menú *Edit*, se elige *Create element* y entonces *Table control*.
- Se obtiene un cuadro de diálogo donde se introduce el nombre de la tabla de control y se inicializan los atributos.
- Se va luego al modo de selección y se determina el tamaño de una tabla de control posicionando el control. Entonces, se definen los campos de la tabla de control.
- Puede usarse también campos de ABAP/4 Dictionary o crear nuevos en el programa.





## Definición de una tabla de control en "Module Pool"

```
CONTROLS ctrl TYPE TABLEVIEW USING SCREEN scr.
```

El tipo **TABLEVIEW** corresponde a la estructura **CXTAB\_CONTROL** con los siguientes campos:

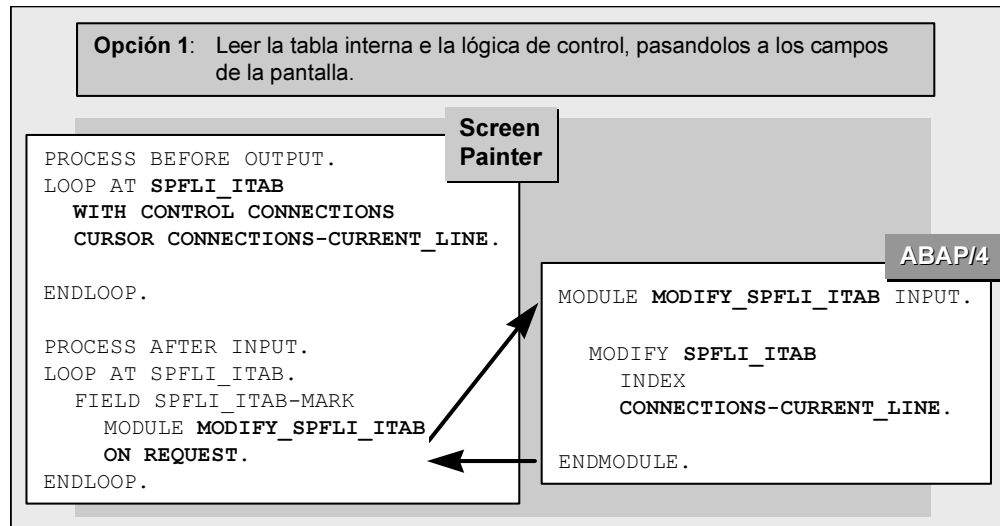
FIXED_COLS	TYPE I	Número de columnas fijas
LINES	TYPE I	Número de líneas para el desplazamiento vertical
TOP_LINE	TYPE I	Primera línea en el siguiente PBO
CURRENT_LINE	TYPE I	Línea actual (en un LOOP ... ENDLOOP)
LEFT_COL	TYPE I	Primera columna desplegada y movable
LINE_SEL_MODE	TYPE I	Selección de línea (0=ninguna, 1=simple, 2=múltiple)
COL_SEL_MODE	TYPE I	Selección de columna (0=ninguna, 1=simple, 2=múltiple)
LINE_SELECTOR		Indicador de línea seleccionada
V_SCROLL		Indicador de barra de desplazamiento vertical
H_GRID		Indicador de línea de grid horizontal
V_GRID		Indicador de línea de grid vertical
COLS	TYPE <b>CXTAB_COLUMN</b>	OCCURS 10

El tipo **CXTAB\_COLUMN** consta de los siguientes campos:

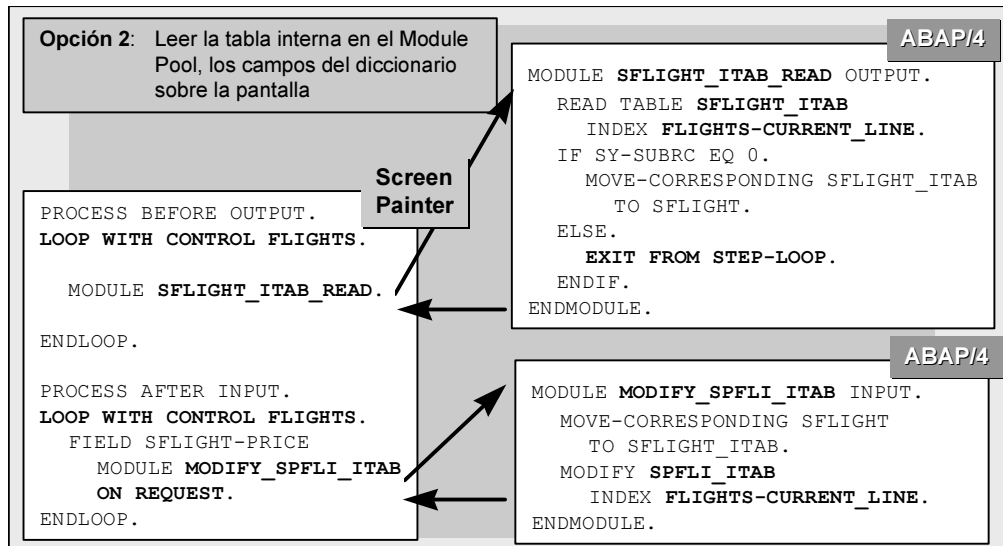
SCREEN	LIKE SCREEN	Atributos de la estructura SCREEN
INDEX	TYPE I	Posición de columna (secuencia de despliegue)
SELECTED		Indicador de columna seleccionada
VISLENGTH	LIKE ICON-OLENG	Ancho visible de columna
INVISIBLE		Indicador de columna invisible

- La instrucción **CONTROLS** define un objeto de dato complejo del tipo **TABLEVIEW** el cual corresponde a el tipo **CXTAB\_CONTROL** definido en el ABAP/4 Dictionary (ver grupo tipo **CXTAB**).
- Se mantienen los valores iniciales en el Screen Painter. Con la adición **USING SCREEN**, se determina la pantalla de la cual se quieren obtener los valores iniciales para la tabla de control.
- Puede usarse la instrucción **REFRESH CONTROL ctrl FROM SCREEN scr** alguna vez para inicializar una pantalla con valores iniciales de la pantalla screen.

## Flujo lógico de la tabla de control

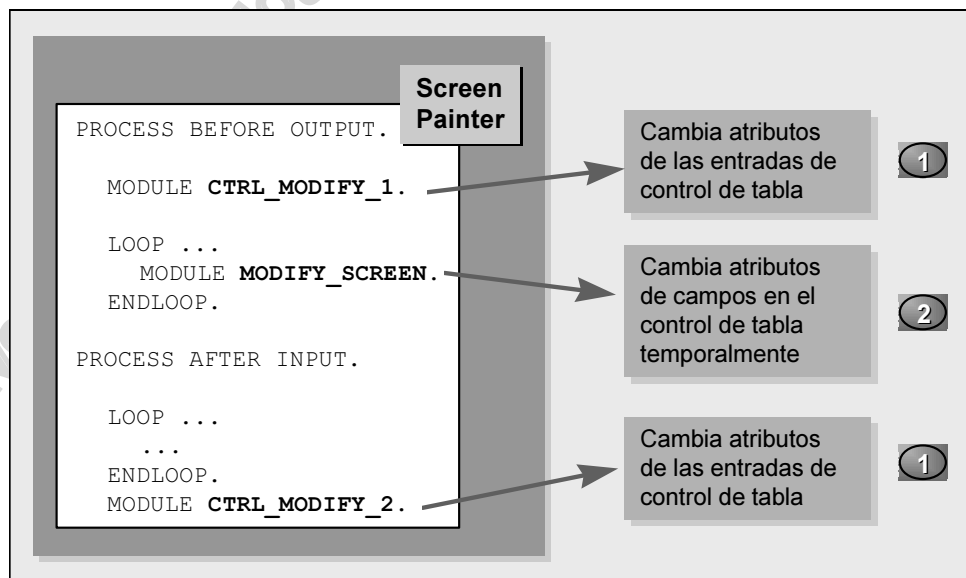


- En el flujo lógico, puede leerse una tabla interna usando la instrucción **LOOP**. Se define la referencia a la tabla de control especificando **WITH CONTROL <ctrl>**.
- Se determina cuál entrada de tabla es leída especificando **CURSOR<ctrl>-CURRENT\_LINE**. El sistema calcula  $\langle ctrl \rangle - CURRENT\_LINE$  de  $\langle ctrl \rangle - TOP\_LINE + SY-STEPL$ , el cual es el ciclo especial indexado por **LOOPS** en el flujo lógico.
- El sistema calcula  $\langle ctrl \rangle - TOP\_LINE$  cuando el usuario desplaza scrolls con la scroll bar, pero no hace lo mismo para desplazar una página. Hay que programarlo.
- Después de leer la operación, el contenido del campo es colocado en la línea de encabezado de la tabla interna. Si los campos en la tabla de control tienen los mismos nombres que los campos de la tabla interna, enseguida son llenados.
- Deben reflejarse algunos cambios que el usuario hace a los campos de una tabla de control en la tabla interna. De otra manera, no aparecerán cuando la pantalla se vuelve a mostrar después de que PBO es ejecutado otra vez, por ejemplo, cuando el usuario presiona ENTER o scrolls.
- Sin embargo, este procesamiento se debe ejecutar solo si los cambios han sido hechos a la línea de la tabla de control.



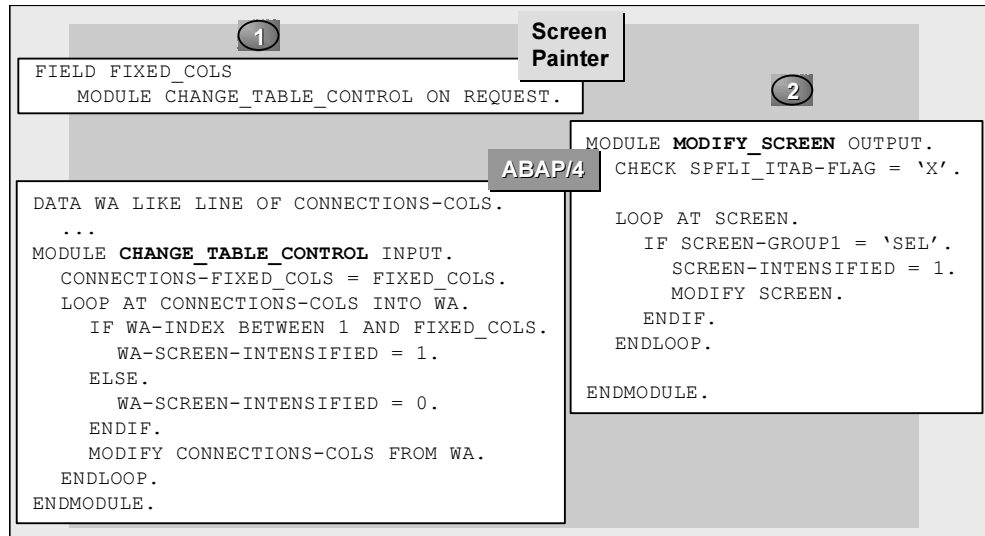
- Si se usa una instrucción LOOP sin una tabla interna en el flujo lógico, se debe leer el dato en un módulo PBO el cual es llamado cada vez que el ciclo es procesado.
- Ya que el sistema no puede determinar el número de entradas de tabla interna, se debe usar la instrucción EXIT FROM STEP-LOOP para asegurar que no hay líneas vacías blank desplegadas en la tabla de control si no hay más entradas de tablas internas correspondientes.
- También se debe determinar el número de líneas para desplazamiento vertical como punto conveniente: <ctrl>-LINES = número de entradas.

## Modificación



- Pueden cambiarse los atributos de la tabla de control por los contenidos de campos.

- Para modificar los atributos de celdas individuales temporalmente, se cambia la tabla SCREEN en un módulo PBO que es procesado entre LOOP y ENDLOOP en el flujo lógico. (Use LOOP AT SCREEN, MODIFY SCREEN).



- El ejemplo 1 de arriba muestra cómo se puede reaccionar al requerir el componente de una columna. Si el usuario requiere el componente de una columna, la tabla de control es por consiguiente cambiada (CONNECTIONS-FIXED\_COLS = FIXED\_COLS). Los componentes de columnas son remarcados en color. Este es almacenado cambiando la tabla de control.
- El ejemplo 2 de arriba muestra una posible reacción de una línea seleccionada. Durante el PAI, el campo ayuda itab-FLAG es llenado con 'X'. En el PBO, el procesamiento reacciona a esto remarcando el color todos los campos en una línea donde el itab-FLAG es colocado a 'X' (todos ellos tienen el grupo atributo 'SEL' ). Esto es almacenado cambiando la tabla SCREEN.

## Control de páginas

- Para implementar desplazamiento de página, calcular lo necesario usando el atributo de tabla de control <ctrl>-TOP\_LINE.
- Para esto (en el PAI), se necesita conocer el número actual de líneas en la tabla de control.
- En el PBO, el campo de sistema SY-LOOPC contiene el número actual de líneas de tabla de control. En el PAI, contiene el número de líneas actualmente llenas.
- SY-LOOPC solo contiene un valor entre LOOP y ENDLOOP porque cuenta el número de ciclos.

PROCESS BEFORE OUTPUT.

```
LOOP ... .
  MODULE GET_LOOPLINES.
ENDLOOP.
```

PROCESS AFTER INPUT.

```
LOOP ... .
...
ENDLOOP.

MODULE USER_COMMAND_0200.
```

DATA: LOOPLINES LIKE SY-LOOPC

```
...
MODULE GET_LOOPLINES OUTPUT.
  LOOPLINES = SY-LOOPC.
ENDMODULE.
```

MODULE USER\_COMMAND\_0200 INPUT.

```
...
WHEN 'F21'.
  FLIGHTS-TOP_LINE = 1.
WHEN 'F22'.
  FLIGHTS-TOP_LINES =
    FLIGHTS-TOP_LINE - LOOPLINES.
  IF FLIGHTS-TOP_LINES < 1.
    FLIGHTS-TOP_LINES = 1.
...
WHEN 'F24'.
  FLIGHTS-TOP_LINE =
    FLIGHTS-TOP_LINES
    - LOOPLINES + 1.
```

ENDMODULE.

## Posición del cursor

### Sintaxis

#### GET CURSOR

```
FIELD f
VALUE v
LINE l
OFFSET o.
```

#### SET CURSOR

```
FIELD f
VALUE v
LINE l
OFFSET o.
```

### Determinar posición del cursor:

¿Cuál entrada de la tabla interna  
corresponde a la línea de la tabla de control?

#### ABAP/4

```
DATA: SELLINE LIKE SY-STEPL,
      TABIX LIKE
      CONNECTIONS-TOP_LINE.
...
GET CURSOR LINE SELLINE.
TABIX = CONNECTIONS-TOP_LINE
      + SELLINE - 1.
READ TABLE SPFLI_ITAB
INDEX TABIX.
```

- El parámetro LINE en la instrucción GET o SET se refiere a el campo de sistema SY-STEPL, el cual es el ciclo de flujo lógico especial indexado.

- Para determinar la entrada de tabla interna que corresponde a la línea de tabla de control seleccionada, se calcula como sigue:

Línea requerida = <ctrl>-TOP\_LINE + línea de posición del cursor determinada -1.

- La instrucción GET CURSOR es apoyada por el código de regreso encontrado en el campo de sistema SY-SUBRC. Si el valor es 0, el cursor es posicionado en un campo. Si el valor es 4, el cursor no está en un campo.

A continuación, en la próxima Unidad, veremos todos los detalles y funcionalidades del ALV (Abap List Viewer), herramienta muy útil para la confección avanzada de reportes. En tanto, los invitamos a realizar las actividades de la Unidad actual.

---