ALV Grid Control (BC-SRV-ALV)

HELP.BCSRVALV

Release 4.6C

Copyright

© Copyright 2001 SAP AG. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

Some software products marketed by SAP AG and its distributors contain proprietary software components of other software vendors.

Microsoft®, WINDOWS®, NT®, EXCEL®, Word®, PowerPoint® and SQL Server® are registered trademarks of
Microsoft Corporation.

IBM®, DB2®, OS/2®, DB2/6000®, Parallel Sysplex®, MVS/ESA®, RS/6000®, AIX®, S/390®, AS/400®, OS/390®, and OS/400® are registered trademarks of IBM Corporation.

ORACLE® is a registered trademark of ORACLE Corporation.

INFORMIX®-OnLine for SAP and Informix® Dynamic Server™ are registered trademarks of Informix Software Incorporated.

UNIX®, X/Open®, OSF/1®, and Motif® are registered trademarks of the Open Group.

HTML, DHTML, XML, XHTML are trademarks or registered trademarks of W3C®, World Wide Web Consortium,
Massachusetts Institute of Technology.

JAVA® is a registered trademark of Sun Microsystems, Inc.

JAVASCRIPT® is a registered trademark of Sun Microsystems, Inc., used under license for technology invented and implemented by Netscape.

SAP, SAP Logo, R/2, RIVA, R/3, ABAP, SAP ArchiveLink, SAP Business Workflow, WebFlow, SAP EarlyWatch, BAPI, SAPPHIRE, Management Cockpit, mySAP.com Logo and mySAP.com are trademarks or registered trademarks of SAP AG in Germany and in several other countries all over the world. All other products mentioned are trademarks or registered trademarks of their respective companies.

Icons

Icon	Meaning
⚠	Caution
	Example
	Note
	Recommendation
	Syntax

Contents

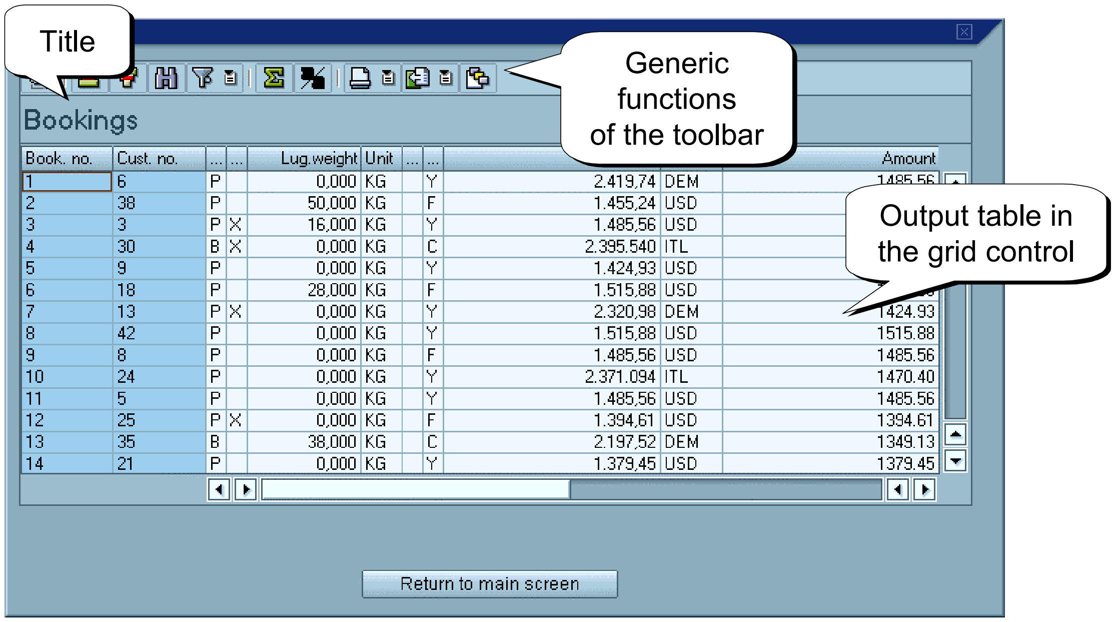
ALV Grid Control (BC-SRV-ALV)

Purpose

The ALV Grid Control (ALV = ***S****AP* **L***ist* **V***iewer*) is a flexible tool for displaying lists. The tool provides common list operations as generic functions and can be enhanced by self-defined options. This allows you to use the ALV Grid Control in a large range of application programs.

In SAP development projects, the ALV Grid Control is also used as a tool for changing and creating tables. However, this functionality is currently only used in pilot projects and has not yet been released for customers.

The following graphic shows a list displayed with the ALV Grid Control in a dialog box:


Title
Generic functions of the toolbar
Output table in the grid control
Bookings
Return to main screen

Book. no.	Cust. no.	...	...	Lug.weight	Unit	...	...			Amount
1	6	P		0,000	KG		Y	2.419,74	DEM	1485.56
2	38	P		50,000	KG		F	1.455,24	USD	[illegible]
3	3	P	X	16,000	KG		Y	1.485,56	USD	[illegible]
4	30	B	X	0,000	KG		C	2.395.540	ITL	[illegible]
5	9	P		0,000	KG		Y	1.424,93	USD	[illegible]
6	18	P		28,000	KG		F	1.515,88	USD	[illegible]
7	13	P	X	0,000	KG		Y	2.320,98	DEM	1424.93
8	42	P		0,000	KG		Y	1.515,88	USD	1515.88
9	8	P		0,000	KG		F	1.485,56	USD	1485.56
10	24	P		0,000	KG		Y	2.371.094	ITL	1470.40
11	5	P		0,000	KG		Y	1.485,56	USD	1485.56
12	25	P	X	0,000	KG		F	1.394,61	USD	1394.61
13	35	B		38,000	KG		C	2.197,52	DEM	1349.13
14	21	P		0,000	KG		Y	1.379,45	USD	1379.45

As the user sees it, the ALV Grid Control consists of a toolbar, a title and the output table displayed in a grid control. If required, the user can hide the title and the standard functions of the toolbar.

Implementation Considerations

The ALV Grid Control uses controls technology to achieve state-of-the-art on-screen display. Like all control wrappers, the ALV Grid Control offers methods through a global class in the system which can be used to affect its behavior.

SAP does not guarantee that the methods, events and attributes of this class that are **not** public will remain unchanged or will be available in future releases. This is why you should not derive the class to access protected objects of the class. Using these objects makes it more difficult to upgrade to subsequent releases.

As a result of using ABAP Objects, lists are displayed through an ALV instance, and programmers make use of the event management of ABAP Objects.

Integration

Controls are software components that are installed on the local PC. In order to communicate with these components, all control wrappers use methods of the *Control Framework*. In this context, the ALV Grid Control is a special case as it uses a control that is already wrapped: the grid control. For the programmer, this can eliminate certain steps in the process of event management for controls.

The ALV Grid Control uses the SAP context menu to integrate standard functions. This menu can be customized to meet individual requirements.

This documentation focuses on using the ALV Grid Control in the *SAP GUI for the Windows™ Enviroment* or the *SAP GUI for the Java™ Enviroment*. It is also possible to display the ALV Grid Control in the Web [Page 43].

Features

This is an overview of the functions provided by the ALV Grid Control. You can:

- Display non-hierarchical lists consistently with a modern design
- Use typical list functions - such as sorting and filtering - without extra programming effort
- Adapt predefined list functions and their enhancements
- Program responses to user actions (such as double-clicking a line) individually
- Connect to the report/report interface.

For practical examples of the ALV Grid Control, see development class **SLIS**.

Constraints

The ALV Grid Control does not allow you to display block or hierarchical lists. Currently, simple lists can be displayed in single-line format only. However, users can define multiple-line format in the print preview and print the list as such.

Instance for the ALV Grid Control

Definition

This instance is defined with reference to class **`cl_gui_alv_grid`**:

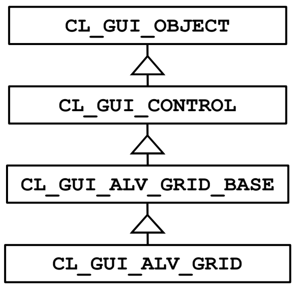
```
data <name of reference variable> type ref to cl_gui_alv_grid.
```

Use

An instance for the ALV Grid Control manages all information pertaining to a list on your screen. You can call methods on this instance with which you can define and change the properties of this control.

Do not derive this class to be able to access protected areas of the class. This is not required for using the ALV Grid Control.

Inheritance Hierarchy



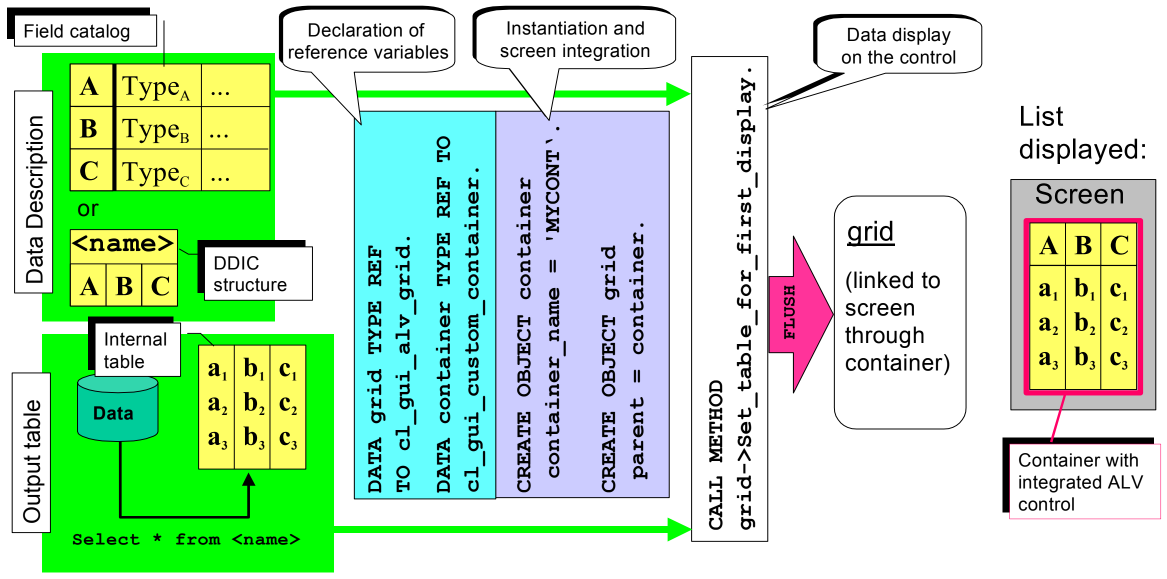
Integration

Class **`cl_gui_alv_grid`** contains both control-specific methods [Page 66] and methods of the OO Control Framework [Page 167].

Working With the ALV Grid Control

Basics

The following graphic illustrates the steps required to display a list with the ALV Grid Control:


Field catalog
Data Description
A Type_A ...
B Type_B ...
C Type_C ...
or
<name>
A B C
DDIC structure
Output table
Internal table
Data
a1 b1 c1
a2 b2 c2
a3 b3 c3
Select * from <name>
Declaration of reference variables
Instantiation and screen integration
DATA grid TYPE REF TO cl_gui_alv_grid.
DATA container TYPE REF TO cl_gui_custom_container.
CREATE OBJECT container container_name = 'MYCONT'.
CREATE OBJECT grid parent = container.
CALL METHOD grid->set_table_for_first_display.
Data display on the control
FLUSH
grid (linked to screen through container)
List displayed:
Screen
Container with integrated ALV control

As a minimum, you must provide the following two types of information for displaying the data:

- An internal table with the data to be displayed, called the *output table*
- A description of the structure of this data that is declared to the ALV Grid Control through the *field catalog* or through the corresponding structure of the Data Dictionary.

Generally, the output table contains data that you previously selected from database tables.

The reference to the output table that you pass to the ALV Grid Control should be valid as long as the ALV Grid Control operates on it. Besides defining this reference as a global table, you can also hold the reference in an ABAP Objects instance using a public attribute.

The field catalog [Page 129] is a table that contains information on the fields to be displayed. For example, the ALV uses this table to identify the type of a field. You can also use special fields of this catalog to determine the number format and column properties of the list to be output.

Working With Controls

As a result of using ABAP Objects for Release 4.6A, SAP made controls programming consistent for the following procedures:

- Creating a control and integrating it into the screen
- Passing methods from the backend to the frontend

- Handling events triggered by the control at the frontend
- Destroying the control (Lifetime Management)

For Basis controls (such as the textedit control, the HTML viewer control or the picture control), the same programming model is applicable to the above procedures.

This programming model is also valid for the ALV Grid Control, with some restrictions, however, in the area of event handling (see next section). To make yourself familiar with this general model, see Creating a Control [Ext.] (and all cross-references). Also, you are absolutely recommended to read the next section on the special aspects of the ALV Grid Control.

Special Event Handling Aspects of the ALV Grid Control

The ALV Grid Control uses the grid control to display the output table. So we can say that the ALV Grid Control is a wrapper that uses the wrapper of a Basis control. As the 'outer layer', this wrapper spares the developer from having to register the events on the frontend in order to simplify event handling. The ALV Grid Control differs from Basis controls in the following respects:

- All events are registered as system events when the control is instantiated.

  If you want to register all events as application events, you must use parameter **`I_APPL_EVENTS`** (**See also:** CONSTRUCTOR [Page 69]). As usual, you must then call method **`CL_GUI_CFW=>DISPATCH`** in the PAI module.

- Events **`DELAYED_CALLBACK`** or **`DELAYED_CHANGED_SEL_CALLBACK`** are registered using method register_delayed_event [Page 86].
- For Drag & Drop With the ALV Grid Control [Page 24] no **`DISPATCH`** call is required.

First Steps

This section describes the easiest way to display a list with selected data in the ALV Grid Control. To do this, you must:

1. Create an instance of the ALV Grid Control and integrate it into a screen.
2. Select the data to be displayed and pass it together with a description of the fields to the instance.

See also sample report BCALV_GRID_DEMO in development class **SLIS**.

Creating an ALV Grid Control

You instantiate an ALV Grid Control in the same way as other controls:

1. Declare reference variables for the ALV Grid Control and the container. In addition, declare an internal table that you fill with selected data later on:

```
DATA: grid  TYPE REF TO cl_gui_alv_grid,
      g_custom_container TYPE REF TO cl_gui_custom_container
      gt_sflight TYPE TABLE OF sflight.
```

In order to integrate a control into a screen, you can use four different container controls [Ext.] (in this example, we use the *custom container control*).

2. Create a standard screen and mark an area for the custom container control in the graphical Screen Painter (icon identified by letter 'C'). Assign name CCCONTAINER to this area.

Exercise 1: Reserving an Area for a Control [Ext.] of the Controls Tutorial explains how to mark an area in the alphanumerical Screen Painter version.

3. In the PBO module of the screen, you must now instantiate the container control and the ALV Grid Control. By doing this, you create a link between the container control and the screen, using the container created in the Screen Painter. Using parameter **parent**, you define the container control as the parent of the ALV Grid Control:

```
IF g_custom_container IS INITIAL.

    CREATE OBJECT g_custom_container
       EXPORTING
          CONTAINER_NAME = 'CCCONTAINER'.

    CREATE OBJECT GRID1
       EXPORTING
         I_PARENT = g_custom_container.

ENDIF.
```

The **IF** query of reference variable **g_custom_container** ensures that the instances are only generated when the PBO is processed for the first time.

Normally, you must use method **cl_gui_cfw=>flush** to pass the methods called to the frontend. However, since the Control Framework performs an automatic flush at the end of the PBO module, this step is not required here.

When you start the program, although the two instances (the container control and the ALV Grid Control) are generated, they are not visible.

Displaying a List in the ALV Grid Control

Once you have created the ALV Grid Control and integrated it into a screen using a container control, you must pass the data and its structure to the ALV Grid Control:

1. Fill the internal table with data:

```
SELECT * FROM sflight INTO TABLE gt_sflight.
```

2. Pass the output table and the structure data to the ALV Grid Control. Again, ensure to call this method only once after the ALV Grid Control is created:

```
CALL METHOD grid->set_table_for_first_display
             EXPORTING I_STRUCTURE_NAME = 'SFLIGHT'
             CHANGING  IT_OUTTAB        = gt_sflight.
```

In this case, the structure data is provided through the Data Dictionary. The ALV Grid Control gets the field information from table SFLIGHT and displays all fields of the table.

Application-Specific Programming

The following features of the ALV Grid Control make it a multi-purpose tool suitable for many applications:

- *Layouts* allow users to adapt list output to their requirements. Application developers determine which saving options users have available for the layout of a list.
- *Self-defined functions* created by application developers can be added to the toolbar. In addition, developers can adapt the standard functions to their individual application or even hide them if they are not needed.
- *Drag & Drop functions* are programmable. The ALV Grid Control allows developers to use the Drag & Drop control for implementing application-specific functions between the ALV Grid Control and other custom controls.
- *Formatting, output and control options* are provided through three different structures:
  - The Field Catalog [Page 129]
  - The Layout Structure [Page 157]
  - The Print Structure [Page 165]

The most important scenarios are described in this section. For more information, see the description of methods, events, and structures.

Layouts

With the help of layouts, users can customize the list output to meet their requirements.

Application developers control user authorizations for layouts using parameters `I_SAVE`, `IS_VARIANT` and `I_DEFAULT` of method set_table_for_first_display [Page 100]. Three modes are available:

Mode	**Result**	**Values of `I_SAVE` and `IS_VARIANT` when `set_table_for_first_display` is called**

Changing the current layout only	The ALV Grid Control displays only the layout icon (▦) but no menu in the toolbar. By clicking this icon, users can change the current layout (basically, they can modify the selection and the order of the columns displayed).	`IS_VARIANT = SPACE.`
		
		`I_SAVE = SPACE.`
		
		(default setting)

Loading delivered layouts only	The ALV Grid Control displays both the layout icon and a menu in the toolbar. The menu contains only functions for selecting and changing layouts.	`IS_VARIANT` = <layout structure> `I_SAVE` = `SPACE`. (or `IS_VARIANT` is `SPACE` and `I_SAVE` is not `SPACE`)
Loading and saving layouts	The ALV Grid Control displays both the layout icon and a menu in the toolbar. By clicking this icon, users can select an existing layout. The menu provides functions for selecting, changing, saving and managing layouts.	`IS_VARIANT` = <layout structure> `I_SAVE` = <'X', 'U' or 'A'> See also: Saving Layouts [Page 17]

By setting parameter `I_DEFAULT`, you can allow users to save default layout settings (set in the default setting).

If users have the required authorization, they can transport layouts in layout management by choosing *Layout ->Transport...*. from the menu.

Saving Layouts

Purpose

The standard function `Change Layout` can be accessed by users by clicking an icon in the toolbar. This icon allows users to adjust the current layout to their requirements for the lifetime of the control instance.

The application developer determines if users are allowed to save layouts and defines the saving options available.

See also report **BCALV_GRID_09** in development class **SLIS**.

Prerequisites

In the default setting, users can save a default layout. If such a default layout exists, it is loaded when method set_table_for_first_display [Page 100] is called. If you do not want to allow users to create default layouts, set parameter **I_DEFAULT** of this method to **SPACE**.

Process Flow

1. Declare a variable for determining the saving options available to the user, and a structure for identifying a layout:

```
DATA: X_SAVE,                          "for Parameter I_SAVE
      GS_variant TYPE DISVARIANT. "for parameter IS_VARIANT
```

2. The structure of type **DISVARIANT** must contain the report ID as a minimum:

```
G_REPID = SY-REPID.
GS_variant-REPORT = G_REPID.
```

3. Determine the saving options available to the user:

```
X_SAVE = 'U'.    "layouts can only be saved as user specific ones
```

4. Pass both variables using call **set_table_for_first_display**:

```
CALL METHOD GRID1->SET_TABLE_FOR_FIRST_DISPLAY
         EXPORTING I_STRUCTURE_NAME = 'SFLIGHT'
                   IS_VARIANT       = GS_variant
                   I_SAVE           = X_SAVE
         CHANGING  IT_OUTTAB        = GT_SFLIGHT.
```

If you pass only the actual parameter for **IS_VARIANT**, users can only load layouts but not save new ones. If you use parameter **I_SAVE**, passing a layout structure with **IS_VARIANT** is a required step.

Result

The toolbar now includes a pull-down menu next to the layout icon that allows users to save and load layouts. Depending on the value of parameter **`I_SAVE`**, the following options for saving layouts are available:

Saving Options for Layouts

`I_SAVE = SPACE`	Layouts cannot be saved.
`I_SAVE = 'U'`	Only user-defined layouts can be saved.
`I_SAVE = 'X'`	Only global layouts can be saved.
`I_SAVE = 'A'`	Both user-defined and global layouts can be saved.

Integration of Self-Defined Functions

The ALV Grid Control provides standard functions for editing lists. Users can access these functions from the toolbar or the context menu. To integrate your own functions, you can:

- Replace existing functions with self-defined functions
- Add new functions to the toolbar or the context menu

You can also hide or disable functions that are not needed in a specific context.

Changing Standard Functions

Purpose

The standard functions are not designed for specific applications. As a result, special knowledge of the data displayed cannot be considered. In individual cases, you may design a standard function - such as sorting by a specific column - more effectively in a specific application.

Process Flow

1. Define an event handler method for event before_user_command [Page 108]. This event is triggered after the user has selected a function. This means that the ALV Grid Control passes control to the application **before** the function is executed. Using event parameter `I_UCOMM` you can restrict the function code to the function you want to modify.
2. Implement your own algorithm for the function within the event handler method. In this context, you can call methods of the ALV Grid Control.
3. Reset the function code to ensure that the standard function is no longer executed:

```
CALL METHOD <Instance of the ALV Ccntrol>->set_user_command
                                     exporting I_UCOMM = SPACE.
```

If you never need specific functions of the tool bar, you can hide them for the entire life-cycle of the control. To do this, you use parameter `IT_TOOLBAR_EXCLUDING` of method set_table_for_first_display [Page 100] to pass a table that contains the function codes to be hidden.

Defining GUI Elements in the Toolbar

Process Flow

1. Define an event handler method for event **`TOOLBAR`**.
2. Declare a structure for defining a toolbar element:

```
data: ls_toolbar  TYPE stb_button.
```

3. For a pushbutton, for example, you would fill the following fields:

```
CLEAR ls_toolbar.
MOVE 0 TO ls_toolbar-butn_type.
MOVE 'BOOKINGS' TO ls_toolbar-function.
MOVE icon_employee TO ls_toolbar-icon.
MOVE 'Show Bookings'(111) TO ls_toolbar-quickinfo.
MOVE SPACE TO ls_toolbar-disabled.
```

In the **`butn_type`** field, you specify the type of the GUI element for the ALV Grid Control. For possible values, see the value range of domain **`TB_BTYPE`**.

4. Use event parameter **`E_OBJECT`** to append the new definition to table **`mt_toolbar`**:

```
APPEND ls_toolbar TO e_object->mt_toolbar.
```

5. If you want to define additional elements, go back to step 3.
6. Call method set_toolbar_interactive [Page 103], if you want to rebuild the toolbar.

Result

In the event handler method of event **`USER_COMMAND`**, you can query the function code you specified in field **`function`** to implement the associated function.

See also report **`BCALV_GRID_05`** in development class **`SLIS`**.

Defining a Context Menu

Use

The ALV Grid Control uses the context menu [Ext.] and assigns standard functions to it. Depending on the context, you can add self-defined functions to this menu and hide or disable preset functions.

Integration

The context menu is an instance of class **CL_CTMENU**. If event **CONTEXT_MENU_REQUEST** is triggered by the user, event parameter **E_OBJECT** contains a reference to the standard context menu.

See also report **BCALV_GRID_06** in development class **SLIS**.

Features

You can change this menu as follows:

- To add an option to the menu, call method **add_function**:

```
CALL METHOD E_OBJECT->ADD_FUNCTION
            EXPORTING FCODE   = 'DELE'
            TEXT     = TEXT-003. "Delete
```

- To disable existing functions (and gray them out), you pass all corresponding function codes of method **disable_functions** in a table:

-
  ```
  DATA: LT_FCODES   TYPE UI_FUNCTIONS,
  CLEAR LT_FCODES.
  APPEND CL_GUI_ALV_GRID=>MC_FC_COL_OPTIMIZE TO LT_FCODES.
  APPEND CL_GUI_ALV_GRID=>MC_FC_HELP TO LT_FCODES.
  CALL METHOD E_OBJECT->DISABLE_FUNCTIONS
        EXPORTING FCODES    = LT_FCODES.
  ```

- To hide existing functions, you follow the same procedure and call method **hide_functions** instead of method **disable_functions**.

With methods **enable_functions** and **show_functions** you can enable or display the relevant functions again.

You read the function codes for self-defined functions at event user_command [Page 128].

Defining a Menu in the Toolbar

Purpose

Just as a context menu, a menu of the toolbar is an instance of class **CL_CTMENU**. You can freely define several menus and add them to the toolbar.

Process Flow

1. Define a GUI element of type 1 (menu with a default button) or type 2 (menu without a default button) in the toolbar. For a description of the procedure, see Defining GUI Elements in the Toolbar [Page 21].

   See also: Special Aspects for Menus with a Default Button [Ext.]

2. In the event handler method of event **MENU_BUTTON**, query the function code you assigned for the menu in step 1 (using event parameter **E_UCOMM**). This way, you can distinguish between the different toolbar menus.
3. For each function code, define a menu as described in Defining a Context Menu [Page 22].

For each menu function, assign additional function codes in step 3 that you can query in event user_command [Page 128].

See also report **BCALV_GRID_07** in development class **SLIS**.

Drag & Drop With the ALV Grid Control

The ALV Grid Control allows you to use the Drag & Drop control. For this control, you must first define a *Drag & Drop behavior* and then use a handle to set this behavior for elements of the controls concerned. The Drag & Drop control then uses the handles to decide which operations the user may perform and modifies the mouse pointer when the mouse is clicked.

To associate actions with the Drag & Drop behavior set, the ALV Grid Control provides events **`OnDrag`**, **`OnDrop`**, **`OnDropComplete`** and **`OnDropGetFlavor`**.

This section describes how to link the handles for the Drag & Drop behavior with elements of the ALV Grid Control. The Control Framework [Ext.] documentation covers Drag & Drop [Ext.] programming (Process Flow of a Drag & Drop Operation [Ext.], Drag & Drop Events [Ext.]).

Examples

Development class **`SLIS`** contains demo reports for Drag & Drop operations with the ALV Grid Control:

- **`BCALV_DND_01`** (dragging rows of the grid control to nodes of the tree control)
- **`BCALV_DND_02`** (dragging function icons of the tree control to rows of the grid control)
- **`BCALV_DND_03`** (defining Drag & Drop behavior for cells of the grid control)
- **`BCALV_DND_04`** (moving or copying rows within an ALV Grid Control)

D&D Behavior for all Rows/Columns

Purpose

You can use this process if you want to define a standard Drag & Drop behavior for all rows or columns. For example, if attribute **`dragsource`** is set in the behavior, you can link all rows or columns to objects that have set attribute **`droptarget`** and use the same flavor.

It does not make any difference if you define a behavior for all rows or for all columns.

Although you can use this method to define an identical behavior for all rows, you can determine in events **`OnDrop`** and **`OnDropGetFlavor`** if you want to cancel the Drag & Drop operation. To do this, you use the data object passed and the flavor as a basis. In the data object, you normally store the data of the row dragged.

Process Flow

1. Define the layout structure of type **`LVC_S_LAYO`**.
2. Define your Drag & Drop behavior and get the associated handle using method **`get_handle`** of class **`cl_dragdrop`**.
3. Assign the handle to field **`s_dragdrop-row_ddid`** or **`s_dragdrop-col_ddid`** of the layout structure.
4. Pass the layout structure with method set_table_for_first_display [Page 100].

You can also subsequently pass the layout structure with the Drag & Drop behavior using method set_frontend_layout [Page 92].

Result

Each row or column can be dragged by the user or appears as the target for any Drag & Drop operation.

If you want to check the Drag & Drop behavior defined on the screen, you must have at least one drag source and one drop target that have the same flavor.

D&D Behavior for Special Columns

Purpose

You can use this process to define a Drag & Drop behavior for special columns of the grid control. Each column can have its own Drag & Drop behavior.

Process Flow

1. Define an internal table of type **`LVC_T_FCAT`** for the field catalog.
2. Define one or several Drag & Drop behaviors and get their handles using method **`get_handle`** of class **`cl_dragdrop`**.
3. Create the field catalog manually [Page 133] or semi-automatically [Page 135]. Assign the corresponding handle to field **`dragdropid`** for selected columns (you can use field **`fieldname`** to refer to the desired column).
4. Pass the field catalog with method set_table_for_first_display [Page 100].

   You can also subsequently pass the field catalog with the Drag & Drop behavior using method set_frontend_fieldcatalog [Page 91].

Result

The columns have a Drag & Drop behavior according to the handle passed.

If you want to check the Drag & Drop behavior defined on the screen, you must have at least one drag source and one drop target that have the same flavor.

D&D Behavior for Special Rows/Cells

Purpose

You can use this process to define a Drag & Drop behavior for special rows or cells of the grid control. Each row or cell can have its own Drag & Drop behavior.

Process Flow

1. Define the layout structure of type **LVC_S_LAYO**.
2. Add a Drag & Drop table of type **LVC_T_DRDR** to your output table, as shown in the example below:

```
DATA: BEGIN OF GT_OUTTAB OCCURS 0.
        INCLUDE STRUCTURE <DDIC-Struktur>.

DATA:   CT TYPE LVC_T_DRDR.  "Table for d&d cell behaviour

DATA: END OF GT_OUTTAB.
```

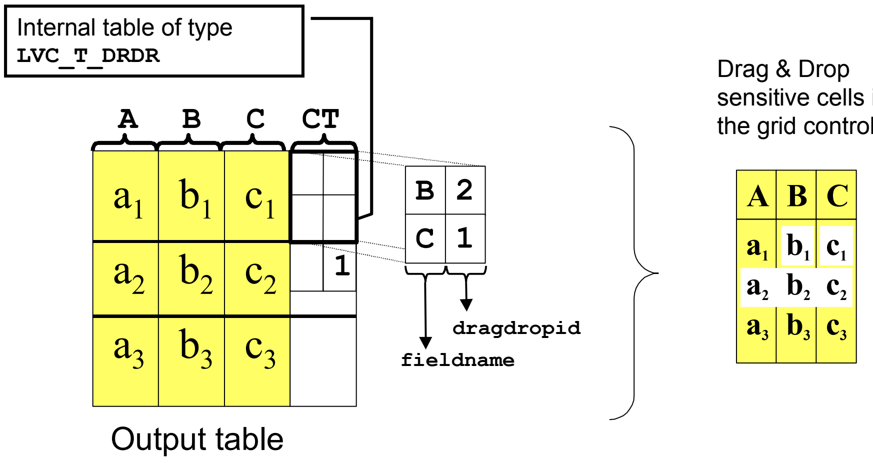
3. Define your Drag & Drop behavior and get its handle with method **get_handle** of class **cl_dragdrop**.
4. Select your data and copy it in the output table.
5. Read one row of the output table at a time in a loop. A row of the Drag & Drop table has two fields. Assign values to these fields as follows (see also the graphic in the next section):
   – If the entire row should have a Drag & Drop behavior, assign the corresponding handle to field **dragdropid**. Field **fieldname** does not receive a value in this case.
   – If only specific columns of the row should have a Drag & Drop behavior, you must append one row for each column to the Drag & Drop table. Assign the name of the desired column to field **fieldname**, and the corresponding handle to field **dragdropid**.
6. Assign the name of the internal table to field **s_dragdrop-fieldname** of the layout structure (in our case **'CT'**, see step 2).
7. Pass the layout structure and the output table with method set_table_for_first_display [Page 100].

You can also subsequently pass the layout structure with the Drag & Drop behavior using method set_frontend_layout [Page 92]. The ALV Grid Control refreshes this table when method refresh_table_display [Page 85] is called. The instance of the ALV Grid Control and the associated output table must have the same lifetime.

Example

The following graphic shows an output table where cells b_1 and c_1 have different Drag & Drop behaviors and the entire second row has the same Drag & Drop behavior as cell c_1:

D&D Behavior for Special Rows/Cells



Output of Exceptions

Purpose

Exceptions are a graphical means to indicate that a threshold value in a list line has been exceeded. For example, as far as the flight model is concerned, you would use the red traffic light to indicate that a flight is full. The different colors help the user to quickly understand what the data displayed means. You can use three different statuses:

Display	Internal value	Indicates (for example)
⚫⚫🟢	3	Low occupancy
⚫🟡⚫	2	Medium to high occupancy (critical: flight almost full)
🔴⚫⚫	1	High occupancy (no seats available)

You can also display an exception as an LED (see Exceptions [Page 161]).

Application developers must both determine the threshold values and adapt the value of an exception to the threshold.

See also: Sample report **BCALV_GRID_04** in development class **SLIS**.

Process Flow

1. Define the layout structure of type **LVC_S_LAYO**:

```
DATA gs_layout TYPE LVC_S_LAYO.
```

2. Add a variable of type C to your output table, as shown in the following example:

```
DATA: BEGIN OF GT_OUTTAB OCCURS 0.
        INCLUDE STRUCTURE <DDIC-Struktur>.
DATA:   light TYPE C.   "to display exceptions
DATA: END OF GT_OUTTAB.
```

3. Set field **EXCP_FNAME** of the layout structure to the field name of the exception:

```
gs_layout-excp_fname = 'LIGHT'.
```

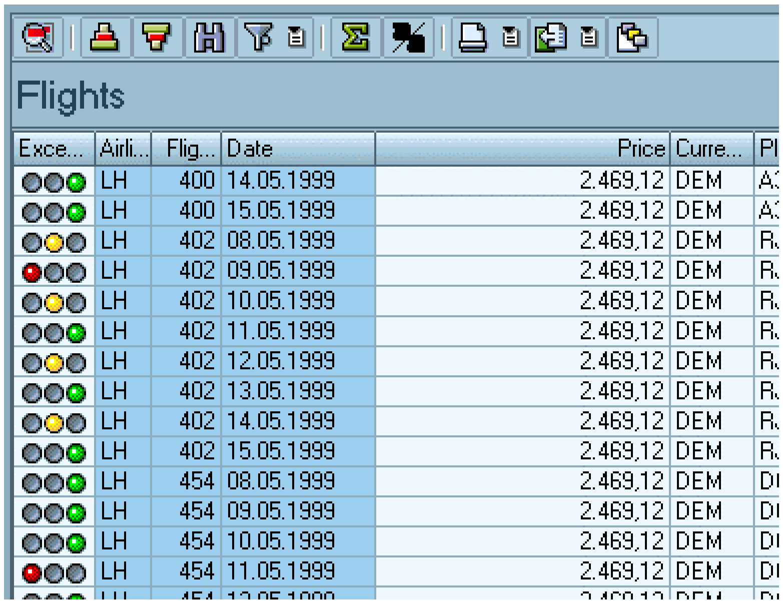
4. If you want to display the exception as an LED, you must set field **EXCP_LED** of the layout structure.
5. Read one row of the output table at a time in a loop, and query the fields that are related to the exception. Set your variable for the exception display format (which is **LIGHT** in our example) to '1', '2' or '3' (see above), depending on the threshold value you have chosen.

8. Pass the layout structure and the output table using method set_table_for_first_display [Page 100].

If you have changed the values for an exception in the output table, refresh the output using method refresh_table_display [Page 85].

Result

The ALV Grid Control contains an exception column at the beginning of the list:



Flights

Exce...	Airli...	Flig...	Date	Price	Curre...	Pl
🟢	LH	400	14.05.1999	2.469,12	DEM	A
🟢	LH	400	15.05.1999	2.469,12	DEM	A
🟡	LH	402	08.05.1999	2.469,12	DEM	R
🔴	LH	402	09.05.1999	2.469,12	DEM	R
🟡	LH	402	10.05.1999	2.469,12	DEM	R
🟢	LH	402	11.05.1999	2.469,12	DEM	R
🟡	LH	402	12.05.1999	2.469,12	DEM	R
🟢	LH	402	13.05.1999	2.469,12	DEM	R
🟡	LH	402	14.05.1999	2.469,12	DEM	R
🟢	LH	402	15.05.1999	2.469,12	DEM	R
🟢	LH	454	08.05.1999	2.469,12	DEM	D
🟢	LH	454	09.05.1999	2.469,12	DEM	D
🟢	LH	454	10.05.1999	2.469,12	DEM	D
🔴	LH	454	11.05.1999	2.469,12	DEM	D

Coloring Rows

Purpose

You can change the color of rows in the grid control to highlight data in the list.

The column color can be set using field **EMPHASIZE** of the field catalog (see: Output Options of Columns [Page 145]).

Process Flow

1. Define the layout structure [Page 157] of type **LVC_S_LAYO**.
2. Add a four-digit character field to your output table, as shown in the example below:

```
DATA: BEGIN OF GT_OUTTAB OCCURS 0.
        INCLUDE STRUCTURE <DDIC-Struktur>.
DATA:   linecolor(4) type c.  "Color for corresponding line
DATA: END OF GT_OUTTAB.
```

3. Select your data and copy it into the output table.
4. Read one row of the output table at a time in a loop. To change the color of a row, assign a four-digit color code to the character field.

   For more information on color codes, see field **EMPHASIZE** of the field catalog (see Output Options of Columns [Page 145]).
5. Assign the name of the internal table to field **INFO_FNAME** of the layout structure (in our case **'LINECOLOR'**, see step 2).
6. Pass the layout structure and the output table with method set_table_for_first_display [Page 100].

   If you do not want to color the cells for the first display, postpone step 4 and refresh the output table with method refresh_table_display [Page 85].

Coloring Cells

Purpose

You can change the color of cells in the grid control to highlight data in the list. Although you can also use this process to color entire rows, this is much more time-consuming than the procedure described in Coloring Rows [Page 31].

The D&D Behavior for Special Rows/Cells [Page 27] process is the analogous method to the method for selecting cells.

Process Flow

1. Define the layout structure [Page 157] of type **LVC_S_LAYO**.
2. Add a color table of type **LVC_T_SCOL** to your output table, as shown in the example below:

```
DATA: BEGIN OF GT_OUTTAB OCCURS 0.
        INCLUDE STRUCTURE <DDIC-Struktur>.
DATA:   CT TYPE LVC_T_SCOL.  "Table for colors
DATA: END OF GT_OUTTAB.
```

3. Select your data and copy it into the output table.
4. Read one row of the output table at a time in the loop. One row of the color table has three fields. If field **NOKEYCOL** is set, you can change the color of key fields. Assign values to the remaining fields as follows:
   – If you want to color the entire row, assign the corresponding values to the fields of structure **COLOR**. Field **fname** does not receive a value in this case.
   – If you want to color specific columns of the row only, you must append one row for each column to the color table. Assign the name of the desired column to field **fname**, and the corresponding values to the fields of structure **COLOR**.

   For information on the meaning of the color settings, see F1 help on the format command. Class **CL_GUI_RESOURCES** contains constant attributes for fields **INT** and **INV** of structure **COLOR**.

5. Assign the name of the internal table to field **CTAB_FNAME** of the layout structure (in our case **'CT'**, see step 2).
6. Pass the layout structure and the output table with method set_table_for_first_display [Page 100].

   If you do not want to color the cells for the first display, postpone step 4 and refresh your output table with method refresh_table_display [Page 85].

Displaying Cells as Pushbuttons

Purpose

If you assign the style **`mc_style_button`** to cells, rows or columns, the ALV Grid Control displays the associated cells as pushbuttons. Users know immediately that they can obtain further information about the cell by clicking on the pushbutton. The ALV Grid Control then triggers the event button_click [Page 109].

Process Flow

To display all cells of a column as pushbuttons, you use the field STYLE of the field catalog [Page 145].

To display rows or individual cells as pushbuttons, proceed as follows:

6. Define the layout structure [Page 157] of type **`LVC_S_LAYO`**.
7. Add a row table of type **`LVC_T_STYL`** to your output table, as shown in the following example:

```
DATA: BEGIN OF GT_OUTTAB OCCURS 0.
        INCLUDE STRUCTURE <DDIC structure>.

DATA:   CT TYPE LVC_T_DRDR.   "Table buttons

DATA: END OF GT_OUTTAB.
```

8. Select your data and copy it into the output table.
9. Read one output table row at a time in a loop. One row of the row table has the fields **`FIELDNAME`** and **`STYLE`**. Assign values to these fields as follows:
   – If you want to display all cells of a row as pushbuttons, assign the attribute **`cl_gui_alv_grid=>mc_style_button`** to the field **`style`**. In this case, field **`fieldname`** remains empty.
   – If you want to display only specific columns of the row as pushbuttons, append one row for each column to the cell table. Assign the name of the desired column to the field **`fieldname`**, and assign the attribute **`cl_gui_alv_grid=>mc_style_button`** to the field **`style`**.
9. Assign the name of the internal table to the field **`stylefname`** of the layout structure (in our case **`'CT'`**, see step 2).
10. Pass the layout structure and the output table using method set_table_for_first_display [Page 100].

Result

The ALV Grid Control displays all cells to which the attribute has been assigned as pushbuttons.

Including Hyperlinks

Purpose

You use the field HREF_HNDL [Page 145] of the field catalog to assign a handle for a hyperlink address to **all** cells of a column.

This section describes how you link the cells of a column with different hyperlink addresses.

Process Flow

1. Define an internal table for the field catalog with reference to type **`LVC_T_FCAT`** and a hyperlink table with reference to type **`LVC_T_HYPE`**:

```
data: gt_fieldcat type lvc_t_fcat,
      gt_hypetab type lvc_t_hype.
```

2. If you want to display a hyperlink in only one cell of a column, you need an additional field in your output table. Add a field of type **`INT4`** to your output table, as shown in the following example:

```
DATA: BEGIN OF GT_OUTTAB OCCURS 0.
        INCLUDE STRUCTURE <DDIC structure>.

DATA:   HL_FNAME TYPE INT4.  "hyperlink handle for field FNAME

DATA: END OF GT_OUTTAB.
```

   For each column of the output table that should have hyperlinks at cell level, you need to define an additional field for the handle. In this example, hyperlinks should appear in one column only.

3. Generate the field catalog for your output table either manually [Page 133] or semi-automatically [Page 135]. In the field catalog, assign the name of the field for the hyperlink handle to the field **`WEB_FIELD`** (in our example: **`HL_FNAME`**, see above). The ALV Grid Control is then able to correctly interpret the additional field.

4. Create your hyperlink table with all desired navigation targets, as shown below:

```
data: ls_hype type lvc_s_hype.

ls_hype-handle = '1'.
ls_hype-href = 'http://www.sap.com'.
append ls_hype to gt_hypetab.

ls_hype-handle = '2'.
ls_hype-href = 'http://www.mysap.com'.
append ls_hype to gt_hypetab.
```

5. Select your data and copy it into the output table.

6. Read one output table row at a time in a loop. Assign the desired handle to the field for the hyperlink handle (in our example: **`HL_FNAME`**). If the handle remains initial for a row, no hyperlink is displayed.

11. Pass the field catalog, the hyperlink table, and the output table using method set_table_for_first_display [Page 100].

Result

All cells of the column to which a handle has been assigned using the additional field, are displayed as hyperlinks. If no entry exists in the hyperlink table for the handle assigned, the cell contents are displayed in the normal way.

Grouping Fields for Field Selection

Purpose

If you have numerous output fields, the number of fields available on the selection screen for the current layout becomes overwhelming. If you group the output fields, the user can then restrict the number of fields displayed on the selection screen by choosing the relevant group from a list box.

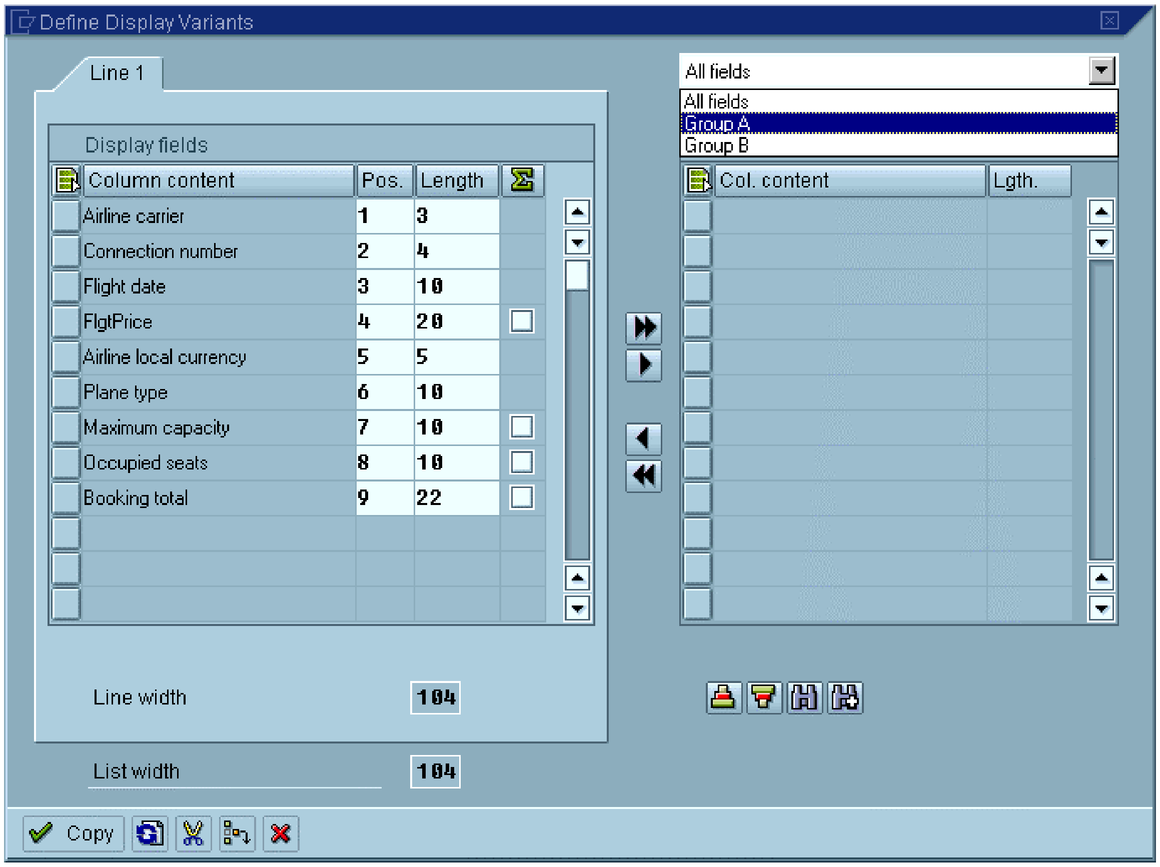
The list box refers only to the hidden fields.

Process Flow

1. Define an internal table of type **`LVC_T_SGRP`** for the *field group texts* and an internal table of type **`LVC_T_FCAT`** for the field catalog.
2. Set up the field catalog manually [Page 133] or semi-automatically [Page 135].
3. Use field **`sp_group`** of the field catalog to assign a group key for each field of the output table.
4. Define a group text in your field group table for each group.
5. Pass the field group table and the field catalog with method set_table_for_first_display [Page 100] (using parameter **`IT_SPECIAL_GROUPS`** or **`IT_FIELDCATALOG`**).

Result

The system displays a list box on the selection screen for the current layout that can be used to select the relevant group:



Define Display Variants

Line 1

Display fields

Column content	Pos.	Length	Σ
Airline carrier	1	3	
Connection number	2	4	
Flight date	3	10	
FlgtPrice	4	20	☐
Airline local currency	5	5	
Plane type	6	10	
Maximum capacity	7	10	☐
Occupied seats	8	10	☐
Booking total	9	22	☐

All fields
- All fields
- Group A
- Group B

Col. content	Lgth.

Line width 104

List width 104

Using the Conversion Exit

Purpose

You use a conversion exit to define a jump to a conversion routine for a column of your output table. As the exit you specify the **<conv>** part of a function module called **CONVERSION_EXIT_**<conv>**_OUTPUT**. For example, you can use conversion exit **ALPHA** (see function module **CONVERSION_EXIT_ALPHA_OUTPUT**) to suppress leading zeros of account numbers.

The conversion exit is implemented through WRITE addition **USING EDIT MASK**.

Prerequisites

So that the ALV Grid Control can perform the conversion, it must know the internal and the external length of the field, which is the same as the length before and the length after the conversion.

The following example that uses editing template **EDIT MASK** for the WRITE command shows you how these lengths are specified:

```
DATA TIME TYPE T VALUE '154633'.
WRITE (8) TIME USING EDIT MASK '__:__:__'.  "Output: 15:46:33
```

In this example, the internal length (of type T) is six characters, while the output length is eight.

The column width (which can be set using the current layout) does not depend on these values and needs not be adjusted accordingly.

Process Flow

1. Declare the internal and the external length of the field to the ALV Grid Control.

- For fields with DDIC reference, the ALV Grid Control automatically uses the internal and the external length.
- For fields without DDIC reference, you must specify the internal length using field **INTLEN** and the external length using field **DD_OUTLEN** of the field catalog (see Parameters for Fields Without DDIC Reference [Page 153]).

2. Specify the conversion exit using field **EDIT_MASK** of the field catalog (see Formatting Column Contents [Page 148]).
3. Pass the field catalog with method set_table_for_first_display [Page 100] before the list is displayed for the first time.

Result

The values of the columns are run through the conversion routine before display.

Using the ALV Grid Control in WANs

The protocol used for communication between the application server and the frontend ensures that the ALV Grid Control can be used in WANs. The protocol controls the data transfer from the application server to the frontend.

Initially, the control receives only the data needed to display the lines currently visible. All other data is only transferred to the control if the user scrolls there. In the WAN, packages are rather small so that each package can be transferred within a short time.

Constraints

Programmers who use the ALV Grid Control must consider one special aspect for WANs. Events **`DELAYED_CALLBACK`** and **`DELAYED_CHANGED_SEL_CALLBACK`** should be used in WANs in exceptional cases only. The reason is not to be found in the events themselves but in the fact that other controls are frequently filled with data in response to these events. Since this leads to considerable delays (> 3 seconds) in many WANs, users may feel uncomfortable with this situation.

The ALV Grid Control in the Web

To display ALV lists in the Web, you use functions of the ALV Grid Control [Ext.] on the backend. Basically, there are two ways to display ALV lists:

- In the SAP GUI for HTML [Page 44]. This corresponds - with restrictions - to a 1:1 mapping of the transaction in the SAP System.
- In the Workplace [Page 48]. Highly simplified versions of the ALV Grid Control (MiniALV and MidiALV) are used to integrate lists into applications of the Workplace.

It is indeed possible to display an SAP transaction with an ALV Grid Control in the Workplace. In this case, the SAP GUI for HTML is integrated into the Workplace.

Display in the SAP GUI for HTML

Purpose

The SAP GUI for HTML allows you to display SAP transactions in a Web browser. Applications using the class `CL_GUI_ALV_GRID` for displaying table data are therefore automatically visible in the Web through this SAP GUI.

Implementation Considerations

In the SAP GUI for HTML, the ALV Grid Control is displayed differently than in the SAP GUI for Windows. The SAP GUI for HTML supports fewer colors, for example. Also, user interaction for several processes is restricted or has been implemented differently.

Features

In most cases, the methods, events, and attributes of the class `CL_GUI_ALV_GRID` also work in the SAP GUI for HTML.

No keyboard commands are available in the SAP GUI for HTML. Functions are normally called by clicking the mouse. This is why the F1 help has been replaced by a question mark icon in the toolbar.

The user is required to select and deselect rows, columns or cells individually. (This also depends on the selection mode [Page 159]).

Restrictions

To ensure high performance, several functions are available on a restricted basis only. See the table below for a list of the most important restrictions.

Most important restrictions in the SAP GUI for HTML

Restriction	Classification (I: Interface V: Visual A: Interaction)
No context menu is available. You do not define menus in the toolbar as dynamic but as static [Page 46].	I/A
The scrollbar position cannot be set (see set_scroll_info_via_id [Page 93]).	I
Due to missing fonts, no symbols can be displayed.	I/V
Users can select only one table element type (row, column, cell) at a time. Combined selections are not possible.	I/A
The events delayed_callback [Page 111] and delayed_changed_sel_callback [Page 112] cannot be used (see also Using the ALV Grid Control in WANs [Page 42]).	I/A
The user cannot swap columns using Drag and Drop.	I/A
No autoscroll function is available (the system does not scroll automatically when the user keeps the mouse button pressed).	A

Subtotals lines cannot be compressed.	A
The toolbar is not wrapped automatically if the number of functions available do not fit into one line.	V/A
Only eight background colors are used. Other color values (such as *intensified*) are mapped to other color values.	V

Defining a Static Menu

Use

In the SAP GUI for Windows™, you set up menus in the toolbar dynamically using the events toolbar [Page 127], menu_button [Page 115] and context_menu_request [Page 110]. In the SAP GUI for HTML, you are not recommended to follow this procedure for reasons of performance. Instead, you should define all menus of the toolbar as static as described below (context menus are generally not supported). You define these menus at the event **toolbar** before the ALV Grid Control is displayed.

Process Flow

7. Define an event handler method for the event **TOOLBAR**.

8. Declare a structure for defining an element in the toolbar, a menu structure and a reference variable to the context menu class:

```
data: ls_toolbar  TYPE stb_button,
      ls_menu    TYPE stb_btnmnu,
      lc_menu    TYPE REF TO cl_ctmenu.
```

9. Define a pushbutton of the menu type in the toolbar first:

```
CLEAR ls_toolbar.
ls_toolbar-function   = 'STATIC_MENU1'.
ls_toolbar-icon       = icon_led_interactive.
ls_toolbar-butn_type = '2'.
ls_toolbar-text       = 'first static menu'.
```

You use the field **butn_type** to communicate the type of the GUI element to the ALV Grid Control. For menus with a default button, you use type '**1**'.

10. Use the event parameter **E_OBJECT** to append the new definition to the table **mt_toolbar**:

```
APPEND ls_toolbar TO e_object->mt_toolbar.
```

11. Create an instance of the context menu class, and use the method **ADD_FUNCTION** to add functions to the menu (in this example, only one function is added):

```
create object lc_menu.
call method lc_menu->add_function
   exporting fcode   = 'STATIC_MENU1_FUNC1'
   text              = 'first menu entry'.
```

12. To link the menu entry with the menu, you fill the menu structure accordingly:

```
ls_menu-ctmenu = lc_menu.
ls_menu-function = 'STATIC_MENU1'.
```

13. Pass the menu structure to the event parameter **E_OBJECT**:

```
append ls_menu to e_object->mt_btnmnu.
```

14. Call the static method **set_focus** for your instance:

```
call method cl_gui_control=>set_focus
    exporting control = sender.
```

Result

You can retrieve the function code that you have defined in the field **`function`** in the event handler method for the event **`USER_COMMAND`** to implement the associated function.

Display in the Workplace

The Workplace technology allows you to group applications based on the requirements of a specific type of user. Since these applications run in a Web browser, they are designed for a different user community than applications running in the SAP System, and are easier to use.

You can find more information on the Workplace in SAPnet using the alias *workplace*.

For Web applications that display lists in the Workplace, two versions of the ALV Grid Control are available. Depending on the complexity or length of a list, you either use the MiniALV [Page 50] or the MidiALV [Page 60].

Integration

Both the MiniALV and the MidiALV build on functions of the ALV Grid Control [Ext.] on the backend. To display a report in the Web, you need a report or a query (which generates a report) in the SAP System. For Web Reporting, the report results are used and transferred to the Web browser.

This report is called *proxy report* in the following.

Before a report can be displayed in the Workplace using the MidiALV or MiniALV, the report must fulfill certain prerequisites [Page 49].

Prerequisites

The prerequisites described below apply both to the MiniALV and the MidiALV.

Prerequisites for the Report

In Web Reporting, the lists are displayed in the Web browser. Therefore, the proxy report must not create any frontend objects.

If the proxy report outputs lists in the fullscreen mode (application functions are located in the application toolbar), the application is not required to adapt the ABAP program for the proxy report.

If the proxy report uses the class **`CL_GUI_ALV_GRID`**, the report must determine whether it runs in the SAP GUI or in the Web. To allow the report to do this, you use the static method offline() [Page 84] which you call instead of reading the system field **`SY-BATCH`**. If the proxy report then executes in batch mode and writes the results into the spool, this report is automatically suited for output as MiniALV or MidiALV. If the report is nevertheless not displayed in the Web, check if the report and the selection variants or layouts specified exist in the system (for queries, the user group must be available).

Authorizations

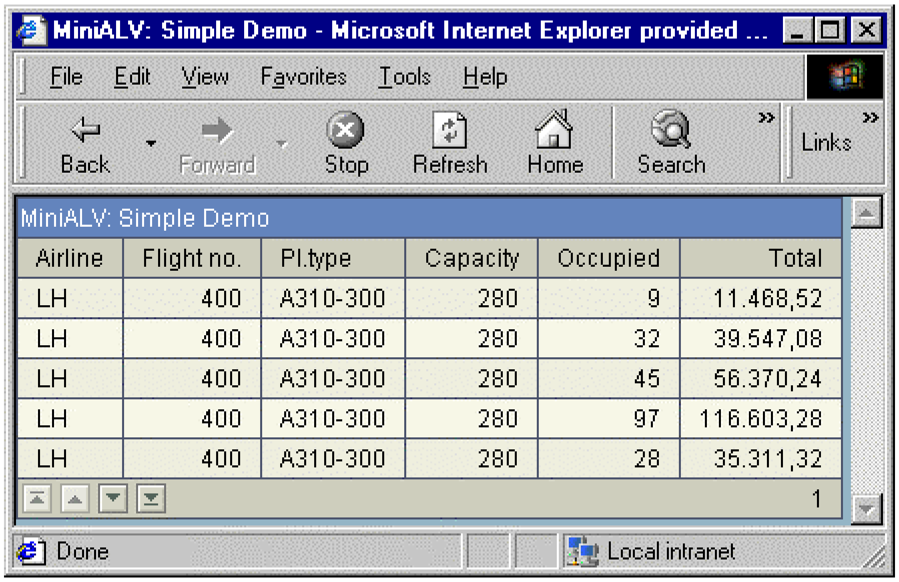
To enable the report or query to be displayed in the Web, you must additionally make two settings:

- You must release the report using the Web Repository (transaction **`SMW0`**) (see Releasing Objects for Web Reporting [Ext.]).
- You must assign the report to an authorization group. Customers use the report **`RSCSAUTH`** to use their own authoriation groups without making modifications to the standard system.

The MiniALV

Use

The *MiniALV* is designed for displaying lists in a small output area of the screen (for example, overview of system processes). For this reason, you should not output subtotals in these lists. The MiniALV can be used both as a stand-alone MiniApp and as part of a MiniApp. User interaction is restricted to the essential functions.


MiniALV: Simple Demo - Microsoft Internet Explorer provided ...
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Links
Done
Local intranet

MiniALV: Simple Demo

Airline	Flight no.	Pl.type	Capacity	Occupied	Total
LH	400	A310-300	280	9	11.468,52
LH	400	A310-300	280	32	39.547,08
LH	400	A310-300	280	45	56.370,24
LH	400	A310-300	280	97	116.603,28
LH	400	A310-300	280	28	35.311,32

1

The above screen shot shows the MiniALV in page mode (more pages can be displayed by using the scroll pushbuttons in the bottom left corner). The example displays a report from the SAP System in the Web browser that outputs the fields `Airline`, `Flight no.`, `Pl.type`, `Capacity`, `Occupied` and `Total` for flight table **`SFLIGHT`**. If layouts or selection variants have been saved for a report, the MiniALV displays a list box above the list from which the user can choose one of the available list variants.

Prerequisites

The function module **`FLOW_LOGIC_ALV_CALL`** must be available in the system. This function module is called through WebRFC to display list data in the browser.

See also: Prerequisites [Page 49] and Error Analysis [Page 59].

Features

The MiniALV is implemented by means of the flow logic technology. While the tutorial Internet Application Development With Flow Files [Ext.] describes how you use the *SAP@Web Studio* for this purpose, the documentation Integration of Internet Services [Ext.] explains how to implement a MiniApp using the *Web Application Builder*.

The following possibilities are available to integrate the MiniALV into the Workplace:

- You start the MiniALV service as a stand-alone MiniApp [Page 52] using a parametrized URL.
- You integrate the MiniALV service into a MiniApp of your own [Page 53] and therefore do not need a flow file.
- You use the MiniALV as an interactive Web control [Page 54] and therefore need a flow file to respond to flow events.

The user can choose selection variants or layouts saved from a list box in the MiniALV. Also, hyperlinks [Page 36] embedded into the list are active in the MiniALV.

Using a service parameter [Page 62] (**`ma_style=wap1`**), you can serialize the ALV table for the output on small screens [Page 57]. This ensures that the data can be output on very small screens for mobile devices (*pervasive computing*).

Constraints

The ITS cannot process flow events of services included. If you want to process your own flow events, you must embed the MiniALV as an interactive Web control (see above).

As a MiniApp

Use

You want to display a report from the SAP System in the Web browser und assign it to a role. This means that the report is not to be integrated into a MiniApp of your own but started directly as a MiniApp itself.

Procedure

1. You start the MiniALV using a URL for your report. The URL has the following form:

<WEBSERVER> [Page 64]**/minialv/!?report=**<report name>**&**<p1>**=**<w1>**&**...**&**<pn>**=**<wn>

You specify the proxy report [Page 50] and additional service parameters [Page 55] (<p1> to <pn>) in this URL.

2. You specify the URL in the role description in transaction **PFCG**. For more information on creating roles, see Users and Roles (BC-CCM-USR) [Ext.].

   You must specify a placeholder in the URL for the path of the Web server. Read the section *The URL in the Workplace* in Calling the Internet Service [Page 64].

Result

The user assigned to the role can call the report in the Workplace.

As an Extension of a MiniApp

Use

You display a report using the MiniALV as an extension of a MiniApp of your own. This allows you to implement other functions in the MiniApp and control the look and feel of the MiniApp in which the MiniALV is displayed.

Prerequisites

You must not use a flow file of your own.

Procedure

You include the MiniALV service in your template and set global variables in advance to pass the name of the proxy report [Page 50] and other parameters [Page 55] (<p1> to <pn) to the MiniALV:

```
`include(~service="system",~theme="dm",~name="templateLibraryDHTML.html
")`
...
`report = "<report name>"`
`<p1> = "<w1>"`
...
`<pn> = "<wn>"`
`include(~service="minialv", "minialv")`
```

Result

When the MiniApp is started, the MiniALV displays the proxy report.

As an Interactive Web Control

Use

You display a report using the MiniALV as an extension of a MiniApp of your own. This allows you to implement other functions in the MiniApp and control the look and feel of the MiniApp in which the MiniALV is displayed. If you use the MiniALV as an interactive Web control, you can process flow events in the MiniApp.

Procedure

1. You include the MiniALV service in your template and set global variables in advance to pass the name of the proxy report [Page 50] and other parameters [Page 55] (<p1> to <pn>) to the MiniALV:

```
`include(~service="system",~theme="dm",~name="templateLibraryDHTML.html
")`
...
`report = "<eport name>"`
`<p1> = "<w1>"`
...
`<pn> = "<wn>"`
`include(~service="minialv", "webcontrol")`
```

2. To respond to flow events, define a state such as `loadList` in your flow file in which the MiniALV is used. Copy the module call of the MiniALV from its flow file into this state. You can find the flow file of the MiniALV in the MiniALV service in the SAP System.

   As soon as the ITS is able to process flow events of includes, the above procedure will become obsolete.

3. End the other states in your main MiniApp with the following statement:

```
<default next_state="loadList"/>
```

This additionally executes the state `loadList` and reloads the table contents.

Result

Your MiniApp can now define flow events of its own and process them using the MiniALV.

Service Parameters

Content

Parameter	Meaning
`report`	Name of the proxy report [Page 50]
`query`	Name of a query (as an alternative to the proxy report)
`usergroup`	For query: user group
`workspace`	For query: query area

Personalisation

Parameter (default value)	Meaning
`variant`	Selection variant with which the report is to be started
`emptyvariant (on|off)`	Allows you start the MiniALV with an empty selection variant
`layout`	Start the MiniALV with an existing layout [Page 16]
`selectiontype (V|P)`	Determines the options available in the list box: • V: The list box displays saved selection variants and/or saved layouts (that are assigned to the report) • P: The application assigns its own values to the list box (by means of a user exit, see `userfb`). Refer to the service `CCMS_ALVIEWER` and the function module `SALWP_AL_VIEW_GET_SHUFF_INFO`.
`userfb`	Function module as a user exit to: • Customize MiniALV parameters • Save settings as user-specific

Visual Properties

Parameter (default value)	Meaning
`ma_title (on|off)`	Show/hide title
`ma_header (short|medium|long|off)`	Affects the text of the column headers
`ma_shuffler (on|off)`	Switch on/off list box display
`ma_style (scroller|pages|text|wap1)`	Change graphical design style in the Web browser
`ma_lines (10)`	Number of lines to be visible simultaneously
`ma_height (150)`	Height of scrollbar table

Properties for 'ma_style=wap1' (see Output on Small Screen [Page 57])

Parameter (default value)	Meaning

`maw_columns`	You can use this parameter to determine which columns to display and which labels to use for the cells.
`maw_sparse`	Summarizes the table. Recurring column values are not output again.

Output on Small Screens

You can use the MiniALV to output ALV lists on very small screens. To do this, you must serialize the columns of the list to make them fit onto the small output area. If you want to use the MiniALV for this purpose, you must set the service parameter `ma_style` auf `wap1`.

Normally, users need to display only certain columns of the ALV list. The MiniALV provides additional service parameters [Page 55] that are designed only for the output style `wap1`. They begin with the prefix `maw_`.

Display Formatting

You can use parameter `maw_columns` to determine which columns to display and which labels to use for the relevant cells. The parameter value is a string with a list separated by commas in which each position corresponds to a column. One position contains the label provided for the value. In addition, control commands are available which you can use to format the display.

Control command	Meaning
`^`	If you use this character at the end of a position, the next column value is displayed in the same line without line break.
`,,`	If you use two commas in sequence, the corresponding position is not displayed.
`_`	The corresponding position is displayed but without label.
`*0`	The normal column header is used for the position.
`*n`	The first n characters of the normal column header are used for the position.

Example

The following example shows an ALV list, the control commands used and the resulting output:

Output on Small Screens

Short	Airline	#	Date	Capacity	Occupied
AA	American	17	03.02.00	660	97
AA	American	18	04.02.00	660	130
LH	Lufthansa	907	03.02.00	660	30
LH	Lufthansa	908	05.02.00	660	60

```
maw_columns=
>^,,^,; ,Capacity=,
Occupied=
```

```
>AA 17; 03.02.00
Capacity= 660
Occupied= 97
>AA 18; 04.02.00
Capacity= 660
Occupied= 130
...
```

```
maw_columns=
>^,,^,;,*3=,*3=
```

```
>AA 17; 03.02.00
Cap= 660, Occ= 97
>AA 18; 04.02.00
Cap= 660; Occ= 130
...
```

```
maw_columns=
,>,#^,; ,*3=,*3=
maw_sparse=2
```

```
>American
>>#17; 03.02.00
Cap= 660; Occ= 97
>>#18, 04.02.00
Cap= 660; Occ= 130
>Lufthansa
...
```

Error Analysis

Notes

- Use the function module **`GUI_IS_ITS`** or the static method **`CL_GUI_OBJECT=>WWW_ACTIVE`** instead of function module **`ITS_PING`**
- If error code `E00` (see below) is returned, first check the prerequisites for using the MiniALV [Page 50]. You can also test if the function module **`FLOW_LOGIN_ALV_CALL`** returns data of the report.
- If you use the MiniALV in combination with an SAP System of a previous release, report names (and selection variants, and so on) may be case-sensitive.

Error Messages

If errors occur, the MiniALV issues messages. In the Web control [Page 54] mode, you can handle the error code in the flow file:

Error code	Meaning
E00	RFC did not return data
E01	Variant does not exist
E02	Error determining the user-specific settings
E03	Error calling the ALV
E04	Error determining the variants
E05	Report does not exist
E06	Error in the selection string; check the syntax
E08	Function module for determining the user-specific variants does not exist
E09	Report cannot be executed without a variant; specify a variant
E10	User group does not exist
E11	Query does not exist
E12	Query is locked
E13	Report generation for the query was terminated
E14	Error generating the report for the query
E15	Specify a report or a query

The MidiALV

Use

The *MidiALV* is suited for displaying rather long lists in the Web. The list appears in the entire output area of the Workplace. The MidiALV also adopts selection screens from the SAP System.

Prerequisites

The MidiALV can only display reports in the Web that use **one** ALV Grid Control as output. The function modules **`WWW_ALV_CALL`** and **`WWW_ALV_SELSCREEN`** must be available in the SAP System used. These modules are called through WebRFC (depending on the application) to display list data in the browser.

See also Prerequisites [Page 49].

Features

You use a parameter to determine if you want to display a selection screen (if available) in the MidiALV (**`WWW_ALV_SELSCREEN`**) or if you simply want to display the results list for a selection (**`WWW_ALV_CALL`**).

The user can choose selection variants or layouts saved from a list box in the MidiALV. Also, hyperlinks [Page 36] embedded into the list are active in the MidiALV.

Special Features of the MidiALV Selection Screen

The results list contains a button `Back to selection screen with which the user goes back to the selection screen in order to select new data.`

The MiniALV selection screen is subject to the following constraints:

- No F4 input help available
- No F1 help available
- No complex restrictions using select-options supported
- No dynamic screen modifications within the PBO supported

Calling the MidiALV

Use

You call the MidiALV through a URL to display it in the Web browser. If you assign this URL to a role, the MidiALV can be integrated into the Workplace.

Procedure

1. You start the MidiALV using a URL for your report. The URL has the following form:

<WEBSERVER> [Page 64]**/webrfc/!?_function=**<function>**&<p1>**=<w1>...**&<pn>**=<wn>

Set:

- `_function=www_alv_call` if you want to display a report without selection screen
- `_function=www_alv_selscreen` if the report has a selection screen that you do not want to skip

To identify the report, you use the following service parameters [Page 62]:

- Parameter `_report` if you refer to a report or
- Parameter `_query _usergroup` and optionally `_workspace` if you refer to a query

2. You specify the URL in the role description in transaction `PFCG`. For more information on creating roles, see Users and Roles (BC-CCM-USR) [Ext.].

   You must specify a placeholder in the URL for the path of the Web server. Read the section *The URL in the Workplace* in Calling the Internet Service [Page 64].

Result

The MidiALV call for your report is now linked to a role. If this role is used in the Workplace, the user assigned to that role can call this report.

Service Parameters

The URL for calling the MidiALV has the following form:

<WEBSERVER> [Page 64]**/webrfc/!?_function=**<function>**&<p1>**=<w1>...**&<pn>**=<wn>

Using the required parameter `_function` you enter the function module that is to be called in the SAP System:

- `www_alv_call`: No selection screen available or required
- `www_alv_selscreen`: Display report selection screen

Besides the function, you must specify the name of a report or the name of a query in the URL.

Content

Parameter	Meaning
`_report`	Name of the proxy report [Page 60]
`_query`	Name of a query (as an alternative to the parameter `_report`)
`_usergroup`	For query: user group
`_workspace=global`	For query (optional): global query area

Personalisation

Parameter	Meaning
`_variant`	Name of the selection variant to be used. (If the ALV is called with a selection screen (*www_alv_selscreen*), appropriate values are preassigned to the selection screen).
`_layout`	Name of the layout [Page 16] with which the list is to be displayed

Visual Properties

Parameter (default value)	Meaning
`_fontsize (1-5)`	Selection of the font type: • 1 = Arial 7pt • 2 = Arial 8pt • 3 = Arial 9pt • 4 = Arial 10pt • 5 = Arial 11pt If this parameter is not set, the font type is determined by the browser setting.

`_header (small|medium|long)`	Selection of the column header: • *small* = Short text • *medium* = Medium text • *long* = Long text
`_shuffler (on|off)`	Switch on/off the display of the list box from which users can choose a selection variant or a layout

Calling the Internet Service

In order to display MiniApps in the Workplace, you publish Internet services with the ITS. The MiniApp can then be started by means of a URL.

See also: Integration of Internet Services [Ext.].

Normally, the URL has the following form:

`<web_protcl>://<web_server>/<web_path_prefix>/<service>/!?<CGI_parameters>`

The URL section **`<web_protcl>://<web_server>/<web_path_prefix>`** is also shortly referred to as **`<WEBSERVER>`** in this documentation.

The section up to and including the exclamation mark ('`!`') describes the Internet service on the relevant ITS. Following the question mark ('`?`'), there are the CGI parameters to which you assign a value directly in the URL. You separate them by an ampersand ('`&`'). For example, you can use the following value for <CGI_parameters>:

`~language=<language>&~client=<client>`

The parameters **`~language`** and **`~client`** can also be passed to the ITS by means of a service file.

The URL in the Workplace

If you add your URL to the Workplace using transaction **`PFCG`**, you must not resolve the string **`<web_path_prefix>`** since it is specific to the Web server. The Workplace replaces this section of the address with the correct path.

Examples

Testing and Calling the MiniALV

To start the MiniALV service with a report called **`enjoy`** on the ITS using the URL **`http://rosebud:1080`** and the Web server prefix **`scripts/wgate/`** (for testing purposes), you call the URL

`http://rosebud:1080/scripts/wgate/minialv/!?report=enjoy`

in the Web browser. In this case, **`<web_path_prefix>`** corresponds to **`scripts`**. In transaction **`PFCG`**, you must then enter the URL

`http://rosebud:1080/<web_path_prefix>/minialv/!?report=enjoy`

for this service

Testing and Calling the MidiALV

To start the MidiALV service with a report called **`list`** on the ITS using the URL **`http://big:1080`** and the Web server prefix **`scripts/wgate/`** (for testing purposes), you call the URL

`http://big:1080/scripts/wgate/webrfc/!?_function=www_alv_call&_report=list`

in the Web browser. In this case, **`<web_path_prefix>`** corresponds to **`scripts`**. In transaction **`PFCG`**, you must then enter the URL

`http://big:1080/<web_path_prefix>/webrfc/!?_function=www_alv_call&_report=list`

for this service.

Methods of Class CL_GUI_ALV_GRID

This class contains both control-specific methods and methods inherited from the Control Framework. This section describes only the control-specific methods. The methods inherited from the Control Framework are described in Methods of the OO Control Framework [Page 167].

Basic Methods

Method	Application
CONSTRUCTOR [Page 69]	Generate an instance of the ALV Grid Control
set_table_for_first_display [Page 100]	Display an output table in the control
refresh_table_display [Page 85]	Refresh the data displayed in the control

Elements of the Grid Control

Method	Application
get_current_cell [Page 70]	Get indexes and properties of a selected cell
set_current_cell_via_id [Page 89]	Set cursor on a specific cell
get_scroll_info_via_id [Page 75]	Get current scroll position
set_scroll_info_via_id [Page 93]	Set scroll position
get_selected_cells [Page 76]	Get content and position of selected cells from frontend
set_selected_cells [Page 95]	Select cells in the ALV Grid Control
get_selected_cells_id [Page 77]	Get indexes of currently selected cells
set_selected_cells_id [Page 96]	Select cells using index table
get_selected_columns [Page 78]	Get field names of selected columns from frontend
get_selected_rows [Page 79]	Get indexes of selected rows from frontend
set_selected_rows [Page 98]	Select rows

Layout and Structure Informationen

Method	Application
get_frontend_fieldcatalog [Page 73]	Get current field catalog from frontend
set_frontend_fieldcatalog [Page 91]	Set field catalog at frontend
get_frontend_layout [Page 74]	Get layout structure at frontend
set_frontend_layout [Page 92]	Set layout structure at frontend
set_3d_border [Page 87]	Enable/disable 3D format of ALV Grid Control border

Generic Functions

Method	Application
get_filtered_entries [Page 71]	Get indexes of rows that are not displayed due to a filter set
get_filter_criteria [Page 72]	Get filter properties for all columns that have a filter set currently
get_subtotals [Page 81]	Get values of current subtotals lines
set_filter_criteria [Page 90]	Set filter properties for columns
get_sort_criteria [Page 80]	Get sort criteria for sorted columns
set_sort_criteria [Page 99]	Set sort criteria for columns
get_variant [Page 83]	Get current layout
save_variant_dark [Page 88]	Disallow users to assign their own layout name
set_graphics_container [Page 94]	Use another container control for diagram display
set_user_command [Page 104]	Change function code to be executed currently

Event Handling

Method	Application
register_delayed_event [Page 86]	Register event **`DELAYED_CALLBACK`** or **`DELAYED_SEL_CHANGED_CALLBACK`**
set_toolbar_interactive [Page 103]	Trigger event toolbar [Page 127].

Interfaces

Method	Application
activate_reprep_interface [Page 68]	Enable report/report interface
offline [Page 84]	Check if the ALV Grid Control runs without frontend handling

activate_reprep_interface

Use

Activate the report/report interface. Using parameter **IS_REPREP**, you specify the sender of the report. If table **TRSTI** contains an assignment of the sender report to receiver reports, function code **BEB**x becomes active (x = function code class).

If a receiver assignment to report writer report group **7KOI** with function code class '3' (SAP setting) exists for sender **RKTFGS15**, this receiver report group is called through the report/report interface at function code **BEB3**. The selections passed to the report/report interface are the report selections and the key information of the selected row.

For more information on the report/report interface, see the documentation on function group **RSTI**.

Prerequisites

Report **RKKBRSTI** must be active in the development system.

Features

```
CALL METHOD <ref. var. to CL_GUI_ALV_GRID>->activate_reprep_interface
      EXPORTING
         IS_REPREP   =   <structure of type LVC_S_RPRP>.
```

Parameter	Meaning
IS_REPREP	Structure for identifying the sender

For an overview, see: Methods of Class CL_GUI_ALV_GRID [Page 66]

CONSTRUCTOR

Use

This method is called automatically if you use **`CREATE OBJECT`** to create an object of class **`CL_GUI_ALV_GRID`**. When the control is generated, it is created on the frontend and linked to a container control.

Features

```
CREATE OBJECT <reference variable to CL_GUI_ALV_GRID>
   EXPORTING
      i_shellstyle    = <var. of type I>
      i_lifetime      = <var. of type I>
      i_parent        = <ref. var. to CL_GUI_CONTAINER>
      i_appl_events  = <var. of type CHAR01>.
```

Parameter	Meaning
`I_appl_events`	If this parameter is set, the ALV Grid Control registers all events as application events. If the parameter is not set, all events are registered as system events.

The other parameters have the same meaning as in method constructor [Page 178] of class **`CL_GUI_CONTROL`**.

For an overview, see: Methods of Class CL_GUI_ALV_GRID [Page 66]

get_current_cell

Use

You use this method to get the value and properties of the grid control cell on which the cursor is placed.

If a column and no cell is selected, the ALV Grid Control sets the row index to 0 and returns only column-specific information. The ALV Grid Control reacts analogously if a row is selected.

The ALV Grid Control returns two row and column indexes: one that refers to the output table (this is the value that you normally need for further processing), and one that refers to the current display in the grid control. These values are different from each other if, for example, rows are not displayed as a result of a filter set.

Features

```
CALL METHOD <ref.var. to CL_GUI_ALV_GRID>->get_current_cell
      IMPORTING
         E_ROW         = <var. of type I>
         E_VALUE       = <var. of type C>
         E_COL         = <var. of type I>
         ES_ROW_ID     = <structure of type LVC_S_ROW>
         ES_COL_ID     = <structure of type LVC_S_COL>.
```

Parameter	Meaning
E_ROW	Row index of selected cell, related to the display in the grid control
E_VALUE	Value of selected cell
E_COL	Column index of selected cell, related to the display in the grid control
ES_ROW_ID	Structure with information on row type and index, related to the output table
ES_COL_ID	Structure with information on column field name, related to the output table

For an overview, see: Methods of Class CL_GUI_ALV_GRID [Page 66]

get_filtered_entries

Use

Returns a table with all indexes that are currently hidden on the screen using the standard "filter" function.

Features

```
CALL METHOD <ref.var. to CL_GUI_ALV_GRID>->get_filtered_entries
       IMPORTING
          ET_FILTERED_ENTRIES     = <internal table of type LVC_T_FIDX>.
```

Parameter	Meaning
ET_FILTERED_ENTRIES	Hashed table with one row index for each table row filtered

For an overview, see: Methods of Class CL_GUI_ALV_GRID [Page 66]

get_filter_criteria

Use

You can use this method to get the current filter settings and pass them on to other components of the system. A row of the table describes the selection conditions for column entries that are not displayed. Columns for which no filter is set are not listed in the table.

You should never 'manually' set up the table with the filter criteria.

Features

```
CALL METHOD <ref.var. to CL_GUI_ALV_GRID>->get_filter_criteria

      IMPORTING
         ET_FILTER     = <internal table of type LVC_T_FILT>.
```

Parameter	Meaning
`ET_FILTER`	Table with filter settings for columns that have a filter set

For an overview, see: Methods of Class CL_GUI_ALV_GRID [Page 66]

get_frontend_fieldcatalog

Use

Get the current field catalog from the frontend. You use this method if you want to modify the field catalog used during display (for hiding specific columns, for example).

See also: The Field Catalog [Page 129] **and** set_frontend_fieldcatalog [Page 91]**.**

Features

```
CALL METHOD <ref.var. to CL_GUI_ALV_GRID>->get_frontend_fieldcatalog
       IMPORTING
          ET_FIELDCATALOG      = <internal table of type LVC_T_FCAT>.
```

Parameter	Meaning
`ET_FIELDCATALOG`	Structure description for the entire output table

For an overview, see: Methods of Class CL_GUI_ALV_GRID [Page 66]

get_frontend_layout

Use

Get the current layout structure from the frontend. In the layout structure, you determine properties of the grid control (see also: The Layout Structure [Page 157]).

Features

```
CALL METHOD <ref.var. to CL_GUI_ALV_GRID>->get_frontend_layout
   IMPORTING
      ES_LAYOUT    = <structure of type LVC_S_LAYO>.
```

Parameter	Meaning
ES_LAYOUT	Structure with fields for display options, graphical grid control properties, totals options, exceptions, colors, drag and drop and other interaction elements.

For an overview, see: Methods of Class CL_GUI_ALV_GRID [Page 66]

get_scroll_info_via_id

Use

Get the current scroll position in the grid control:

- The row index (field **`index`** of structure **`lvc_s_row`**) indicates which row is displayed first (vertical scrolling).
- The column name (field **`fieldname`** of structure **`lvc_s_col`**) indicates which scrollable column (the key fields are static) is displayed first on the left (horizontal scrolling).

Features

```
CALL METHOD <ref.var. to CL_GUI_ALV_GRID>->get_scroll_info_via_id

   IMPORTING
     ES_ROW_INFO   = <structure of type LVC_S_ROW>
     ES_COL_INFO   = <structure of type  LVC_S_COL>.
```

Parameter	Meaning
`ES_ROW_INFO`	Information on vertical scrolling
`ES_COL_INFO`	Information on horizontal scrolling

For an overview, see: Methods of Class CL_GUI_ALV_GRID [Page 66]

get_selected_cells

Use

Get selected cells in the cell selection mode. The ALV Grid Control returns the values, the column name and the row index of the relevant cell.

The ALV Grid Control returns only the indexes of cells that are **selected individually**. If an entire row or column is selected, the table returned remains empty.

Prerequisites

You must enable the 'cell selection' mode to allow users to select single cells (and multiple cells with the Ctrl key) (see also: Properties of the Grid Control [Page 159]). To do this, pass the layout structure [Page 157] before first display using method set_table_for_first_display [Page 100]).

Features

```
CALL METHOD <ref.var. to CL_GUI_ALV_GRID>->get_selected_cells
   IMPORTING
      ET_CELL   =   <internal table of type LVC_T_CELL>.
```

Parameter	Meaning
ET_CELL	Table with information on one row per cell

For an overview, see: Methods of Class CL_GUI_ALV_GRID [Page 66]

get_selected_cells_id

Use

Get the column and row index of the cells currently selected.

Features

CALL METHOD <ref. var. to **CL_GUI_ALV_GRID**>->**get_selected_cells_id**

```
    IMPORTING
       ET_CELLS   =   <internal table of type LVC_T_CENO>.
```

Parameter	Meaning
ET_CELLS	Table with row [Ext.] and column indexes (**COL_ID**).

For an overview, see Methods of Class CL_GUI_ALV_GRID [Page 66]

Activities

The COL_ID [Page 140] is assigned internally. In order to determine the field name of your output table, you get the field catalog using method get_frontend_fieldcatalog [Page 73]. In the field catalog, the **COL_ID** is uniquely assigned to the field name.

get_selected_columns

Use

Get the field names of selected columns.

If only rows or cells are selected, the control returns an empty table.

Features

```
CALL METHOD <ref.var. to CL_GUI_ALV_GRID>->get_selected_columns
   IMPORTING
      ET_INDEX_COLUMNS   =   <internal table of type LVC_T_COL>.
```

Parameter	**Meaning**
`ET_INDEX_COLUMNS`	Table with field names of the selected columns

For an overview, see: Methods of Class CL_GUI_ALV_GRID [Page 66]

get_selected_rows

Use

Get indexes of selected rows.

- The numbering of grid control rows starts with 1.
- If only cells or columns are selected, the ALV Grid Control returns an empty table.
- If the user has selected multiple rows, the indexes are generally sorted in ascending order in the table.

Prerequisites

To allow users to select multiple rows, you must set field **`sel_mode`** of the layout structure to **`'A'`**, **`'C'`** or **`'D'`** (see also: Properties of the Grid Control [Page 159]).

Features

```
CALL METHOD <ref.var. to CL_GUI_ALV_GRID>->get_selected_rows
   IMPORTING
      ET_INDEX_ROWS  =  <internal table of type LVC_T_ROW>.
```

Parameter	Meaning
`ET_INDEX_ROWS`	Table with indexes of the selected rows

For an overview, see: Methods of Class CL_GUI_ALV_GRID [Page 66]

get_sort_criteria

Use

Get the current sort criteria for sorted columns of the grid control.

Features

CALL METHOD <ref.var. to **CL_GUI_ALV_GRID**>->**get_sort_criteria**

```
    IMPORTING
       ET_SORT   =   <internal table of type LVC_T_SORT>.
```

Parameter	Meaning
ET_SORT	Each table row returns the field name and sort criteria of the sorted columns (see also: Fields of the Sort Table [Ext.]).

For an overview, see: Methods of Class CL_GUI_ALV_GRID [Page 66]

get_subtotals

Use

Returns the current subtotals of the ALV Grid Control. Having created totals for at least one column, users can calculate at most nine subtotals. The list is sorted by the values of one or several columns (called the subtotals column). If a value is changed in the subtotals column, the subtotal is displayed (this is also referred to as *control level change*).

Integration

Before you access the subtotals value, you use the method get_sort_criteria [Page 80] to get the sort table [Ext.]. One row of this table describes properties of a column in the output table.

- If the field **SUBTOT** is set for a column in this table, the column is a subtotals column.
- The field **SPOS** then indicates at which level (see below) the subtotal has been calculated.
- The field **FIELDNAME** contains the name of the subtotals column in the output table.

Based on this information, you specifically access the values of the tables passed (using reference variables **COLLECT01** to **COLLECT09**).

Features

```
CALL METHOD <ref. var. to CL_GUI_ALV_GRID>->get_subtotals
   IMPORTING
      EP_COLLECT00    =  <reference variable of type REF TO DATA>
      EP_COLLECT01    =  <reference variable of type REF TO DATA>
      EP_COLLECT02    =  <reference variable of type REF TO DATA>
      EP_COLLECT03    =  <reference variable of type REF TO DATA>
      EP_COLLECT04    =  <reference variable of type REF TO DATA>
      EP_COLLECT05    =  <reference variable of type REF TO DATA>
      EP_COLLECT06    =  <reference variable of type REF TO DATA>
      EP_COLLECT07    =  <reference variable of type REF TO DATA>
      EP_COLLECT08    =  <reference variable of type REF TO DATA>
      EP_COLLECT09    =  <reference variable of type REF TO DATA>
      ET_GROUPLEVELS  =  <internal table of type LVC_T_GRPL>.
```

In order to access the values of **EP_COLLECT00** to **EP_COLLECT09**, you use **ASSIGN** to dereference the relevant reference variable in a field symbol of your output table type (see below).

Parameters	Meaning
EP_COLLECT00	Points to the totals line. Since there is only one totals line which, in addition, has a unique totalling area, no further information is available for this table in table **ET_GROUPLEVELS**.

EP_COLLECT01 to **EP_COLLECT09**	Point to the subtotals line. For each subtotals level, there is one reference variable. Each of these variables points to an internal table that has the type of the output table. **EP_COLLECT01** points to the subtotal at the hightest level, while **EP_COLLECT02** points to the subtotal at the second highest level, and so on. The levels are derived from the sort priority (field **SPOS** in the sort table [Ext.]). The column by which the data was sorted first, is the highest subtotals level.
ET_GROUPLEVELS	Manages all indexes for the individual control levels. The fields of the table have the following meaning: • **INDEX_FROM**, **INDEX_TO**: Rows of the output table for which the subtotal was created • **LEVEL:** Subtotals level (see above) • **COUNTER**: Number of rows for which the subtotal was created • **COMPRESS**: For this subtotals line, the user has hidden the associated rows. • **COLLECT**: Indicates the subtotals table (**01-09**) in which the values are stored

For an overview, see Methods of Class CL_GUI_ALV_GRID [Page 66]

Activities

Access the subtotals tables (in our example, only the totals and the first subtotals level) using field symbols:

```
data: total type ref to data,
      subtotal1 type ref to data.

field-symbols <total> like gt_sflight.
field-symbols <subtotal1> like gt_sflight.

call method grid1->get_subtotals
   importing
      ep_collect00 = total
      ep_collect01 = subtotal1.

assign total->* to <total>.
assign subtotal1->* to <subtotal1>.
```

get_variant

Use

Get the current layout.

☐

Layout in this context refers to the layout for the filter, sort and column settings, and **not** to the layout structure [Page 157].

Prerequisites

The current layout has been saved. So that the user can save layouts, at least parameters I_SAVE and IS_VARIANT must contain a value when set_table_for_first_display [Page 100] is called.

Features

```
CALL METHOD <ref. var. to CL_GUI_ALV_GRID>->get_variant
   IMPORTING
      ES_VARIANT  =  <structure of type  DISVARIANT>.
```

Parameter	Meaning
`ES_VARIANT`	Structure with information on the layout currently used

For an overview, see: Methods of Class CL_GUI_ALV_GRID [Page 66]

offline

Use

You use this method to determine if the ALV Grid Control runs without frontend handling.

The ALV Grid Control runs without frontend handling in batch mode (**sy-batch** has the value '**X**'), in print mode, in combination with CATT and during Web Reporting.

In these cases, you should not create frontend objects (see below).

Features

```
CALL METHOD CL_GUI_ALV_GRID=>offline
   RECEIVING
      E_OFFLINE  =  <variable of type INT4>.
```

Parameter	Meaning
E_OFFLINE	If this parameter is set to 1, no frontend handling takes place.

For an overview, see Methods of Class CL_GUI_ALV_GRID [Page 66]

Activities

You can use this method, for example, to create the control container for the ALV Grid Control only for online processing:

```
DATA: g_dock TYPE REF TO cl_gui_docking_container,
      g_grid TYPE REF TO cl_gui_alv_grid.

IF cl_gui_alv_grid=>offline( ) is initial.
    CREATE OBJECT g_dock.
ENDIF.

CREATE OBJECT g_grid
    EXPORTING I_PARENT = g_dock.
```

The constructor of the ALV Grid Control checks if the instance runs offline. If this is the case, the constructor does not create frontend objects. You then work only with the instance at the backend.

refresh_table_display

Use

You use this method to refresh the output table in the grid control. This is necessary if you change the display of the data by means of methods (for example, using set_frontend_layout [Page 92]), or if you want to display new data selected in the same ALV Grid Control.

Demo program **BCALV_GRID_03** in development class **SLIS** uses this method to refresh the output table after a new selection.

Features

CALL METHOD <ref.var. to **CL_GUI_ALV_GRID>->refresh_table_display**

```
    EXPORTING
      IS_STABLE      = <structure of type LVC_S_STBL>
      I_SOFT_REFRESH  = <variable of type CHAR01>.
```

Parameter	Meaning
`IS_STABLE`	If the row or col field of this structure is set, the position of the scroll bar for the rows or columns remains stable.
`I_SOFT_REFRESH`	This parameter is used only in exceptional cases. If you set this parameter, any totals created, any sort order defined and any filters set for the data displayed remain unchanged when the grid control is refreshed. This makes sense, for example, if you have not modified the data of the data table and want to refresh the grid control only with regard to layout or field catalog changes.

For an overview, see: Methods of Class CL_GUI_ALV_GRID [Page 66]

register_delayed_event

Use

You use this method to register either one of the delayed events (you cannot register both events at the same time).

You are **strongly** recommended to read Using the ALV Grid Control in WANs [Page 42] before using these events.

Features

CALL METHOD <ref. var. to **CL_GUI_ALV_GRID**>**->register_delayed_event**

```
    EXPORTING
        I_EVENT_ID   =   <var. of type I>.
```

Parameter	Meaning
I_EVENT_ID	Event ID: • **cl_gui_alv_grid=>mc_evt_delayed_move_curr_cel** for event delayed_callback [Page 111] • **cl_gui_alv_grid=>mc_evt_delayed_change_select** for event delayed_changed_sel_callback [Page 112]

For an overview, see: Methods of Class CL_GUI_ALV_GRID [Page 66]

set_3d_border

Use

You can use this method to display the border of the ALV Grid Control in 3D format.

Features

```
CALL METHOD <ref.var. to CL_GUI_ALV_GRID>->set_3d_border
    EXPORTING
        BORDER   =   <var. of type  I>.
```

Parameter	Meaning
BORDER	Display border in 3D format (1=yes, 0=no)

Go back to: Methods of Class CL_GUI_ALV_GRID [Page 66]

save_variant_dark

Use

This method assigns a name for the layout [Page 16] to be saved. If the user saves the current layout, the ALV Grid Control suppresses the dialog box for saving a layout and issues a simple confirmation message instead.

Prerequisites

The user must be authorized to save layouts. For information on this, see Saving Layouts [Page 17].

Features

```
CALL METHOD <ref. var. to CL_GUI_ALV_GRID>->save_variant_dark
    IMPORTING
        IS_VARIANT   =   <structure of type DISVARIANT>.
```

Parameter	Meaning
`IS_VARIANT`	Structure for identifying a layout. As a minimum, you must fill the fields **REPORT** (sy-repid) and **VARIANT** (technical name of the layout).

For an overview, see Methods of Class CL_GUI_ALV_GRID [Page 66]

set_current_cell_via_id

Use

Set the cursor on a specific cell in the grid control. The grid control scrolls to the position specified if the cell is not visible.

Features

```
CALL METHOD <ref.var. to CL_GUI_ALV_GRID>->set_current_cell_via_id
    EXPORTING
       IS_ROW_ID     =   <structure of type LVC_S_ROW>
       IS_COLUMN_ID =   <structure of type LVC_S_COL>.
```

Parameter	**Meaning**
`IS_ROW_ID`	Structure with line index
`IS_COLUMN_ID`	Structure with column field names

For an overview, see: Methods of Class CL_GUI_ALV_GRID [Page 66]

set_filter_criteria

Use

Set current filter settings. A row of the table describes the selection conditions for column entries that are not to be displayed.

You should never manually set up the internal table with the filter settings. Use this method only to set filter criteria that you got using get_filter_criteria [Page 72] or a layout.

Features

```
CALL METHOD <ref. var. to CL_GUI_ALV_GRID>->set_filter_criteria
   EXPORTING
      IT_FILTER  =  <internal table of type LVC_T_FILT>.
```

Parameter	Meaning
`IT_FILTER`	Table with filter settings

For an overview, see: Methods of Class CL_GUI_ALV_GRID [Page 66]

set_frontend_fieldcatalog

Use

Set the field catalog for an ALV Grid Control instance.

After setting the field catalog, you refresh the list display using method refresh_table_display [Page 85]. If the field catalog refers to an output table that you display in an ALV Grid Control instance with another table structure, you must pass the field catalog using method set_table_for_first_display [Page 100].

See also: The Field Catalog [Page 129] **and** get_frontend_fieldcatalog [Page 73]**.**

Prerequisites

You have either got the current field catalog using method get_frontend_fieldcatalog [Page 73] or you have set up the field catalog semi-automatically [Page 135] before.

Features

```
CALL METHOD <ref.var. to CL_GUI_ALV_GRID>->set_frontend_fieldcatalog
   EXPORTING
      IT_FIELDCATALOG   =   <internal table of type  LVC_T_FCAT>.
```

Parameter	Meaning
`IT_FIELDCATALOG`	Description of the column attributes of the output table

For an overview, see: Methods of Class CL_GUI_ALV_GRID [Page 66]

Restrictions

In this method, the ALV Grid Control does not access the Data Dictionary. Consequently, it is not possible to **subsequently** create a Dictionary reference, for example, to automatically copy field labels as column texts that are stored in the Dictionary.

set_frontend_layout

Use

Set a layout structure at the frontend that you have got and modified using method get_frontend_layout [Page 74].

After setting the layout structure, you must refresh the list display using method refresh_table_display [Page 85].

In the layout structure you determine the properties of the grid control (see also The Layout Structure [Page 157]).

Using parameter **`IS_LAYOUT`** when method set_table_for_first_display [Page 100] is called, you set initial properties of the grid control.

Features

`CALL METHOD` <ref.var. to **`CL_GUI_ALV_GRID`**>**`->set_frontend_layout`**

```
   EXPORTING
      IS_LAYOUT   =   <structure of type LVC_S_LAYO>.
```

Parameter	**Meaning**
`IS_LAYOUT`	Structure with fields for display options, graphical grid control properties, totals options, exceptions, colors, drag and drop and other interaction elements.

For an overview, see: Methods of Class CL_GUI_ALV_GRID [Page 66]

set_scroll_info_via_id

Use

Set the scroll position in the grid control:

- The row index (field **`index`** of structure **`lvc_s_row`**) indicates which row is displayed first (vertical scrolling).
- The column name (field **`fieldname`** of structure **`lvc_s_col`**) indicates which scrollable column (the key fields are static) is displayed first on the left (horizontal scrolling).

Features

```
CALL METHOD <ref.var. to CL_GUI_ALV_GRID>->set_scroll_info_via_id
   EXPORTING
      IS_ROW_INFO    =   <structure of type  LVC_S_ROW>
      IS_COLUMN_INFO =   <structure of type  LVC_S_COL>.
```

Parameter	Meaning
`IS_ROW_INFO`	Information on vertical scrolling
`IS_COLUMN_INFO`	Information on horizontal scrolling

For an overview, see: Methods of Class CL_GUI_ALV_GRID [Page 66]

set_graphics_container

Use

The toolbar of the ALV Grid Control includes a function for displaying graphics that can be used to display the table data in a diagram using the Graphical Framework (GFW). In the default setting, this graphic is integrated into a dialog box container control. If you want to use a different container control, pass the associated reference variable using this method. The ALV Grid Control then displays the graphic in this container.

Integration

The SAP Container [Ext.] documentation describes all container types available.

Features

```
CALL METHOD <ref. var. to CL_GUI_ALV_GRID>->set_graphics_container

 EXPORTING
  I_GRAPHICS_CONTAINER   =   <reference variable to  CL_GUI_CONTAINER>.
```

Parameter	Meaning
`I_GRAPHICS_CONTAINER`	Reference variable to a container control in which the diagram with the output table data is to be displayed.

For an overview, see Methods of Class CL_GUI_ALV_GRID [Page 66]

set_selected_cells

Use

Select cells in the grid control. All other selections are removed before the ALV Grid Control selects the cells passed with the call.

Features

```
CALL METHOD <ref.var. to CL_GUI_ALV_GRID>->set_selected_cells
       EXPORTING
          IT_CELLS          = <Table of type LVC_T_CELL>.
```

Parameter	Meaning
`IT_CELLS`	Table with the field names of the cells to be selected. The row type of the table is **`LVC_S_COL`**; this structure contains one structure of type **`LVC_S_COL`** and one structure of type **`LVC_S_ROW`** that you use to determine the column and the row of the cell (see set_selected_rows [Page 98] and set_selected_columns [Page 97]).

For an overview, see: Methods of Class CL_GUI_ALV_GRID [Page 66]

set_selected_cells_id

Use

Select cells using row and column indexes.

Features

```
CALL METHOD <ref. var. to CL_GUI_ALV_GRID>->set_selected_cells_id
    IMPORTING
        IT_CELLS   =   <internal table of type LVC_T_CENO>.
```

Parameter	Meaning
`IT_CELLS`	Table with row [Ext.] and column indexes (`COL_ID`).

For an overview, see Methods of Class CL_GUI_ALV_GRID [Page 66]

Activities

The COL_ID [Page 140] is assigned internally. To determine the correct **`COL_ID`** for a field name in your output table, you get the field catalog using method get_frontend_fieldcatalog [Page 73]. In the field catalog, the **`COL_ID`** is uniquely assigned to the field name.

set_selected_columns

Use

Select columns in the grid control. All other selections are removed before the ALV Grid Control selects the columns passed with the call.

Features

```
CALL METHOD <ref.var. to CL_GUI_ALV_GRID>->set_selected_columns
      EXPORTING
         IT_COL_TABLE           = <table of type LVC_T_COL>.
```

Parameter	Meaning
`IT_COL_TABLE`	Table with the field names of the columns to be selected (row type: **`LVC_S_COL`**; field to be filled: **`FIELDNAME`**).

For an overview, see: Methods of Class CL_GUI_ALV_GRID [Page 66]

set_selected_rows

Use

Select rows in the grid control using the table index.

The numbering of grid control rows starts with 1.

Features

```
CALL METHOD <ref.var. to CL_GUI_ALV_GRID>->set_selected_rows
   EXPORTING
      IT_INDEX_ROWS   =   <internal table of type LVC_T_ROW>.
```

Parameter	Meaning
`IT_INDEX_ROWS`	Table with indexes of the rows to be selected

For an overview, see: Methods of Class CL_GUI_ALV_GRID [Page 66]

set_sort_criteria

Use

Set sort criteria for columns of the grid control. Settings only take effect if you refresh the table using method refresh_table_display [Page 85].

Features

CALL METHOD <ref.var. to **CL_GUI_ALV_GRID**>->**set_sort_criteria**

```
    EXPORTING
        IT_SORT   =   <internal table of type LVC_T_SORT>.
```

Parameter	Meaning
IT_SORT	For each table row, the sort criteria for a column are defined (see also: Fields of the Sort Table [Ext.]).

For an overview, see: Methods of Class CL_GUI_ALV_GRID [Page 66]

set_table_for_first_display

Use

Display an output table in the ALV Grid Control instance. In the call sequence, you must specify either a reference structure of the Data Dictionary or a suitable field catalog [Page 129]. Before execution, you can use optional parameters to load a layout, sort the table by fields, set a filter for columns and define properties of the grid control.

If you want to refresh the data displayed in the output table, use method refresh_table_display [Page 85]. Method **set_table_for_first_display** must only be called a second time if the structure of the output table changes.

Report **BCALV_GRID_DEMO** of development class **SLIS** illustrates the simplest way to call this method.

Prerequisites

The output table must either be defined globally or be a public attribute of a class.

Features

CALL METHOD **<ref. var. to** CL_GUI_ALV_GRID**>**->set_table_for_first_display

```
    EXPORTING
        I_BUFFER_ACTIVE      = <any type (ANY)>
        I_STRUCTURE_NAME     = <string of type DD02L-TABNAME>
        IS_VARIANT           = <structure of type DISVARIANT>
        I_SAVE               = <var. of  type CHAR01>
        I_DEFAULT            = <var. of  type CHAR01>
        IS_LAYOUT            = <structure of type LVC_S_LAYO>
        IS_PRINT             = <structure of type LVC_S_PRNT>
        IT_SPECIAL_GROUPS    = <internal table of type LVC_T_SGRP>
        IT_TOOLBAR_EXCLUDING = <internal table of type UI_FUNCTIONS>
        IT_HYPERLINK         = <internal table of type LVC_T_HYPE>
        IT_ALV_GRAPHICS      = <internal table of type DTC_T_TC>
    CHANGING
        IT_OUTTAB            = <internal table>
        IT_FIELDCATALOG      = <internal table of type LVC_T_FCAT>
        IT_SORT              = <internal table of type LVC_T_SORT>
        IT_FILTER            = <internal table of type LVC_T_FILT>
```

Parameter	Meaning
I_BUFFER_ACTIVE	Flag to be set by the application if the method call is static. This means the method is always called with the same field catalog. In this case, the field catalog can be held in a special buffer. This accelerates the display of small lists, in particular.

I_STRUCTURE_NAME	Name of the DDIC structure (for example, 'SFLIGHT') for the data in the output table. If you specify this parameter, the field catalog is generated automatically.
IS_VARIANT	Determines the layout to be used for displaying the output table. If you use this parameter, you must at least fill field **REPORT** of the structure of type **DISVARIANT**.
I_SAVE	Determines the options available to the user for saving a layout: • 'X': global saving only • 'U': user-specific saving only • 'A': corresponds to 'X' and 'U' • SPACE: no saving
I_DEFAULT	This parameter determines if the user is allowed to define default layouts: • 'X': Default layouts allowed (default setting) • SPACE: Default layouts not allowed If default layouts are allowed and if such a layout exists and no other layout is specified in **IS_VARIANT**, the default layout is automatically loaded when this method is called.
IS_LAYOUT	Determines properties of the grid control. The layout structure has **nothing** to do with the layout for saving filter, sort, and column properties.
IS_PRINT	Parameter for printing on the backend
IT_SPECIAL_GROUPS	If in the field catalog the columns were grouped together with field **SP_GROUP**, you must pass a table with texts for these groups. On the current layout window, it is then possible to use a list box to restrict column selection to one of these groups.
IT_TOOLBAR_EXCLUDING	This table contains function codes of the toolbar that you want to hide for the lifetime of the ALV Grid Control. The function codes are constant attributes and are prefixed with **MC_FC_**.
IT_HYPERLINK	This table assigns a hyperlink address (field **HREF** of **LVC_S_HYPE**) to each handle (field **HANDLE** of **LVC_S_HYPE**). Using this handle, you can then include hyperlinks in the grid [Page 36].
IT_ALV_GRAPHICS	Settings for displaying the ALV list as a diagram (for example, axis labels)
IT_OUTTAB	Output table with the data to be displayed
IT_FIELDCATALOG	Determines the structure of the output table and the format of the data to be displayed
IT_SORT	Table with sort properties for columns that are to be sorted initially

set_table_for_first_display

`IT_FILTER`	Table with filter properties for columns for which a filter is to be set initially

For an overview, see: Methods of Class CL_GUI_ALV_GRID [Page 66]

Activities

☐

During this call, the ALV Grid Control generates the field catalog for the output table automatically [Page 131] before display, using a DDIC structure:

```
DATA: GRID1 TYPE REF TO CL_GUI_ALV_GRID,
      GT_SFLIGHT TYPE TABLE OF SFLIGHT.
```

<Instantiation of **GRID1** and integration into screen>

```
CALL METHOD GRID1->SET_TABLE_FOR_FIRST_DISPLAY
         EXPORTING I_STRUCTURE_NAME = 'SFLIGHT'
         CHANGING  IT_OUTTAB        = GT_SFLIGHT.
```

☐

Explicitly pass the field catalog:

```
DATA: GRID1 TYPE REF TO CL_GUI_ALV_GRID,
      GT_SFLIGHT TYPE TABLE OF SFLIGHT,
      GT_FIELDCAT TYPE LVC_T_FCAT.
```

<Manual [Page 133] or semi-automatic [Page 135] generation of the field catalog >
<Instantiation of **GRID1** and integration into screen>

```
CALL METHOD GRID1->SET_TABLE_FOR_FIRST_DISPLAY
     CHANGING  IT_FIELDCATALOG = GT_FIELDCAT
               IT_OUTTAB       = GT_SFLIGHT.
```

set_toolbar_interactive

Use

This method triggers the toolbar [Page 127] event. If you add self-defined functions to the toolbar, you use this method to recreate the toolbar (see also: Defining GUI Elements in the Toolbar [Page 21]).

Features

CALL METHOD <ref.var. to **CL_GUI_ALV_GRID**>**->set_toolbar_interactive.**

For an overview, see: Methods of Class CL_GUI_ALV_GRID [Page 66]

set_user_command

Use

You can use this method to replace standard functions of the ALV Grid Control with self-defined functions. To do this, you must get the current function code in event before_user_command [Page 108] and change it using **set_user_command**.

See also: Changing Standard Functions [Page 20]

Features

```
CALL METHOD <ref.var. to CL_GUI_ALV_GRID>->set_user_command
   EXPORTING
      I_UCOMM   =   <var. of type SY-UCOMM>.
```

Parameter	Meaning
I_UCOMM	Function code (function codes are defined as constant attributes of the class (and are prefixed with **MC_FC_**))

For an overview, see: Methods of Class CL_GUI_ALV_GRID [Page 66]

Events of Class CL_GUI_ALV_GRID

The section Working With the ALV Grid Control [Page 11] describes the special points and issues you must consider when you handle events of the ALV Grid Control.

User-defined Text Output

Event	Application
print_end_of_list [Page 121]	Define output text to be printed at the end of the entire list
print_top_of_list [Page 123]	Define output text to be printed at the beginning of the entire list
print_end_of_page [Page 122]	Define output text to be printed at the end of each page
print_top_of_page [Page 124]	Define output text to be printed at the beginning of each page
subtotal_text [Page 125]	Define self-defined subtotals texts

Mouse-controlled Actions in the Grid Control

Event	Application
button_click [Page 109]	Query a click on a pushbutton in the ALV Grid Control
double_click [Page 113]	Query a double-click on a cell of the ALV Grid control
hotspot_click [Page 114]	Query a hotspot click on columns defined for this purpose in advance
onDrag [Page 116]	Collect information when elements of the ALV Grid Control are dragged
onDrop [Page 117]	Process information when elements of the ALV Grid Control are dropped
onDropComplete [Page 118]	Perform final actions after successful Drag&Drop
onDropGetFlavor [Page 119]	Distinguish between options for Drag&Drop behavior

Processing of Self-defined and Standard Functions

Event	Application
before_user_command [Page 108]	Query self-defined and standard function codes
user_command [Page 128]	Query self-defined function codes
after_user_command [Page 107]	Query self-defined and standard function codes

Definition of Self-defined Functions

Event	Application

toolbar [Page 127]	Change, delete or add GUI elements in the toolbar
menu_button [Page 115]	Define menus for menu buttons in the toolbar
context_menu_request [Page 110]	Change context menu
onf1 [Page 120]	Define self-defined F1 help

after_user_command

Use

The ALV Grid Control triggers this event after the user_command [Page 128] event if the user has chosen a standard or a self-defined function. You can use this event to perform final actions after the relevant function has been completed.

Standard functions are already completed in this event.

Event parameter	Meaning
E_UCOMM ***TYPE SY-UCOMM***	Standard or self-defined function code (standard function codes are constant attributes and prefixed with **MC_FC_**).

For an overview, see: Events of Class CL_GUI_ALV_GRID [Page 105]

before_user_command

Use

The ALV Grid Control triggers this event before the user_command [Page 128] event if the user has chosen a standard or a self-defined function. You can use this event to perform introductory actions before the relevant function is executed.

See also: Changing Standard Functions [Page 20]

Event parameter	Meaning
`E_UCOMM`	Standard or self-defined function code (standard function codes are constant attributes and prefixed with `MC_FC_`).

For an overview, see: Events of Class CL_GUI_ALV_GRID [Page 105]

button_click

Use

After the user has clicked a pushbutton in the ALV Grid Control, you handle this event in the event handler method.

Prerequisites

You have displayed one or more cells as pushbuttons in the ALV Grid Control (see Displaying Cells as Pushbuttons [Page 34]).

Event parameters	Meaning
ES_COL_ID TYPE *LVC_S_COL*	Structure with field **FIELDNAME** for identifying the column
ES_ROW_NO *TYPE* *LVC_S_ROID*	Structure for identifying the row [Ext.]

For an overview, see Events of Class CL_GUI_ALV_GRID [Page 105]

context_menu_request

Use

In this event, you can add self-defined functions to the standard context menu of the ALV Grid Control and hide and disable existing functions.

See also: Defining a Context Menu [Page 22]

Event parameter	Meaning
E_OBJECT ***TYPE REF TO CL_CTMENU***	Reference variable to the standard context menu of the ALV Grid Control

For an overview, see: Events of Class CL_GUI_ALV_GRID [Page 105]

delayed_callback

Use

This event is triggered if the user clicks a cell other than that currently selected. The ALV Grid Control triggers this event with a short delay of 1,5 seconds.

You are **strongly** recommended to read Using the ALV Grid Control in WANs [Page 42] before using this event.

The ALV Grid Control triggers this event also for method **`set_table_for_first_display`** and for standard function `Sum`. For standard function `Find`, the event is only triggered if a cell found is not identical to the cell selected before.

For an overview, see: Events of Class CL_GUI_ALV_GRID [Page 105]

delayed_changed_sel_callback

Use

The event is triggered if the user selects a row or column that has not yet been selected. The ALV Grid Control triggers this event with a short delay of 1,5 seconds.

You are **strongly** recommended to read Using the ALV Grid Control in WANs [Page 42] before using this event.

For an overview, see: Events of Class CL_GUI_ALV_GRID [Page 105]

double_click

Use

The ALV Grid Control triggers this event if a user double-clicks on cells of the grid control.

In demo report **`BCALV_GRID_03`** of development class **`SLIS`**, double-clicking takes you to a details list.

Event parameter	Meaning
`E_ROW` ***`TYPE REF TO LVC_S_ROW`***	Structure with row index
`E_COLUMN` ***`TYPE REF TO LVC_S_COL`***	Structure with column name

For an overview, see: Events of Class CL_GUI_ALV_GRID [Page 105]

hotspot_click

Use

In this event, you respond to a user's mouse-click on a column that is marked as a hotspot (the shape of the mouse pointer changes accordingly).

Prerequisites

To enable users to select columns through a hotspot click, you must designate the column as a hotspot column. To do this, you use:

- Field **`hotspot`** of the field catalog for single columns (see also: Output Options of a Column [Page 145])
- Field **`keyhot`** of the layout structure for all key fields of an output table (see also: Interaction Control [Page 164])

Event parameter	Meaning
`E_ROW_ID` *`TYPE REF TO LVC_S_ROW`*	Structure with row index
`E_COLUMN_ID` *`TYPE REF TO LVC_S_COL`*	Structure with column name

For an overview, see: Events of Class CL_GUI_ALV_GRID [Page 105]

menu_button

Use

In this event, you query function codes for menus that you have defined in the toolbar [Page 127] event. For each function code, you define a menu in the same way as for the standard context menu of the ALV Grid Control.

See also: Defining a Menu in the Toolbar [Page 23]

The event is triggered each time a user clicks a menu button in the toolbar.

Event parameter	Meaning
`E_OBJECT` *`TYPE REF TO CL_CTMENU`*	Reference to the menu of the toolbar that the user has chosen. You use methods of this object to build this menu.
`E_UCOMM` *`Type SY-UCOMM`*	Function code of a menu that you have defined in the toolbar event. You use this parameter to distinguish between the menus defined in the toolbar.

For an overview, see: Events of Class CL_GUI_ALV_GRID [Page 105]

onDrag

The meaning of this event is explained in Drag & Drop Events [Ext.] of the Control Framework documentation. For information on passing the handle of the Drag&Drop behavior, see Drag & Drop With the ALV Grid Control [Page 24].

This event can be named differently for other controls.

Event parameter	Meaning
E_ROW *Type LVC_S_ROW*	Structure with the index of the row dragged
E_COLUMN *Type LVC_S_COL*	Structure with the index of the column dragged
E_DRAGDROPOBJ *Type Ref To CL_DRAGDROPOBJ ECT*	Drag&Drop object that can be used to pass an application-specific data object. The attributes also contain information on the user action (copy or move) and the flavor associated with that action.

onDrop

The meaning of this event is explained in Drag & Drop Events [Ext.] of the Control Framework documentation. For information on passing the handle of the Drag&Drop behavior, see Drag & Drop With the ALV Grid Control [Page 24].

This event can be named differently for other controls.

Event parameter	Meaning
E_ROW *Type LVC_S_ROW*	Structure with the index of the row dropped
E_COLUMN *Type LVC_S_COL*	Structure with the index of the column dropped
E_DRAGDROPOBJ *Type Ref To CL_DRAGDROPOBJECT*	Drag&Drop object that can be used to pass an application-specific data object. The attributes also contain information on the user action (copy or move) and the flavor associated with that action.

onDropComplete

The meaning of this event is explained in Drag & Drop Events [Ext.] of the Control Framework documentation. For information on passing the handle of the Drag&Drop behavior, see: Drag & Drop With the ALV Grid Control [Page 24].

This event can be named differently for other controls.

Event parameter	Meaning
E_ROW *Type LVC_S_ROW*	Structure with the index of the row dragged
E_COLUMN *Type LVC_S_COL*	Structure with the index of the column dragged
E_DRAGDROPOBJ *Type Ref To CL_DRAGDROPOBJECT*	Drag&Drop object that can be used to pass an application-specific data object. The attributes also contain information on the user action (copy or move) and the flavor associated with that action.

onDropGetFlavor

The meaning of this event is explained in Drag & Drop Events [Ext.] of the Control Framework documentation. For information on passing the handle of the Drag&Drop behavior, see: Drag & Drop With the ALV Grid Control [Page 24].

This event can be named differently for other controls.

Event parameter	Meaning
E_ROW ***Type LVC_S_ROW***	Structure with the index of the row dragged
E_COLUMN ***Type LVC_S_COL***	Structure with the index of the column dragged
E_DRAGDROPOBJ ***Type Ref To CL_DRAGDROPOBJECT***	Drag&Drop object that can be used to pass an application-specific data object. The attributes also contain information on the user action (copy or move) and the flavor associated with that action.
E_FLAVORS ***Type CNDD_FLAVORS***	Possible flavors

onf1

Use

If the user requests the F1 help for a cell, the ALV Grid Control tries to call the documentation for the associated data element. If no documentation exists (for example, because the field has no reference to the Data Dictionary), you can use this event to display a self-defined F1 help. You also have the option of overriding the default F1 help.

To inform the ALV Grid Control that you use your own F1 help, set the attribute **`er_event_data->m_event_handled`** at the end of your event handling.

Event parameters	**Meaning**
`E_FIELDNAME` ***`TYPE LVC_FNAME`***	Name of the output table field for which a self-defined F1 help is to be implemented
`ES_ROW_NO` ***`TYPE LVC_S_ROID`***	Structure for identifying the row [Ext.]
`ER_EVENT_DATA` ***`TYPE REF TO CL_ALV_EVENT_DATA`***	Set the attribute **`M_EVENT_HANDLED`** of this object if you have handled the event. The ALV Grid Control then suppresses the default F1 help.

For an overview, see Events of Class CL_GUI_ALV_GRID [Page 105]

print_end_of_list

Use

You use this event to output text with the **WRITE** statement during print output. The text is inserted at the end of the entire list and is displayed in the print preview. In this context, the ALV Grid Control goes to the list output of the classic ALV.

Demo program **BCALV_GRID_01** in development class **SLIS** illustrates how **print_end_of_list** is used.

For an overview, see: Events of Class CL_GUI_ALV_GRID [Page 105]

print_end_of_page

Use

You use this event to output text with the **`WRITE`** statement during print output. The text is inserted at the end of each page. During print output, the ALV Grid Control goes to the list output of the classic ALV.

In the print preview (classic ALV), the text for this event is **not** displayed.

Demo program **`BCALV_GRID_01`** in development class **`SLIS`** illustrates how **`print_end_of_page`** is used.

Prerequisites

To allow output at the end of each page, you must reserve several lines for these pages. To do this, use field **`reservelns`** of a structure of type **`lvc_s_prnt`** and pass this structure with method set_table_for_first_display [Page 100].

For an overview, see: Events of Class CL_GUI_ALV_GRID [Page 105]

print_top_of_list

Use

You use this event to output text with the **WRITE** statement during print output. The event is triggered once before the print_top_of_page [Page 124] event. The text is inserted at the beginning of the entire list and is displayed in the print preview. In this context, the ALV Grid Control goes to the list output of the classic ALV.

At the beginning of the print output, the ALV Grid Control first outputs the title of the list, provided you have determined a title with field **grid_title** of the layout structure.

Demo program **BCALV_GRID_01** in development class **SLIS** illustrates how **print_top_of_list** is used.

For an overview, see: Events of Class CL_GUI_ALV_GRID [Page 105]

print_top_of_page

Use

You use this event to output text with the **`WRITE`** statement during print output. The text is inserted at the beginning of each page and is displayed in the print preview. In this context, the ALV Grid Control goes to the list output of the classic ALV.

Demo program **`BCALV_GRID_01`** in development class **`SLIS`** illustrates how **`print_top_of_page`** is used.

For an overview, see: Events of Class CL_GUI_ALV_GRID [Page 105]

subtotal_text

Use

Output text for subtotals in the grid control if the subtotal criterion (column used by the user for calculating the subtotal) is hidden. In the default setting, the ALV Grid Control outputs the column header of the subtotal criterion and the value to which the subtotal calculated refers.

Event parameters	Meaning
ES_SUBTOTTXT_INFO *Type LVC_S_STXT*	Structure with information on the subtotal criterion
EP_SUBTOT_LINE *Type Ref To DATA*	Reference variable to the inserted subtotal line in the grid control. Columns for which no total was calculated remain set to their initial value.
E_EVENT_DATA *Type Ref To CL_ALV_EVENT_DATA*	Attribute **M_DATA** of this object is a reference to the subtotal text.

For an overview, see: Events of Class CL_GUI_ALV_GRID [Page 105]

Example

You display a table with data of structure **SFLIGHT** in an ALV Grid Control. We now change the pre-set subtotal text for subtotal criterion `plane type` of table **SFLIGHT**:

1. Define and implement an event handler method for event **subtotal_text**. Register this event with **SET HANDLER**.
2. Call a subroutine within this method and pass the event parameters to that subroutine.
3. Event parameters **ep_subtot_line** and **e_event_data** contain reference variables to generic data type **DATA**. This is why you must use field symbols in the subroutine:

```
FORM method_subtotal_text USING es_subtottxt_info TYPE lvc_s_stxt
                                ep_subtot_line TYPE REF TO data
                                e_event_data TYPE REF TO
                                cl_alv_event_data.

  DATA ls_sflight LIKE sflight.
  FIELD-SYMBOLS: <fs1> STRUCTURE sflight DEFAULT ls_sflight,
                 <fs2>.

  IF es_subtottxt_info-criteria = 'PLANETYPE'.
    ASSIGN ep_subtot_line->* TO <fs1>.
    ASSIGN  e_event_data->m_data->* TO <fs2>.
    CONCATENATE es_subtottxt_info-keyword ': '
                <fs1>-planetype INTO <fs2>.
  ENDIF.
```

```
ENDFORM.                                        " METHOD_SUBTOTAL_TEXT
```

Result

Check your result as follows:

1. Calculate the total for a column.
2. Calculate subtotals for column `plane type`.
3. Hide column `plane type`. To do this, use either the standard context menu or a new layout.

The new text defined is displayed at the beginning of a subtotal line.

toolbar

Use

This event is triggered by the ALV each time the toolbar of the control needs to be regenerated. To add self-defined functions to the toolbar, you trigger the event using method set_toolbar_interactive [Page 103] and write an event handler method (**see also:** Defining GUI Elements in the Toolbar [Page 21]).

You can hide the entire toolbar using field **no_toolbar** of the layout structure.

In demo report **BCALV_GRID_05** of development class **SLIS**, a new pushbutton is added to the toolbar.

Event parameter	Meaning
E_OBJECT ***Type Ref To CL_ALV_EVENT_TOOLBAR_SET***	The object contains only one attribute with a table for the functions of the toolbar.
E_INTERACTIVE ***Type CHAR01***	If this flag is set, you triggered the event using method **set_toolbar_interactive**. If this flag is not set, the event was triggered by the ALV Grid Control.

For an overview, see: Events of Class CL_GUI_ALV_GRID [Page 105]

user_command

Use

The ALV Grid Control triggers this event only **for self-defined functions** chosen by the user. Query your function code and call your function in this event.

You can query the function codes of your self-defined functions also in the before_user_command [Page 108] and after_user_command [Page 107] events. These events are triggered by the control before or after the `user_command` event.

Event parameter	**Meaning**
`E_UCOMM` *`TYPE SY-UCOMM`*	Function code for self-defined function

For an overview, see: Events of Class CL_GUI_ALV_GRID [Page 105]

The Field Catalog

Definition

The field cataog is a table of type **LVC_T_FCAT** that contains information on the fields to be displayed. The ALV uses this table to recognize the type of a field, for example.

Use

You can use fields of the catalog to determine the number format and column properties of the list to be displayed.

In some exceptional cases, it is not necessary to pass the field catalog (see also: Generating the Field Catalog Automatically [Page 131]).

Structure

The field catalog contains more than 60 fields, some of which are only used internally. Those fields that are relevant to application developers are described in Fields of the Field Catalog [Page 136].

Integration

Generally, you are recommended to fill the fields of the field catalog before the list is displayed for the first time and pass them in method set_table_for_first_display [Page 100]. To adjust a field catalog generated by the ALV to your special requirements before list output, you use function module **LVC_FIELDCATALOG_MERGE**.

Methods get_frontend_fieldcatalog [Page 73] and set_frontend_fieldcatalog [Page 91] allow you to change the field catalog after list output.

Report **BCALV_GRID_03** in development class **SLIS** uses this function module to hide columns before the list is displayed.

Generating the Field Catalog

Basically, you need a field catalog for each list that is displayed using the ALV Grid Control. You have several options of generating a field catalog:

- Automatically through a Data Dictionary structure
- Manually in your ABAP program
- Semi-automatically by combining the above two procedures.

Generating the Field Catalog Automatically

Purpose

If the structure of your output table corresponds to a structure stored in the Data Dictionary (DDIC), the ALV Grid Control can use this information to generate the field catalog automatically. In this case, all fields of this DDIC structure are displayed in the list.

Process Flow

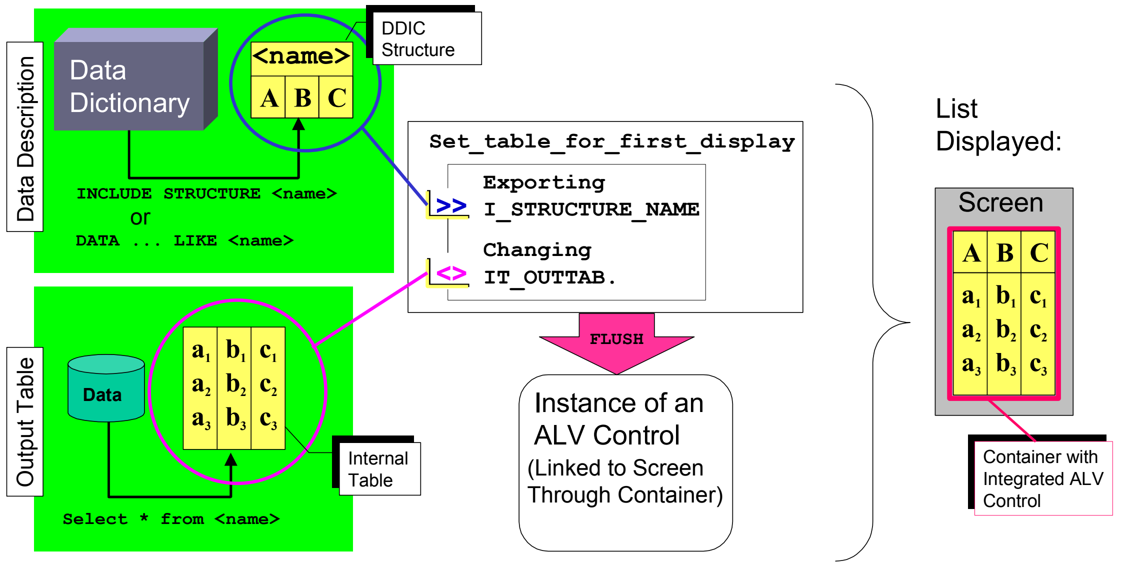
To pass the structure to the ALV Grid Control:

1. Reference the structure with **`LIKE`**, or include the structure in a separate structure with **`INCLUDE STRUCTURE`**.

   If you use **`INCLUDE STRUCTURE`**, you can have the ALV generate part of your output table automatically, and then add more fields. For a description of the procedure, see Generating the Field Catalog Semi-Automatically [Page 135].

2. Pass the structure in method set_table_for_first_display [Page 100] with parameter **`I_STRUCTURE_NAME`** to the control created.

If you do not want to add more fields to the structure or if you want to hide specific fields, you do not need to pass the field catalog (see graphic).



Generating the Field Catalog Manually

Purpose

It may be the case that the data you want to display is not at all or only partially represented by a structure in the Data Dictionary. Then you must use the fields of the field catalog to describe the structure of the output table.

Process Flow

The field catalog is defined in the Data Dictionary through table type **`LVC_T_FCAT`**. Each row of the field catalog table explains a field in your output table. Depending on whether a reference structure exists in the DDIC or not, you must at least fill the following fields of the field catalog structure for each field:

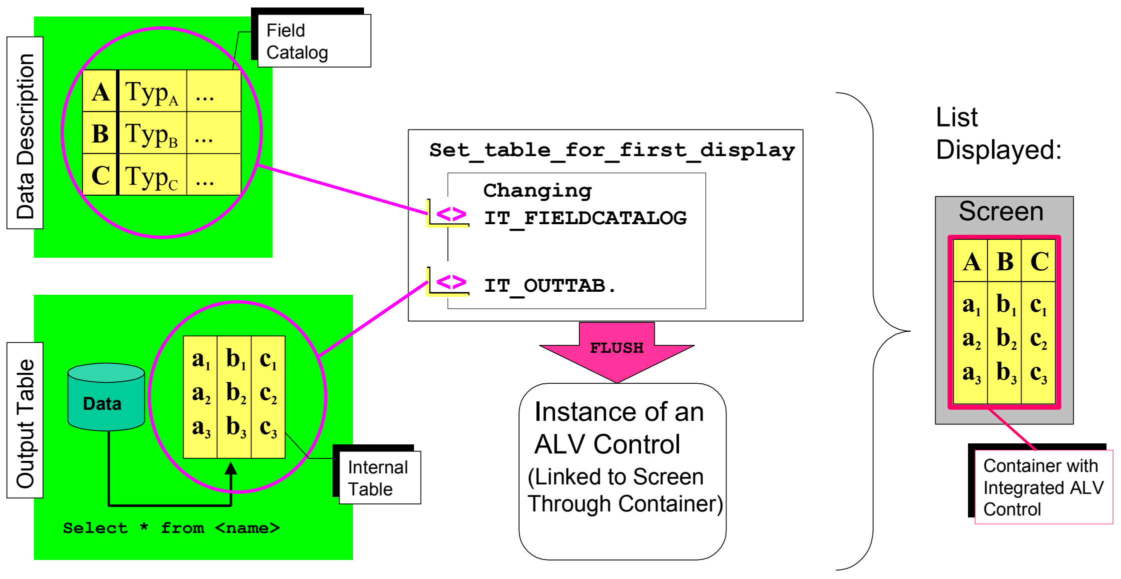
Required Fields of the Field Catalog

Output table fields *with* DDIC reference	Output table fields *without* DDIC reference	Explanation
`FIELDNAME`	**`FIELDNAME`**	Name of the field of the internal output table
`REF_TABNAME`		Name of the DDIC reference structure
`REF_FIELDNAME`		Name of the DDIC reference field (only needed if other than **`FIELDNAME`**)
	`INTTYPE`	ABAP data type of the field of the internal output table
	`OUTPUTLEN`	Column width
	`COLTEXT`	Column header
	`SELTEXT`	Column description in column selection for layout

See also: Texts in the Field Catalog [Ext.]

You pass the field catalog using parameter IT_FIELDCATALOG (see graphic).

Generating the Field Catalog Manually



Generating the Field Catalog Semi-Automatically

Purpose

When generating the field catalog semi-automatically, you combine structure information of the Data Dictionary with your own structure information. In this context, you can:

- modify or
- add structure descriptions of new fields to

the field catalog generated automatically using the DDIC structure. For example, this method is helpful for the following scenarios:

You want to display a Data Dictionary table without displaying all possible columns initially (using field **NO_OUT** of the field catalog).

You want to display additional columns containing icons or other information. See the table in Generating the Field Catalog Manually [Page 133] for the required fields. This section also explains how to define texts for a new column.

Process Flow

To generate a field catalog semi-automatically:

1. Declare an internal table of type **LVC_T_FCAT**.
2. Call function module **LVC_FIELDCATALOG_MERGE** and pass the DDIC structure of the output table and the internal table for the field catalog. The function module generates the field catalog and fills the internal table accordingly.
3. Read the rows you want to change, and adapt the fields accordingly. If your output table contains more fields than are stored in the Data Dictionary, you must append one row for each new field to the field catalog.

To display the output table with the settings of the field catalog, pass the field catalog in method set_table_for_first_display [Page 100].

Fields of the Field Catalog

The following table assigns a purpose to each field. For information on the minimum selection of required fields in the field catalog, see Generating the Field Catalog Manually [Page 133].

You use field **FIELDNAME** of the field catalog to define a reference to a field in the output table. All settings made through fields in the same row of the field catalog refer to the output column specified in **FIELDNAME**.

Internally, the ALV Grid Control uses the field **COL_ID** to identify columns at the frontend.

Alphabetic index

Field name	Short description	Purpose
CFIELDNAME	Field name for currency unit referenced	Value Display With Currency/Quantity Unit [Page 142]
CHECKBOX	Output as checkbox	Output Options of Columns [Page 145]
COL_ID	Numerical column identification (**read-only**)	Reference to the Output Table [Page 140]
COL_POS	Position of output column	Output Options of Columns [Page 145]
COLDDICTXT	Definition of DDIC text reference	Texts [Page 151]
COLTEXT	Column label for dialog functions	Texts [Page 151]
CURRENCY	Currency unit	Value Display With Currency/Quantity Unit [Page 142]
DD_OUTLEN	Output length in characters	Parameters for Fields Without DDIC Reference [Page 153]
DECIMALS_O	Number of decimal places for output	Formatting Column Contents [Page 148]
DECMLFIELD	Field name with DECIMALS specification	Formatting Column Contents [Page 148]
DO_SUM	Totals calculation for column values	Output Options of Columns [Page 145]
DRAGDROPID	Drag & Drop handle for Drag & Drop object	Other Fields [Page 155]
EDIT_MASK	EditMask for output	Formatting Column Contents [Page 148]
EMPHASIZE	Column color	Output Options of Columns [Page 145]

EXPONENT	Exponent for float representation	Formatting Column Contents [Page 148]
FIELDNAME	Field name of internal table field	Reference to the Output Table [Page 140]
HOTSPOT	Single-click sensitive	Output Options of Columns [Page 145]
HREF_HNDLE	Assign hyperlink	Output Options of Columns [Page 145]
ICON	Output as icon	Formatting Column Contents [Page 148]
INTLEN	Internal length in bytes	Parameters for Fields Without DDIC Reference [Page 153]
INTTYPE	ABAP data type (C,D,N,...)	Parameters for Fields Without DDIC Reference [Page 153]
JUST	Justification	Formatting Column Contents [Page 148]
KEY	Key column	Output Options of Columns [Page 145]
LOWERCASE	Lower case allowed/not allowed	Output Options of Columns [Page 145]
LZERO	Display leading zeros	Formatting Column Contents [Page 148]
NO_MERGING	Do not merge cells	Output Options of Columns [Page 145]
NO_OUT	Do not display columns	Output Options of Columns [Page 145]
NO_SIGN	Supress sign in display	Formatting Column Contents [Page 148]
NO_SUM	No totals calculation for column values	Output Options of Columns [Page 145]
NO_ZERO	Suppress zeros in display	Formatting Column Contents [Page 148]
OUTPUTLEN	Column width in characters	Output Options of Columns [Page 145]
QFIELDNAME	Field name for quantity unit referenced	Value Display With Currency/Quantity Unit [Page 142]
QUANTITY	Quantity unit	Value Display With Currency/Quantity Unit [Page 142]

REF_FIELD	Reference field name for internal table field	Reference to the Data Dictionary [Page 141]
REF_TABLE	Reference table name for internal table field	Reference to the Data Dictionary [Page 141]
REPREP	Property is selection criterion for report/report interface	Other Fields [Page 155]
REPTEXT	Header (DDIC text of the corresponding data element)	Texts [Page 151]
ROLLNAME	Data element for F1 help	Parameters for Fields Without DDIC Reference [Page 153]
ROUND	ROUND value	Formatting Column Contents [Page 148]
ROUNDFIELD	Field name with ROUND specification	Formatting Column Contents [Page 148]
SCRTEXT_L	Long field label (DDIC text of the corresponding data element)	Texts [Page 151]
SCRTEXT_M	Medium field label (DDIC text of the corresponding data element)	Texts [Page 151]
SCRTEXT_S	Short field label (DDIC text of the corresponding data element)	Texts [Page 151]
SELDDICTXT	Definition of DDIC text reference	Texts [Page 151]
SELTEXT	Column label for dialog function	Texts [Page 151]
SP_GROUP	Group key	Other Fields [Page 155]
STYLE	Output as pushbutton	Output Options of Columns [Page 145]
SYMBOL	Output as symbol	Formatting Column Contents [Page 148]
TECH	Technical fields	Output Options of Columns [Page 145]
TIPDDICTXT	Definition of DDIC text reference	Texts [Page 151]
TOOLTIP	Tool tip for column header	Texts [Page 151]
TXT_FIELD	Field name of internal table field	Other Fields [Page 155]
WEB_FIELD	Field name of internal table field (for hyperlink)	Output Options of Columns [Page 145]

Reference to the Output Table

Field name	Comp. type	Dtype(length)	Value range
	Use		
COL_ID	LVC_COLID	INT4(10)	Natural number
	You can only access this field in read-only mode. As of Release 4.6C, this field is used (instead of the field name) to access columns of the output table. You need this field, for example, if you use the method get_selected_cells_id [Page 77] or set_selected_cells_id [Page 96].		
FIELDNAME	LVC_FNAME	Char(30)	Field name of output table (required)
	You use this field to assign a field name of your output table to a row of the field catalog. All settings that you make in this row refer to the corresponding column of the output table.		

Reference to the Data Dictionary

Field name	Comp. type	Dtype(length)	Value range
	Use		
`REF_FIELD`	`LVC_RFNAME`	Char(30)	SPACE, name of a field in the Data Dictionary that corresponds to a field in the output table
	You must fill this field if: • the output table field described by the current entry in the field catalog has a corresponding field in the Data Dictionary and • the field name in the output table is not identical to the field name of the field in the Data Dictionary. If the field names are identical, it is sufficient to specify the DDIC structure or table in field **`REF_TABLE`** of the field catalog.		
`REF_TABLE`	`LVC_RTNAME`	Char(30)	SPACE, name of a structure or table from the Data Dictionary that corresponds to a field in the output table.
	You must fill this field only if the output table field described by the current entry in the field catalog has a corresponding entry in the Data Dictionary. Using this assignment, the ALV Grid Control can copy the text for the column header from the Dictionary, for example.		

Value Display with Currency/Quantity Unit

Use

Certain values of output fields may refer to units (such as kilogram) or currencies (such as Euro). To display these values in the correct unit-specific format (with the correct number of digits after the comma, for example), you must assign these fields to a currency or unit. There are three ways how you can do this in the field catalog:

- Assign a value field to an associated currency or unit field
- Explicitly specify a currency or unit for the entire column
- Format the value field manually (see: Formatting Column Contents [Page 148])

Reference to a Currency or Unit Field

You use fields **`CFIELDNAME`** and **`QFIELDNAME`** (see below) to assign value fields to a currency or unit field. This ensures that when totals are calculated for this column, these are displayed separately by unit.

The field catalog also contains an entry for the unit field. If you do not want to display the unit as a column in the list and do not want to allow users to interactively show it as a column, you can mark the field catalog entry for the unit field as a technical field by setting field **`TECH`**. This makes sense, for example, if the unit is always unique and therefore explicitly output in the list header by the caller.

For initial value or currency fields, you must consider the following points:

Totals calculation and display of value fields

	Value of unit field		
	not initial	*initial*	
Value of value field	*not initial*	Unit-specific display of digits after the comma in the list and in the totals	For such cells, ALV introduces the unit **`SPACE`**. In the totals display, this unit is given as a separate unit.
	initial	Unit-specific output as '0' (provided field **`NO_ZERO`** of the field catalog is initial). When calculating totals, the ALV uses the value '0' and the unit specified.	Output as **`SPACE`**. The value field is ignored when the totals are calculated.

Value or Unit for the Entire Column

For fields that use the same quantity or currency unit for all column values, the field catalog contains fields **`CURRENCY`** and **`QUANTITY`** (see below). You use these fields to determine a unit for your value field (such as **`USD`** or **`KG`**). This makes sense, for example, if there is only one unit or currency for the entire column (which has been entered by the user, for example). In this case, the output table does not need any additional fields.

Prerequisites

For the value field, you must consider the following points:

- The field is of ABAP data type P (see also **`INTTYPE`** in Parameters for Fields Without DDIC Reference [Page 153]).
- There is one field in the internal output table that contains the associated unit.

Relevant Fields in the Field Catalog

Field name	Comp. type / Use	Dtype(length)	Value range
	Use		
`CFIELDNAME`	**`LVC_CFNAME`**	Char(30)	SPACE, name of a field in the output tablee
	Defining a reference to *currency units*. The ALV links the field specified in **`FIELDNAME`** to the field for currency units specified in **`CFIELDNAME`**. The field specified in **`CFIELDNAME`** must have a separate entry in the field catalog.		
`CURRENCY`	**`LVC_CURR`**	Char(5)	SPACE, name of a currency
	Explicitly specifying a currency (such as **`DEM`**, **`USD`**). The ALV displays the values for the column specified in **`FIELDNAME`** according to the conventions for this currency.		
`QFIELDNAME`	**`LVC_QFNAME`**	Char(30)	SPACE, name of a field in the output table
	Defining a reference to *quantity units*. The ALV links the field specified in **`FIELDNAME`** to the field for quantity units specified in **`QFIELDNAME`**. The field specified in **`QFIELDNAME`** must have a separate entry in the field catalog.		
`QUANTITY`	**`LVC_QUAN`**	Char(3)	SPACE, name of a unit
	Explicitly specifying a unit (such as **`KG`**). The ALV displays the values for the column specified in **`FIELDNAME`** according to the conventions for this unit.		

Output Options of Columns

Field name	Comp. type	Dtype(length)	Value range
	Use		
CHECKBOX	`LVC_CHECKB`	CHAR(1)	SPACE, 'X'
	Outputting a checkbox. The checkbox cannot be modified by the user. Parameter `sel_mode` of the layout structure allows the user to select multiple rows in the grid control (see also: get_selected_rows [Page 79]).		
COL_POS	`LVC_COLPOS`	INT4(10)	natural number
	Relevant only if the relative column positions should not be identical to the sequence of fields in the field catalog when the list is displayed for the first time. The parameter determines the relative column position of the field for list output. The user can interactively modify the order of the columns. If this parameter is initial for each field catalog entry, the order of the columns corresponds to the sequence of fields in the field catalog.		
DO_SUM	`LVC_DOSUM`	Char(1)	SPACE, 'X'
	If this field is set, the ALV uses this field to calculate the total (this corresponds to the generic totals function in the toolbar.)		
EMPHASIZE	`LVC_EMPHSZ`	Char(4)	SPACE, 'X' or 'Cxyz' (x:'1'-'9'; y,z: '0'=off '1'=on)
	If the field is set to 'X', the ALV uses a pre-defined color for highlighting the column. If the character field begins with 'C' (color code), the remaining numbers have the following meaning: • x: color number • y: intensified display on/off • y: inverse display on/off For more information on color coding, see the F1 help on the `FORMAT` statement.		

Output Options of Columns

HOTSPOT	**LVC_HOTSPT**	Char(1)	SPACE, 'X'
	If this field is set, all cells of this column are hotspot-sensitive.		
HREF_HNDL	**INT4**	INT4(10)	Natural number
	Handle to which an URL is assigned. The ALV Grid Control displays all cells of the column as hyperlinks. You must maintain the target address of the hyperlink in a table of type **LVC_T_HYPE** and pass it using set_table_for_first_display [Page 100].		
KEY	**LVC_KEY**	Char(1)	SPACE, 'X'
	If this field is set, the ALV Grid Control color-codes the column as a key field and fixes this column during horizontal scrolling. The order of the key columns in the ALV Grid Control can be modified interactively. In contrast to the SAP List Viewer, the ALV Grid Control allows you to directly hide key columns with **NO_OUT** (field **KEY_SEL** is not used).		
LOWERCASE	**LOWERCASE**	Char(1)	SPACE, 'X'
	If this field is set, the ALV Grid Control recognizes upper/lower case in the output table. This affects the sorting of fields, for example.		
NO_OUT	**LVC_NOOUT**	Char(1)	SPACE, 'X'
	If you set this field, you hide the relevant column in the list. Nevertheless, the column is available in the field selection and can be interactively selected by the user as a display field. The ALV displays the contents of hidden fields on the detail screen for a row in the grid control.		
NO_MERGING	**CHAR01**	Char(1)	SPACE, 'X'
	If this field is set, cells with the same value are not merged into a single cell when this column is sorted.		
NO_SUM	**LVC_NOSIGN**	Char(1)	SPACE, 'X'
	If you set this field, you lock totals calculation for the relevant field.		
OUTPUTLEN	**LVC_OUTLEN**	Numc(6)	0 (default setting), n

	Determines the column width of the field: • If the field has a reference to the Data Dictionary, you can leave the field set to its initial value. In this case, the ALV adopts the output length of the relevant domain. • For fields without reference to the DDIC, you must specify the desired field output length.		
STYLE	**LVC_STYLE**	RAW(4)	Attribute **CL_GUI_ALV_GRID=>MC_STYLE_BUTTON**
	Displays all cells of this column as pushbuttons. If a user clicks the pushbutton, the event button_click [Page 109] is triggered.		
TECH	**LVC_TECH**	Char(1)	SPACE, 'X'
	If this field is set, the relevant field is not displayed on the list and cannot be shown interactively. The field is only known in the field catalog. (For example, it must not be specified as a sorting criterion).		
WEB_FIELD	**LVC_NAME**	Char(30)	Name of a field of the output table
	You can use this field to define hyperlinks at cell level [Page 36].		

Formatting Column Contents

Fields **DECIMALS_O**, **EDIT_MASK**, **EXPONENT**, **NO_SIGN** and **ROUND** correspond to WRITE additions **DECIMALS**, **USING EDIT MASK**, **EXPONENT**, **NO-SIGN** and **ROUND**. For information on combining these fields with fields **CURRENCY** and **QUANTITY** (for **WRITE: UNIT**), see the F1 help for the WRITE edit option.

Field name	**Comp. type**	**Dtype(length)**	**Value range**
	Use		
DECIMALS_O	**LVC_DECMLS**	Char(6)	initial, natural number
	Relevant only if no currency field is assigned to the field and if **CURRENCY** has a value. The value in this field determines the number of digits to be displayed after the comma.		
DECMFIELD	**LVC_DFNAME**	Char(30)	SPACE, field name of output table
	Defining the digits after the comma on a row-by-row basis. You can use an additional field in the output table to determine how many digits are to be displayed after the comma in each row.		
EDIT_MASK	**LVC_EDTMSK**	Char(60)	SPACE, conv (conversion exit)
	If you set a conversion exit (for example, conv = '**==ALPHA**' for function module **CONVERSION_EXIT_ALPHA_OUTPUT**), you enforce output conversion for the associated output field. (See also F1 help for **WRITE** edit option **USING EDIT MASK)**. **See also:** Using the Conversion Exit [Page 40]		
EXPONENT	**LVC_EXPONT**	Char(3)	initial, integer
	Determines a fixed exponent for the field. The field must be of internal type F. (See also F1 help for WRITE edit option **EXPONENT**).		
ICON	**LVC_ICON**	Char(1)	SPACE, 'X'

	If this field is set, the column contents of the output table are output as an icon. The column contents must consist of valid icon strings (`@xx@` or `@xx\Q`<Quickinfo>`@`). You should consider the problem of printing icons.		
JUST	**LVC_JUST**	Char(1)	SPACE, 'R', 'L','C'
	Relevant only to fields of data type **CHAR** or **NUMC**. Justifications: • 'R': right justified • 'L': left justified • 'C': centered How the column header is justified, depends on how the column contents are justified. You cannot justify the column header separately.		
LZERO	**LVC_LZERO**	Char(1)	SPACE, 'X'
	Relevant only to fields of data type **NUMC**. In the default setting, the ALV Grid Control displays these fields right justified without leading zeros. If you set **LZERO**, leading zeros are displayed.		
NO_SIGN	**LVC_NOSIGN**	Char(1)	SPACE, 'X'
	Relevant only to value fields. If you set **NO-SIGN**, values are displayed without signs (see also F1 help for the WRITE edit options).		
NO_ZERO	**LVC_NOZERO**	Char(1)	SPACE, 'X'
	If **NO_ZERO** is set, no zeros are displayed for initial value fields. The cell remains empty.		
ROUND	**LVC_ROUND**	Char(10)	initial, natural number
	Displaying a value of type P scaled by the power of ten. For positive ROUND values, the ALV Grid Control moves the comma to the left; otherwise, it moves the comma to the right. (See also F1 help for the WRITE edit options)		
ROUNDFIELD	**LVC_RNDFN**	Char(30)	
	Defining scaled output on a row-by-row-basis. You can use an additional field in the output table to determine how the relevant field is scaled in each row.		
SYMBOL	**LVC_SYMBOL**	Char(1)	SPACE, 'X'

	If this field is set, the column contents are displayed as a symbol. The column contents of the internal table must consist of valid symbol signs. The caller should consider the problem of printing symbols. (It is usually possible to print symbols, but they may not be output correctly depending on the printer configuration).

Texts

You use these fields to determine which texts are used by the ALV Grid Control:

- As column header
- As tool tip (informative text displayed if the user positions the mouse pointer on a column)
- As column selection text (on the dialog box for defining the layout, sorting or filtering).

See Texts in the Field Catalog [Ext.] for an explanation of how the fields are related to each other.

Field name	Comp. type	Dtype(length)	Value range
	Use		
`COLDDICTXT`	`LVC_DDICT`	Char(1)	SPACE, 'L', 'M', 'S' and 'R'
	Relevant only to fields with reference to the Data Dictionary. You use values 'L', 'M', 'S' or 'R' to determine if `SCRTEXT_L`, `SCRTEXT_M`, `SCRTEXT_S` or `REPTEXT` is used as the column header.		
`COLTEXT`	`LVC_TXT`	Char(40)	Freely definable text
	Determines the column header of the column. You should assign a value to this field if it does not have a Data Dictionary reference.		
`REPTEXT`	`REPTEXT`	Char(55)	Text copied from the Data Dictionary
	Relevant only to fields with reference to the Data Dictionary. For such fields, the ALV Grid Control copies the field label for the header of the corresponding data element into this field.		
`SCRTEXT_L`	`SCRTEXT_L`	Char(40)	Text copied from the Data Dictionary
	Relevant only to fields with reference to the Data Dictionary. For such fields, the ALV Grid Control copies the **long** field label of the corresponding data element into this field.		
`SCRTEXT_M`	`SCRTEXT_M`	Char(20)	Text copied from the Data Dictionary

	Relevant only to fields with reference to the Data Dictionary. For such fields, the ALV Grid Control copies the **medium** field label of the corresponding data element into this field.		
`SCRTEXT_S`	`SCRTEXT_S`	Char(10)	Text copied from the Data Dictionary
	Relevant only to fields with reference to the Data Dictionary. For such fields, the ALV Grid Control copies the **short** field label of the corresponding data element into this field.		
`SELDDICTXT`	`LVC_DDICT`	Char(1)	SPACE, 'L', 'M', 'S' and 'R'
	Relevant only to fields with reference to the. You use values 'L', 'M', 'S' or 'R' to determine if `SCRTEXT_L`, `SCRTEXT_M`, `SCRTEXT_S` or `REPTEXT` is used as the text for column selection.		
`SELTEXT`	`LVC_TXT`	Char(40)	Freely definable text
	Determines the text to be used in the column selection for the column. You should assign a value to this field if it does not have a Data Dictionary reference.		
`TIPDDICTXT`	`LVC_DDICT`	Char(1)	SPACE, 'L', 'M', 'S' and 'R'
	Relevant only to fields with reference to the Data Dictionary. You use values 'L', 'M', 'S' or 'R' to determine if `SCRTEXT_L`, `SCRTEXT_M`, `SCRTEXT_S` or `REPTEXT` is used as the tool tip.		
`TOOLTIP`	`LVC_TIP`	Char(40)	Freely definable texxt
	Determines the text to be used as the tool tip for the column. You should assign a value to this field if it does not have a Data Dictionary reference.		

Parameters for Fields Without DDIC Reference

See also: Texts [Page 151]

Field name	Comp. type	Dtype(length)	Value range
	Use		
DD_OUTLEN	**LVC_DDLEN**	NUMC(6)	0 (initial), n
	You use this field to specify the field output length for external display. This is only relevant to fields without reference to the Data Dictionary for which you want to modify the output using a conversion exit (see Using the Conversion Exit [Page 40]). The column width (field **OUTPUTLEN** of the field catalog) does not need to be identical to the output length for external display (**DD_OUTLEN**).		
INTLEN	**INTLEN**	NUMC(6)	0 (initial), n
	You use this field to specify the field output length for internal display. This is only relevant to fields without reference to the Data Dictionary, for which you want to modify the output using a conversion exit (see Using the Conversion Exit [Page 40]).		
INTTYPE	**INTTYPE**	Char(1)	ABAP data type, see value range of domain **INTTYPE**
	Only required for field without reference to the Data Dictionary.		
ROLLNAME	**LVC_ROLL**	Char(30)	SPACE, name of a data element of the Data Dictionary
	If you want to provide F1 help for an output field without Data Dictionary reference or if you want to define a different F1 help than that stored in the DDIC for a field with DDIC reference, you can use this field. If F1 help is called for this field, the documentation for the data element assigned is displayed. If **ROLLNAME** is initial for fields with Data Dictionary reference, the documentation for the data element of the referenced field of the Data Dictionary is displayed.		

Other Fields

Field name	Comp. type	Dtype(length)	Value range
	Use		
DRAGDROPID	**LVC_DDID**	INT4(10)	Drag & Drop handle
	You use this field to define a D&D Behavior for Special Columns [Page 26].		
REPREP	**LVC_CRPRP**	Char(1)	SPACE, 'X'
	'X' = if the report/report interface is called, the value of this field is passed in the selected jump line of the interface as a selection criterion. Prerequisites: • The report/report interface is available in the system (function group **RSTI**, table **TRSTI**) • The report/report interface has been activated using method activate_reprep_interface [Page 68].		
SP_GROUP	**LVC_SPGRP**	Char(4)	SPACE, four-digit group key
	You use the group key to group several fields together. On the dialog box for defining a layout, the user can then limit the list of hidden columns to this group. **See also:** Grouping Fields Together [Page 38]		
TXT_FIELD	**LVC_FNAME**	Char(30)	SPACE, field name of the output table
	You can use this field to define a reference to a field that is used as the description for the current field. If a subtotal is calculated for the current field, the ALV Grid Control displays the descriptions in the field assigned. Example: Your output table contains one column for material numbers and one column for the description of what these numbers mean (such as clockwork). If you calculate subtotals for the material numbers, only these numbers are usually displayed as the subtotals text. Based on the link to **TXT_FIELD**, you can refer to the corresponding column with the material description. This description is then used as the subtotals text.		

The Layout Structure

Definition

The layout structure is of type `LVC_S_LAYO`. It contains fields for settinig graphical properties of the grid control, displaying exceptions, calculating totals and enabling specific interaction options.

Structure

Alphabetic Index

Field name	Short description	Purpose
`CTAB_FNAME`	Field name of table with cell color codes	Colors [Page 162]
`CWIDTH_OPT`	Optimize column width	Properties of the Grid Control [Page 159]
`DETAILINIT`	Display initial values on detail screen	Interaction Control [Page 164]
`DETAILTITL`	Title bar of detail screen	Interaction Control [Page 164]
`EXCP_CONDS`	Inherit exceptions to (sub) total	Exceptions [Page 161]
`EXCP_FNAME`	Field name with exception code	Exceptions [Page 161]
`EXCP_LED`	Exception as LED	Exceptions [Page 161]
`EXCP_ROLLN`	Data element for exception documentation	Exceptions [Page 161]
`GRID_TITLE`	Text of title bar	Properties of the Grid Control [Page 159]
`INFO_FNAME`	Name of field with row color codes	Colors [Page 162]
`KEYHOT`	Key columns as hotspot	Interaction Control [Page 164]
`NO_HEADERS`	Hide column headers	Properties of the Grid Control [Page 159]
`NO_HGRIDLN`	Hide horizontal grid lines	Properties of the Grid Control [Page 159]
`NO_MERGING`	Disable cell merging	Properties of the Grid Control [Page 159]
`NO_ROWMARK`	Hide row marks	Properties of the Grid Control [Page 159]
`NO_TOOLBAR`	Hide toolbar	Properties of the Grid Control [Page 159]
`NO_TOTARR`	Do not display totals arrrows	Totals Options [Page 163]

`NO_TOTEXP`	Do not display expand icons	Totals Options [Page 163]
`NO_TOTLINE`	Do not display totals line	Totals Options [Page 163]
`NO_VGRIDLN`	Hide vertical grid lines	Properties of the Grid Control [Page 159]
`NUMC_TOTAL`	Allow totals calculation for NUMC fields	Totals Options [Page 163]
`S_DRAGDROP`	Drag & Drop control settings	Interaction Control [Page 164]
`SEL_MODE`	Selection mode	Properties of the Grid Control [Page 159]
`SGL_CLK_HD`	Single click on column header	Interaction Control [Page 164]
`SMALLTITLE`	Title size	Properties of the Grid Control [Page 159]
`STYLEFNAME`	Name of the cell table for pushbuttons	Interaction Control [Page 164]
`TOTALS_BEF`	Totals output before single records	Totals Options [Page 163]
`ZEBRA`	Alternating cell color (zebra pattern) for print output	Colors [Page 162]

Integration

Generally, it makes sense to fill the fields of the structure before first list display and pass them in method set_table_for_first_display [Page 100].

Methods get_frontend_layout [Page 74] and set_frontend_layout [Page 92] allow you to modify settings in the layout structure after list output.

Properties of the ALV Grid Control

General display options

Field name	Description	Value range
`CWIDTH_OPT`	If this field is set, the ALV Grid Control optimizes the column width. You can then see the column header and the contents of the cells of this column.	SPACE, 'X'
`SMALLTITLE`	If this field is set, the title size in the grid control is set to the font size of the column header.	SPACE, 'X'

Grid customizing

Field name	Description	Value range
`GRID_TITLE`	Title between grid control and toolbar	Character string of 70 characters at most
`NO_HEADERS`	If this field is set, column headers are hidden.	SPACE, 'X'
`NO_HGRIDLN`	If this field is set, columns are displayed without horizontal grid lines.	SPACE, 'X'
`NO_MERGING`	If this field is set, cells are not merged when a column is sorted.	SPACE, 'X'
`NO_ROWMARK`	If this field is set, the button at the beginning of a row is hidden in selection modes *cell selection* (`SEL_MODE = 'D'`) and *column/row selection* (`SEL_MODE = 'A'`).	SPACE, 'X'
`NO_TOOLBAR`	If this field is set, the toolbar is hidden.	SPACE, 'X'
`NO_VGRIDLN`	If this field is set, columns are displayed without vertical grid lines.	SPACE, 'X'
`SEL_MODE`	Set the selection mode (see table below).	SPACE, 'A', 'B', 'C', 'D'

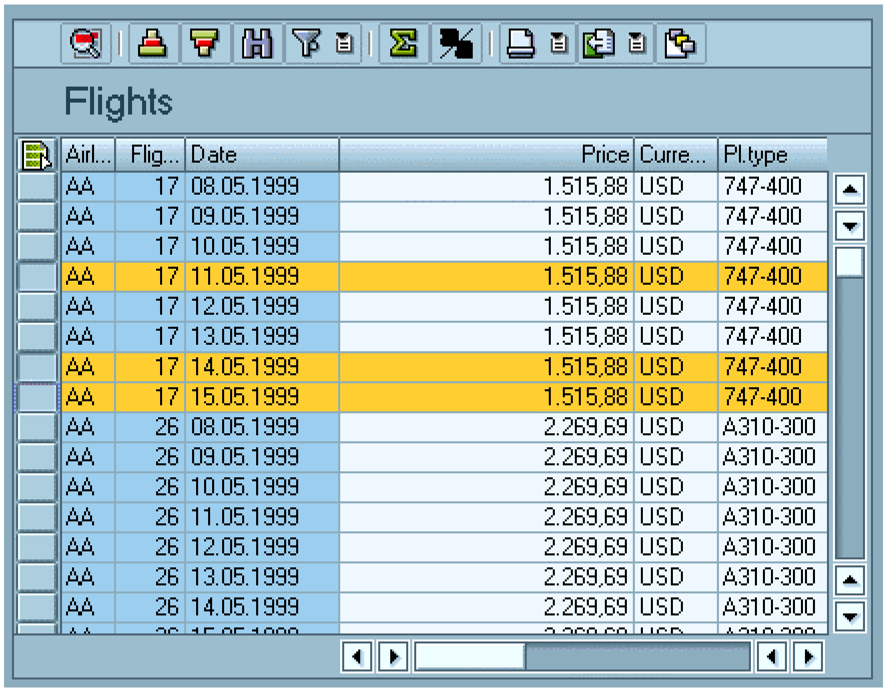
Selection modes for `SEL_MODE`

Value	Mode	Possible selections	Comment
`SPACE`	same as 'B'	see 'B'	Default setting
`'A'`	Column and row selection (see graphic)	• Multiple columns • Multiple rows	The user selects the rows through pushbuttons at the left border of the grid control.
`'B'`	Simple selection, list box	• Multiple columns • Multiple rows	

Properties of the ALV Grid Control

`'C'`	Multiple selection, list box	• Multiple columns • Multiple rows	
`'D'`	Cell selection	• Multiple columns • Multiple rows • Any cells	The user selects the rows through pushbuttons at the left border of the grid control.

ALV Grid Control with column and row selection



Airl...	Flig...	Date	Price	Curre...	Pl.type
AA	17	08.05.1999	1.515,88	USD	747-400
AA	17	09.05.1999	1.515,88	USD	747-400
AA	17	10.05.1999	1.515,88	USD	747-400
AA	17	11.05.1999	1.515,88	USD	747-400
AA	17	12.05.1999	1.515,88	USD	747-400
AA	17	13.05.1999	1.515,88	USD	747-400
AA	17	14.05.1999	1.515,88	USD	747-400
AA	17	15.05.1999	1.515,88	USD	747-400
AA	26	08.05.1999	2.269,69	USD	A310-300
AA	26	09.05.1999	2.269,69	USD	A310-300
AA	26	10.05.1999	2.269,69	USD	A310-300
AA	26	11.05.1999	2.269,69	USD	A310-300
AA	26	12.05.1999	2.269,69	USD	A310-300
AA	26	13.05.1999	2.269,69	USD	A310-300
AA	26	14.05.1999	2.269,69	USD	A310-300

Exceptions

See Output of Exceptions [Page 29] for a description of how to display exceptions in a list.

Field name	Description	Value range
EXCP_CONDS	If this field is set, the ALV also shows an exception in the (sub)totals line. As the color for this exception, the ALV uses the smallest exception value ('1': red, '2': yellow, '3' green) of the rows to which the (sub)total refers.	SPACE, 'X'
EXCP_FNAME	Field name of the output table for displaying an exception	Character string of 30 characters at most
EXCP_LED	The exception is not displayed as a traffic light, but as an LED.	SPACE, 'X'
EXCP_ROLLN	Name of a data element. The F1 help for this data element is then called for the exception column. In addition, the long field label of the element is displayed as the tool tip for this column.	Character string of 30 characters at most

Colors

You color columns using the field catalog (see also **EMPHASIZE** in Output Options of Columns [Page 145]).

Field name	Description	Value range
CTAB_FNAME	Field name in output table for coloring cells (see: Coloring Cells [Page 32])	Character string of 30 characters at most
INFO_FNAME	Field name in output table for coloring rows (see: Coloring Rows [Page 31])	Character string of 30 characters at most
ZEBRA	If this field is set, the list shows a striped pattern in the print preview and when it is printed.	SPACE, 'X'

Totals Options

Field name	Description	Value range
`NO_TOTARR`	The ALV Grid Control displays arrows in the totals line and the subtotals line that additionally indicate the totalling area. Set this parameter to suppress these arrows.	SPACE, 'X'
`NO_TOTEXP`	An icon displayed at the beginning of a (sub)totals line indicates whether the line has been expanded or not. Set this parameter to suppress this icon.	SPACE, 'X'
`NO_TOTLINE`	If this field is set, only subtotals, but no totals, are displayed.	SPACE, 'X'
`NUMC_TOTAL`	If this field is set, the user can calculate totals for fields of data type **`NUMC`** (normally, users are not allowed to do this).	SPACE, 'X'
`TOTALS_BEF`	If this field is set, the ALV displays totals calculated as the first rows in the grid control. Subtotals are displayed before a new value of the subtotals criterion.	SPACE, 'X'

Interaction Control

Field name	Description	Value range
DETAILINIT	If this field is set, the detail screen also shows columns with initial values.	SPACE, 'X'
DETAILTITL	Title in the title bar of the detail screen.	Character string of 30 characters at most
S_DRAGDROP	Structure for Drag & Drop settings (see: Drag & Drop With the ALV Grid Control [Page 24]).	
KEYHOT	If this field is set, all key fields are hotspot-sensitive. If a key field is clicked once, event hotspot_click [Page 114] is triggered.	SPACE, 'X'
SGL_CLK_HD	Enables the *single click on column header* function. This function sorts the list in ascending order when the column is clicked for the first time, and then in descending order when the column is clicked a second time.	SPACE, 'X'
STYLEFNAME	You use this field to pass the name of the cell table for displaying cells as pushbuttons (see also: Displaying Cells as Pushbuttons [Page 34]).	Character string of 30 characters at most

The Print Structure

Use

The print structure contains fields for settings when the list is printed.

Structure

Field name	Description	Value range
GRPCHGEDIT	Enables user-definable group change editing for the print preview mode. If this field is set, the jump to the SAP List Viewer is configured accordingly. On the sort dialog box, the user can then determine how a sorting criterion value change is indicated graphically: as a page break or as an underline. Using the sort table [Ext.] you can dynamically set this formatting.	SPACE, 'X'
NO_COLWOPT	The ALV Grid Control sets all columns to their optimum width before the list is printed or displayed in the print preview. If you set this parameter, this default setting is overridden.	SPACE, 'X'
PRNTLSTINF	Prints list information. If this field is set, information on sorting, subtotals and filters defined as well as data statistics are printed at the beginning of the list.	SPACE, 'X'
PRNT_TITLE	Specifies the time at which the grid title is to be printed (see also parameter **GRID_TITLE** in Properties of the ALV Grid Control [Page 159]).	0-3 with the following meaning: • 0: Before the event **PRINT_TOP_OF_LIST** • 1: After the event **PRINT_TOP_OF_LIST** • 2: Before the event **PRINT_TOP_OF_PAGE** • 3: After the event **PRINT_TOP_OF_PAGE**
RESERVELNS	Number of reserved rows for event print_end_of_page [Page 122]. If no number is specified, the text specified there is overwritten by the list.	Natural number

Integration

Lists that you display with the ALV Grid Control are printed with the SAP List Viewer. The print preview mode takes the user directly to the SAP List Viewer (you can disable this feature with field **PRINT**). The settings made by the user here (such as defining a list with multiple lines) are considered when the list is printed.

The print output of event **PRINT_END_OF_PAGE** and field **PRNTLSTINF** is not visible in the print preview of the SAP List Viewer. If users create a spool request first, they can check the final list layout in transaction **SP01**.

Methods of the OO Control Framework

Methods of Class CL_GUI_CFW

The class `CL_GUI_CFW` contains static methods that apply to all instantiated custom controls when you call them.

dispatch

Use this method to dispatch application events (**see** Event Handling [Ext.]) to the event handlers registered for the events. If you do not call the method within the PAI event of your application program, it is called automatically by the system after the PAI has been processed. The method returns a return code from which you can tell if the call was successful.

```
CALL METHOD cl_gui_cfw=>dispatch
     IMPORTING return_code = return_code.
```

Parameters	Description
return_code	`cl_gui_cfw=>rc_found`: The event was successfully directed to a handler method. `cl_gui_cfw=>rc_unknown`: The event was not registered in the event list. `cl_gui_cfw=>rc_noevent`: No event was triggered in a control. The function code was therefore a normal one (for example, from a menu entry). `cl_gui_cfw=>rc_nodispatch`: No handler method could be assigned to the event.

An event can only be dispatched once. After that, it is "spent". Consequently, attempting to dispatch the events a second time does not trigger the handler events again.

flush

Use this method to synchronize the automation queue [Ext.]. The buffered operations are sent to the frontend using GUI RFC. At the frontend, the automation queue is processed in the sequence in which you filled it.

If an error occurs, an exception is triggered. You must catch and handle this error. Since it is not possible to identify the cause of the error from the exception itself, there are tools available in the Debugger and the SAPgui to enable you to do so.

Debugger: Select the option *Automation Controller: Always process requests synchronously*. The system then automatically calls the method **`cl_gui_cfw=>flush`** after each method called by the Automation Controller.

SAPGUI: In the SAPgui settings, under *Trace*, select *Automation*. The communication between the application server and the Automation Controller is then logged in a trace file that you can analyze at a later date.

```
CALL METHOD cl_gui_cfw=>flush
          EXCEPTIONS CNTL_SYSTEM_ERROR = 1
                 CNTL_ERROR = 2.
```

Do not use any more synchronizations in your program than are really necessary. Each synchronization opens a new RFC connection to the SAPgui.

get_living_dynpro_controls

This method returns a list of reference variables to all active custom controls.

```
CALL METHOD cl_gui_cfw=>get_living_dynpro_controls
                  IMPORTING control_list = control_list.
```

Parameters	Description
`control_list`	List of reference variables of active custom controls. The list has the type **`CNTO_CONTROL_LIST`** (defined in class **`CL_GUI_CFW`**).

set_new_ok_code

You may only use this method in the handler method of a system event. It sets an **`OK_CODE`** that triggers PAI processing. This means that data is transferred from the screen to the program, and you can take control of the program in your PAI modules.

```
CALL METHOD cl_gui_cfw=>set_new_ok_code
        EXPORTING new_code = new_code
        IMPORTING     rc = rc.
```

Parameters	**Description**
new_code	Function code that you want to place in the **`OK_CODE`** field (**`SY-UCOMM`**).
return_code	**`cl_gui_cfw=>rc_posted`**: The OK_CODE was set successfully and the automatic field checks and PAI will be triggered after the event handler method has finished. **`cl_gui_cfw=>rc_wrong_state`**: The method was not called from the handler method of a system event. **`cl_gui_cfw=>rc_invalid`**: The **`OK_CODE`** that you set is invalid.

update_view

Calling the flush [Page 170] method only updates the automation queue if the queue contains return values.

If you have a queue with no return values, and want to ensure that it is synchronized, you can use the Control Framework method **`CL_GUI_CFW=>UPDATE_VIEW`**. You should only use this method if you absolutely need to update the GUI. For example, you might have a long-running application in which you want to provide the user with regular updates on the status of an action.

```
CALL METHOD cl_gui_cfw=>update_view
          EXCEPTIONS CNTL_SYSTEM_ERROR = 1
                CNTL_ERROR       = 2.
```

Methods of Class CL_GUI_OBJECT

The class **`CL_GUI_OBJECT`** contains important methods for custom control wrappers. The only one relevant for application programs is the is_valid [Page 176] method.

free

Use this method to destroy a custom control at the frontend. Once you have called this method, you should also initialize the object reference (`FREE my_control`).

```
CALL METHOD my_control->free
    EXCEPTIONS cntl_error        = 1
               cntl_system_error = 2.
```

is_valid

This method informs you whether a custom control for an object reference still exists at the frontend.

```
CALL METHOD my_control->is_valid
          IMPORTING result = result.
```

Parameters	Description
result	0: Custom control is no longer active at the frontend 1: Custom control is still active

Methods of Class CL_GUI_CONTROL

The class `CL_GUI_CONTROL` contains methods that you need to set control attributes (for example, displaying the control), register events, and destroy controls.

constructor

This method is called by the control wrapper when you instantiate a control.

To instantiate a SAP control, always call the constructor of its class.

```
CREATE OBJECT my_control
  EXPORTING   clsid          = clsid
          lifetime        = lifetime
          shellstyle       = shellstyle
          parent          = parent
          autoalign        = autoalign
  EXCEPTIONS cntl_error        = 1
          cntl_system_error = 2
          create_error      = 3
          lifetime_error    = 4.
```

Parameters	Description
clsid	ID of the class
lifetime	Lifetime management parameter. The following values are permitted: **my_control->lifetime_imode**: The control remains alive for the duration of the internal session (that is, until the session is ended by one of the following statements: **leave program. leave to transaction. set screen 0, leave screen.**). After this, the finalize [Page 180] method is called. **my_control->lifetime_dynpro**: The control remains alive for the lifetime of the screen instance, that is, for as long as the screen remains in the stack. After this, the free [Page 175] method is called. Using this mode automatically regulates the visibility of the control. Controls are only displayed when the screen on which they were created is active. When other screens are active, the controls are hidden. **my_control->lifetime_default**: If you create the control in a container, it inherits the lifetime of the container. If you do not create the control in a container (for example, because it is a container itself), the lifetime is set to **my_control->lifetime_imode**.
Shellstyle	Controls the appearance and behavior of the control You can pass any constants from the ABAP include <**CTLDEF**> that begin with WS. You can combine styles by adding the constants together. The default value sets a suitable combination of style constants internally.
parent	Container in which the SAP Picture Control can be displayed (**see also** SAP Container [Ext.]).
autoalign	' ': Control is not automatically aligned 'X': Control is automatically aligned. This uses the maximum available space within a container.

finalize

This method is redefined by the relevant control wrapper. It contains specific functions for destroying the corresponding control. This method is called automatically by the free [Page 175] method, before the control is destroyed at the frontend.

```
CALL METHOD my_control->finalize.
```

get_focus

This static method returns the object reference of the control that has the focus.

```
CALL METHOD cl_gui_control=>get_focus
    IMPORTING  control          = control
    EXCEPTIONS cntl_error       = 1
           cntl_system_error = 2.
```

Parameters	Description
control	Object reference (**`TYPE REF TO cl_gui_control`**) to the control that has the focus.

get_height

This method returns the height of the control.

```
CALL METHOD control->get_height
    IMPORTING  height          = height
    EXCEPTIONS cntl_error       = 1.
```

Parameters	Description
height	Current height of the control

get_registered_events

This method returns a list of all events registered for custom control **`my_control`**.

```
CALL METHOD my_control->get_registered_events
    IMPORTING  events     = events
    EXCEPTIONS cntl_error = 1.
```

Parameters	**Description**
events	Table of events that you want to register for the custom control **`my_control`**.

The table **`events`** is a list of the events that you want to register. It is defined with reference to table type **`CNTL_SIMPLE_EVENTS`**. The table type is based on the structure **`CNTL_SIMPLE_EVENT`**, which consists of the following fields:

Field	**Description**
EVENTID	Event name
APPL_EVENT	Indicates whether the event is a system event (initial) or an application event (X).

The values that you assign to the field **`EVENTID`** are control-specific and therefore described in the documentation of the individual controls.

For general information about event handling, refer to the Event Handling [Ext.] section of the SAP Control Framework documentation.

get_width

This method returns the width of the control.

```
CALL METHOD control->get_width
    IMPORTING  width           = width
    EXCEPTIONS cntl_error       = 1.
```

Parameters	Description
width	Current width of the control

is_alive

This method informs you whether a custom control for an object reference still exists at the frontend.

```
CALL METHOD my_control->is_alive
    RETURNING state = state.
```

Parameters	Description
state	**`my_control->state_dead`**: Custom control is no longer active at the frontend **`my_control->state_alive`**: Custom control is active on the current screen. **`my_control->state_alive_on_other_dynpro`**: Custom control is not active on the current screen, but is still active (but invisible) at the frontend.

set_alignment

Use this method to align the custom control within its container:

```
CALL METHOD my_control->set_alignment
    EXPORTING  alignment        = alignment
    EXCEPTIONS cntl_error       = 1
           cntl_system_error = 2.
```

Parameters	Description
alignment	Control alignment

The `alignment` parameter may consist of combinations of the following alignments:

Name	Description
my_control->align_at_left	Alignment with left-hand edge
my_control->align_at_right	Alignment with right-hand edge
my_control->align_at_top	Alignment with top edge
my_control->align_at_bottom	Alignment with bottom edge

You can combine these parameters by adding the components:

alignment = my_control->align_at_left + my_control->align_at_top.

set_focus

Use this static method to set the focus to a custom control.

```
CALL METHOD cl_gui_control=>set_focus
     EXPORTING  control          = control
     EXCEPTIONS cntl_error       = 1
                cntl_system_error = 2.
```

Parameters	Description
control	Object reference (`TYPE REF TO cl_gui_control`) to the control on which you want to set the focus.

set_position

Use this method to place the control at a particular position on the screen.

The position of the control is usually determined by its container.

```
CALL METHOD my_control->set_position
    EXPORTING  height        = height
           left         = left
           top           = top
           width         = width
    EXCEPTIONS cntl_error        = 1
           cntl_system_error = 2.
```

Parameters	**Description**
height	Height of the control
left	Left-hand edge of the control
top	Top edge of the control
width	Width of the control

set_visible

Use this method to change the visibility of a custom control.

```
CALL METHOD my_control->set_visible
     EXPORTING  visible          = visible
     EXCEPTIONS cntl_error       = 1
                cntl_system_error = 2.
```

Parameters	Description
visible	`X`: Custom control is visible `' '`: Custom control is not visible