Using SAP List Viewer (ALV) for List Display

An Introduction to the ALV Tool Family for Developers

December 2004

Table of Contents

Summary

In this article we provide an overview of existing ALV tools and current activities for further improvements. Our intention is to help application developers who are considering using ALV, for example to migrate their WRITE lists because of accessibility requirements.

Choosing the Right ALV Tool

If you wish to migrate your lists to ALV, first you have to decide which ALV tool is appropriate for your application. Whenever it is possible to do so, you should use ALVFullscreen instead of ALVGrid, as the former can be switched to ALVList if system performance requires doing so (this switch is done system wide, i.e. application-independent).

The following tools exist:

- **ALVList:** This is a list based ALV displaying a simple list on its own screen.
- **ALVFullscreen:** This is a grid control-based ALV, displaying a simple list on its own screen.
- **ALVGrid:** Class CL_GUI_ALV_GRID, a control-based ALV displaying a simple list in a container to be positioned in a Dynpro.
- **ALVTree:** Class CL_GUI_ALV_TREE, a control-based ALV displaying a column tree in a container to be positioned in a Dynpro.

General Considerations

User experience: ALVGrid and ALVFullscreen are control-based, thus presenting a more enjoyable user interface than the list-based ALVList and ALVHierSeq.

New Developments

- **Rendering object:** For setting header and footer in ALVFullscreen, ALVList, and ALVHierSeq, a new OO-based concept analogous to the layout designer in Web Dynpro is provided. It avoids WRITE statements by the application. Its key advantages are:
    - **Accessibility:** By providing data within semantically enriched objects, the application also ensures that header and footer information is accessible.
    - **Layout:** The layout of header and footer information conforms to SAP standards and is presented in a unified way across applications.

- **ALV object model:** A new OO encapsulation for ALV lists that is for ALVGrid ALVFullscreen and ALVList. Its key points are:
  - **Unified OO Interface:** A single programming model for all ALV tools.
    - **Metadata:** The ALV objects provide a single metadata model for the respective tools. Metadata is encapsulated in a metadata object.
    - **Eventing:** Object oriented eventing model for user interaction.
- **No Data:** in case of empty data sets, it will be possible to set the complete table (including headers) invisible.
- **FullscreenTree:** A control-based ALV displaying a column tree on its own screen. In particular, compared to ALVTree, the same advantage for ALVFullscreen compared to ALVGrid exists.

The following two pages refer to lists only. For use of ALV tree, see below.

Functionality Comparison

While all four ALV tools have much functionality in common, critical differences exist:

- **More than one table:** If you have more then one table to display, you may position them on a Dynpro and use the ALVGrid. However, printing with ALVGrid works on individual instances.
- **Size of data fields:** While the list-based ALVList can display only tables of up to 90 columns, the control-based ALVGrid and ALVFullscreen have a limitation of 128 characters per data cell.
- **Cell merging:** In the control-based ALVGrid and ALVFullscreen, cells of sorted columns with the same value can be merged. This is not possible in the list-based ALVList.
- **Multiline rows:** In the list-based ALVList, you can display a table row in up to three rows at the screen. This is not possible in the control-based ALVGrid and ALVFullscreen.
- **Application interaction:** ALV does not know anything about the rest of your application aside from the data table (see below). Interaction with ALV proceeds via control events and method calls in case of the ALVGrid, via callbacks in case of the other list tools. However, in the new ALV object model, eventing is implemented in a unified way.
- **Leading blanks:** Leading blanks in cell values get lost in the control-based ALVGrid and ALVFullscreen. In general, optimization does not take leading blanks into account.

Working with ALV

- **Data:** While ALV does not know what happens in your applications, you always work with the same data as ALV: the data table is given as a reference to ALV. However, this requires the visibility of the data table to be defined appropriately in case of ALVGrid, as it must continue to exist during processing of ALV grid (beware of locally defined data tables).
- **Fieldcatalog:** The Fieldcatalog serves for describing the data table as it is. That is you cannot change intrinsic properties of a data field (like type, size, etc.) by setting corresponding values in the Fieldcatalog.
- **Print:** The control-based ALVGrid and ALVFullscreen cannot be printed directly. Therefore, printing is done via changing to the ALVList. In particular, ALVGrid is not printed within the Dynpro where it is positioned but can only be printed isolated on its own screen.
- **Column header:** The column headers have important functionality in ALV for sorting, summing, etc., which gets lost if you remove them. Moreover, headers are an important component regarding accessibility (this obstacle will be removed in the future).
  Column headers are single line only. That is, no multi-line headers are possible.
- **Aggregation:** ALV provides a range of aggregation functions (sum, max, count…). However, it is not possible to add application-specific aggregation functions.
- **Callbacks:** Callbacks can be used for arbitrary write statements. However, in this case the application is fully responsible for accessibility. To avoid corresponding accessibility problems the rendering object will be provided. However, it cannot be used for the callbacks before/after_line_output. Therefore it is strongly advised not to use them.
- **Batch processing:** Batch processing is done for the grid via use of the ALVList. In particular, several ALVGrid instances on one Dynpro are processed separately (see above). ALVList itself prints both the data table and the information contained in Callbacks.

Switching from Grid to List

Because of large data volumes it might be necessary to switch to the grid (see above). If you use existing ALV tools, without much programming effort this is only possible for ALVFullscreen. Of course, the above mentioned differences have to be observed. However, in the new ALV object model, simply setting a flag can do this switch.

ALV Tree

The ALV tree is control-based and therefore has similar properties to the ALV Grid. However there are also additional points to consider. The following applies to both ALVTree and FullscreenTree.

- **Size of data fields:** ALV tree has a limitation of 128 characters per data cell.
- **Cell merging:** Cells cannot be merged.
- **Batch:** ALV tree is not batch-input/CATT enabled.
- **Filter:** ALV tree has no filter functionality.
- **Download:** Data download to other applications (e.g. Microsoft Excel) is not possible.

Additional Information

- For the function modules, see also the corresponding R/3 function module documentation.
- More on the ALVGrid can be found in the Reuse Library:
  SAP Basis>Controls>ALV Grid Control. Here you also can find examples.
- For examples on all ALV tools see the reports BCALV_*. Particular aspects are highlighted in reports BCALV_TEST_*.

Critical Points

There are some restrictions that might prevent you from using ALV. The following points have to be checked before migrating your lists to ALV.

- **Mass data:** ALV is not designed for mass data processing but as a display tool and thus keeps all data in memory. More precisely: it is possible to display large volumes of data in ALV, cases of up 100,000 data sets are known in practice. However this uses a corresponding amount of memory; so in cases where data volumes of >50,000 data sets are processed regularly, you should consider carefully the consequences regarding use of resources, in particular as in these cases data processing most times is not done online.
  ALVTree has stricter limitations: it is only designed to display up to 1000 nodes.
- **Deep structures:** ALV requires flat tables, thus you need to flatten existing deep structures in your table. ALV can be used for tables with structured data fields of up to one hierarchy level, however the individual fields have to be defined in the Fieldcatalog appropriately, that is substructures are not detected automatically.

- **Page oriented output:** As ALV is a display tool, page-oriented processing (like carry-over of sums) is not possible.
- **Batch input:** ALV is not batch input-enabled.
- **Relations between columns:** Aside of currencies and units, ALV does not allow you to relate columns, e.g. calculating values of one column from values of other columns.