# ABAP List Viewer

# INTRODUCTION

- The common features of report are column alignment, sorting, filtering, subtotals, totals etc. To implement these, a lot of coding and logic is to be put. To avoid that we can use a concept called ABAP List Viewer (ALV).

- Using ALV, we can have three types of reports:
  1. Simple Report
  2. Block Report
  3. Hierarchical Sequential Report

There are some function modules which will enable to produce the above reports without much effort.

All the definitions of internal tables, structures and constants are declared in a **type-pool** called **SLIS**.

# SIMPLE REPORT

1. Simple Report

The important function modules are:

- Reuse_alv_list_display
- Reuse_alv_fieldcatalog_merge
- Reuse_alv_events_get
- Reuse_alv_commentary_write
- Reuse_alv_grid_display

# SIMPLE REPORT CONTD….

A. **REUSE_ALV_LIST_DISPLAY:** This is the function module which prints the data.

The important parameters are:

**1. Export:**

a. I_callback_program : report id
b. I_callback_pf_status_set : routine where a user can set his own pf status or change the functionality of the existing pf status.
c. I_callback_user_command : routine where the function codes are handled.

# SIMPLE REPORT CONTD...

d. I_structure name : name of the dictionary table

e. Is_Layout : structure to set the layout of the report

f. It_fieldcat : internal table with the list of all fields and their attributes which are to be printed (this table can be populated automatically by the function module REUSE_ALV_FIELDCATALOG_MERGE)

g. It_events : internal table with a list of all possible events of ALV and their corresponding routine names.

**2. Tables:**

a. t_outtab : internal table with the data to be output

# SIMPLE REPORT CONTD...

B. **REUSE_ALV_FIELDCATALOG_MERGE:**

This function module is used to populate a fieldcatalog which is essential to display the data in ALV. If the output data is from a single dictionary table and all the columns are selected, then we need not exclusively create the field catalog. Its enough to mention the table name as a parameter(I_structure_name) in the REUSE_ALV_LIST_DISPLAY. But in other cases we need to create it.

# SIMPLE REPORT CONTD...

The important parameters are:

**1. Export:**

a. I_program_name : report id

b. I_internal_tabname : the internal output table

c. I_inclname : include or the report name where all the dynamic forms are handled.

**2. Changing**

ct_fieldcat : an internal table with the type SLIS_T_FIELDCAT_ALV which is declared in the type pool SLIS.

# SIMPLE REPORT CONTD...

C . **REUSE_ALV_EVENTS_GET**: Returns table of possible events for a a list type

1. **Import:**

Et_Events : The event table is returned with all possible CALLBACK events for the specified list type (column ‘NAME’). For events to be processed by the Callback, their ‘FORM’ field must be filled. If the field is initialized, the event is ignored. The entry can be read from the event table, the field ‘FORM’ filled and the entry modified using constants from the type pool SLIS.

# SIMPLE REPORT CONTD...

**2. Export:**

I_list_type:

0 = simple list
1 = hierarchical-sequential list
2 = simple block list
3 = hierarchical-sequential block list

# SIMPLE REPORT CONTD...

D. **REUSE_ALV_COMMENTARY_WRITE** : This is used in the Top-of-page event to print the headings and other comments for the list.

1. **It_list_commentary** : Internal table with the headings of the type **slis_t_listheader.**

This internal table has three fields:

Typ : ‘H’ - header, ‘S’ - selection, ‘A’ - action

Key : only when typ is ‘S’.

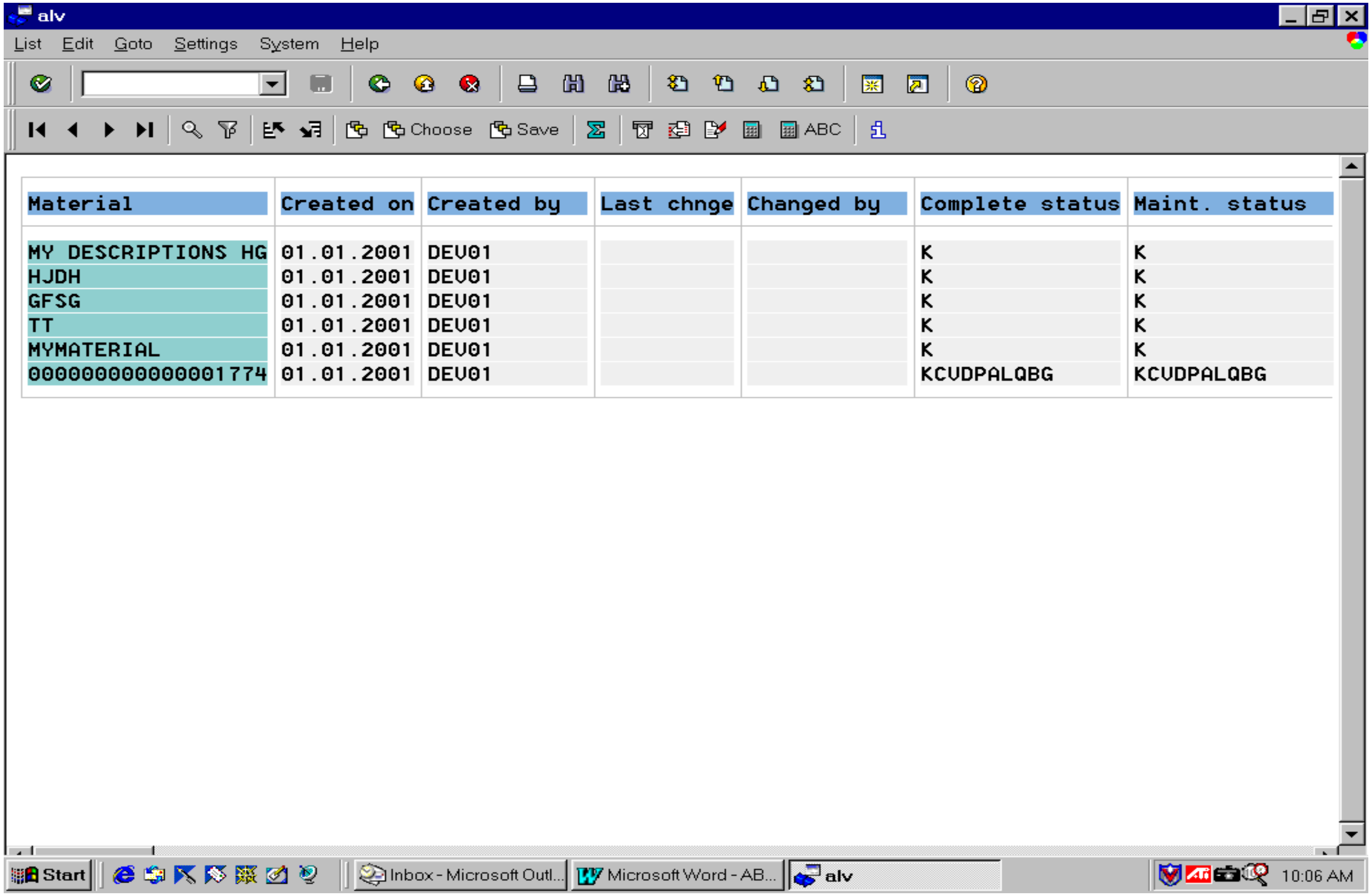
Info : the text to be printed

# SIMPLE REPORT CONTD...

E. **REUSE_ALV_GRID_DISPLAY**: A new function in 4.6 version, to display the results in grid rather than as a list.

Parameters : same as reuse_alv_list_display

# The example of a simple list is as follows:



| Material | Created on | Created by | Last chnge | Changed by | Complete status | Maint. status |
|---|---|---|---|---|---|---|
| MY DESCRIPTIONS HG | 01.01.2001 | DEV01 | | | K | K |
| HJDH | 01.01.2001 | DEV01 | | | K | K |
| GFSG | 01.01.2001 | DEV01 | | | K | K |
| TT | 01.01.2001 | DEV01 | | | K | K |
| MYMATERIAL | 01.01.2001 | DEV01 | | | K | K |
| 000000000000001774 | 01.01.2001 | DEV01 | | | KCVDPALQBG | KCVDPALQBG |

# HIERARCHICAL REPORTS

Hierarchical sequential list output.

The function module is

**A. REUSE_ALV_HIERSEQ_LIST_DISPLAY**

1. Export:
   a. I_CALLBACK_PROGRAM
   b. I_CALLBACK_PF_STATUS_SET
   c. I_CALLBACK_USER_COMMAND
   d. IS_LAYOUT

# Hierarchical Reports Contd.….

e. IT_FIELDCAT

f. IT_EVENTS

g. I_TABNAME_HEADER : Name of the internal table in the program containing the output data of the highest hierarchy level.

h. I_TABNAME_ITEM : Name of the internal table in the program containing the output data of the lowest hierarchy level.

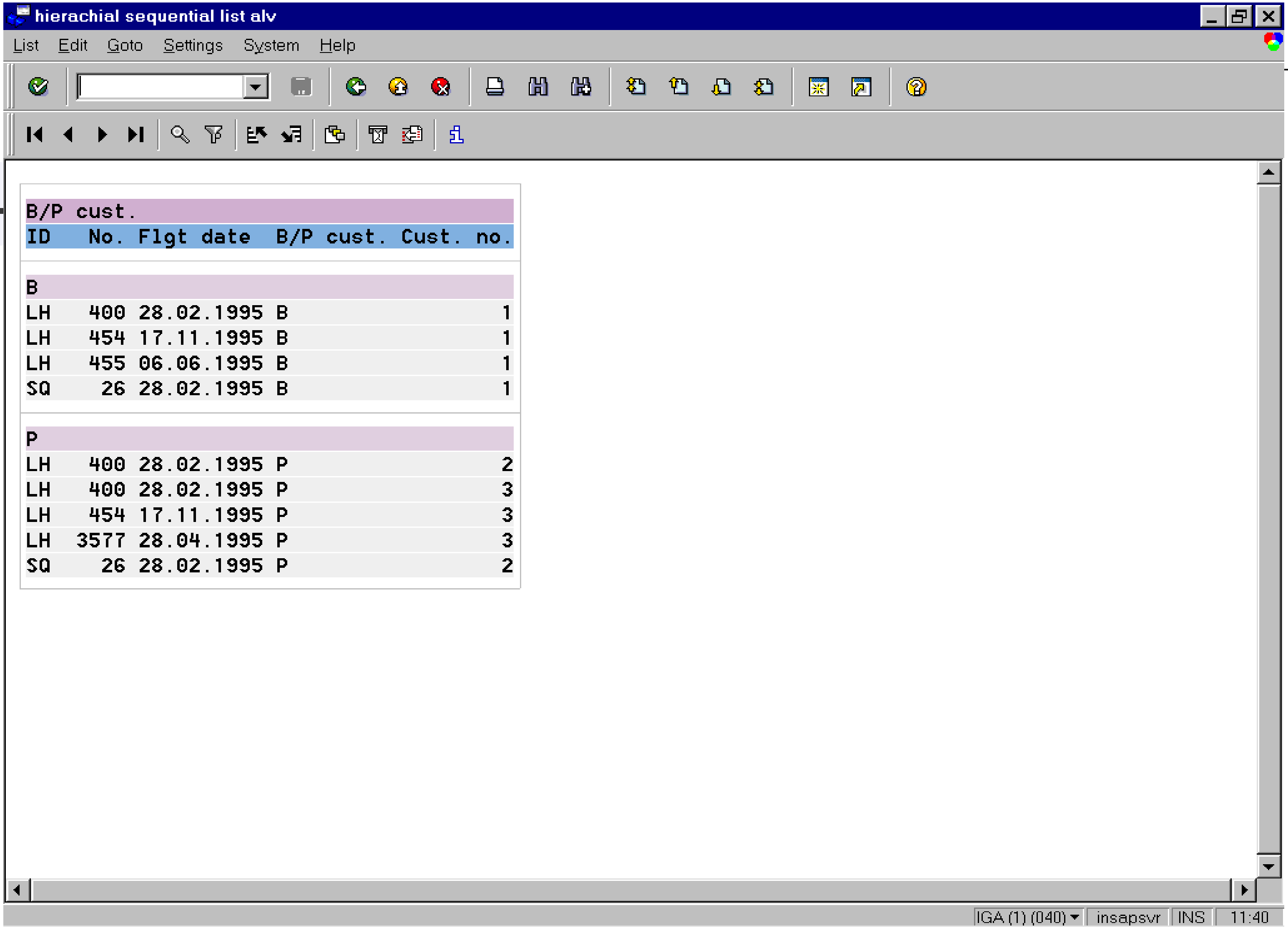
i. IS_KEYINFO : This structure contains the header and item table field names which link the two tables (shared key).

# Hierarchical reports Contd.….

**2. Tables**

a. T_OUTTAB_HEADER : Header table with data to be output

b. T_OUTTAB_ITEM : Name of the internal table in the program containing the output data of the lowest hierarchy level.



```
B/P cust.
ID    No. Flgt date  B/P cust. Cust. no.

B
LH    400 28.02.1995 B                 1
LH    454 17.11.1995 B                 1
LH    455 06.06.1995 B                 1
SQ     26 28.02.1995 B                 1

P
LH    400 28.02.1995 P                 2
LH    400 28.02.1995 P                 3
LH    454 17.11.1995 P                 3
LH   3577 28.04.1995 P                 3
SQ     26 28.02.1995 P                 2
```

# **BLOCK REPORT**

This is used to display multiple lists continuously.

The important functions used in this report are:

A. REUSE_ALV_BLOCK_LIST_INIT

B. REUSE_ALV_BLOCK_LIST_APPEND

D.  REUSE_ALV_BLOCK_HS_LIST_APPEND

C. REUSE_ALV_BLOCK_LIST_DISPLAY

# BLOCK REPORT CONTD..

A. **REUSE_ALV_BLOCK_LIST_INIT**

Parameters:

a. I_CALLBACK_PROGRAM
b. I_CALLBACK_PF_STATUS_SET
c. I_CALLBACK_USER_COMMAND

This function module is used to set the default GUI status etc.

# BLOCK REPORT CONTD..

B**. REUSE_ALV_BLOCK_LIST_APPEND**

Export :

a. IS_LAYOUT : layout settings for block
b. IT_FIELDCAT : field catalog
c. I_TABNAME : Internal table name of the output data
d. IT_EVENTS : internal table name with all possible events

Tables :

a. T_OUTTAB : internal table with output data.

This function module adds the data to the block.

# REUSE_ALV_LIST_HS_APPEND : -

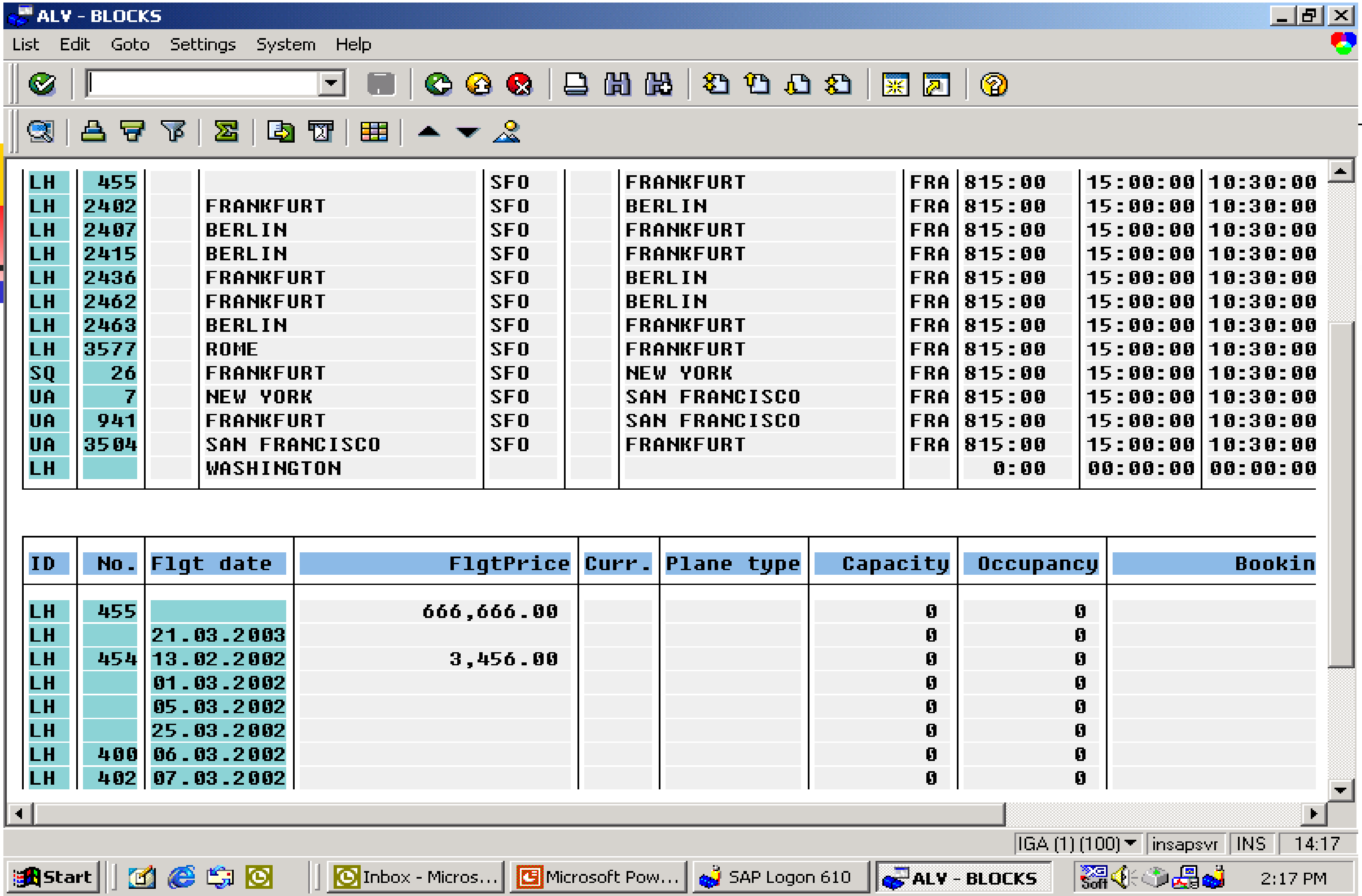
Is used to append the Hierarchical Sequential blocks.

# BLOCK REPORT CONTD..

**C. REUSE_ALV_BLOCK_LIST_DISPLAY**

Parameters : All the parameters are optional.

This function module display the list with data appended by the above function.



| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| LH | 455 | | | SFO | | FRANKFURT | FRA | 815:00 | 15:00:00 | 10:30:00 |
| LH | 2402 | | FRANKFURT | SFO | | BERLIN | FRA | 815:00 | 15:00:00 | 10:30:00 |
| LH | 2407 | | BERLIN | SFO | | FRANKFURT | FRA | 815:00 | 15:00:00 | 10:30:00 |
| LH | 2415 | | BERLIN | SFO | | FRANKFURT | FRA | 815:00 | 15:00:00 | 10:30:00 |
| LH | 2436 | | FRANKFURT | SFO | | BERLIN | FRA | 815:00 | 15:00:00 | 10:30:00 |
| LH | 2462 | | FRANKFURT | SFO | | BERLIN | FRA | 815:00 | 15:00:00 | 10:30:00 |
| LH | 2463 | | BERLIN | SFO | | FRANKFURT | FRA | 815:00 | 15:00:00 | 10:30:00 |
| LH | 3577 | | ROME | SFO | | FRANKFURT | FRA | 815:00 | 15:00:00 | 10:30:00 |
| SQ | 26 | | FRANKFURT | SFO | | NEW YORK | FRA | 815:00 | 15:00:00 | 10:30:00 |
| UA | 7 | | NEW YORK | SFO | | SAN FRANCISCO | FRA | 815:00 | 15:00:00 | 10:30:00 |
| UA | 941 | | FRANKFURT | SFO | | SAN FRANCISCO | FRA | 815:00 | 15:00:00 | 10:30:00 |
| UA | 3504 | | SAN FRANCISCO | SFO | | FRANKFURT | FRA | 815:00 | 15:00:00 | 10:30:00 |
| LH | | | WASHINGTON | | | | | 0:00 | 00:00:00 | 00:00:00 |

| ID | No. | Flgt date | FlgtPrice | Curr. | Plane type | Capacity | Occupancy | Bookin |
|---|---|---|---|---|---|---|---|---|
| LH | 455 | | 666,666.00 | | | 0 | 0 | |
| LH | | 21.03.2003 | | | | 0 | 0 | |
| LH | 454 | 13.02.2002 | 3,456.00 | | | 0 | 0 | |
| LH | | 01.03.2002 | | | | 0 | 0 | |
| LH | | 05.03.2002 | | | | 0 | 0 | |
| LH | | 25.03.2002 | | | | 0 | 0 | |
| LH | 400 | 06.03.2002 | | | | 0 | 0 | |
| LH | 402 | 07.03.2002 | | | | 0 | 0 | |

**Here the functions REUSE_ALV_FIELDCATALOG_MERGE, REUSE_ALV_EVENTS_GET, REUSE_ALV_COMMENTARY_WRITE can be used.**

# INTERNAL TABLES IN SLIS

**Slis_t_fieldcat_alv** : This internal table contains the field attributes. This internal table can be populated automatically by using ‘REUSE_ALV_FIELDCATALOG_MERGE’.

Important Attributes:

| | | |
|---|---|---|
| a. col_pos | : | position of the column |
| b. fieldname | : | internal fieldname |
| c. tabname | : | internal table name |
| d. ref_fieldname | : | fieldname (dictionary) |
| e. ref_tabname | : | table (dictionary) |
| f. key(1) | : | column with key-color |
| g. icon(1) | : | icon |
| h. hotspot(1) | : | hotspot |

| | | |
|---|---|---|
| i. Symbol(1) | : | symbol |
| j. Checkbox(1) | : | checkbox |
| k. just(1) | : | (R)ight (L)eft (C)ent |
| l. do_sum(1) | : | sum up |
| m. no_out(1) | : | (O)blig. (X)no out |
| n. outputlen | : | output length |
| o. seltext-l | : | long key word |
| p. seltext_m | : | middle key word |
| q. seltext_s | : | short key word |
| r. reptext_ddic | : | heading(ddic) |
| s. ddictxt(1) | : | (S)hort (M)iddle (L)ong |
| t. datatype | : | datatype |

**2. SLIS_T_EVENT :** Internal table for storing all the possible events of the ALV. This can be populated by the function module **Reuse_alv_events_get**

The columns are :

| | | |
|---|---|---|
| •name | : | name of the event |
| •form | : | name of the routine |

# SYNTAXES FOR THE ROUTINES

- **I_CALLBACK_PF_STATUS_SET**

Syntax :

```
FORM set_pf_status USING rt_extab TYPE slis_t_extab
```

The table RT_EXTAB contains the function codes which are hidden in the standard interface.

- **I_CALLBACK_USER_COMMAND**

Syntax :

```
 FORM user_command  USING r_ucomm LIKE sy-ucomm
                          rs_selfield TYPE slis_selfield.
```

The parameter r_ucomm contains the function code.

The structure rs_selfield has the details about the current cursor position.