SAP R/3 Style Guide

Print version (PDF)

Enhancing the SAP R/3 User Experience

Enhancing the SAP R/3 User Experience

Access Additional Information

The purpose of this SAP R/3 Style Guide is to help developing user-friendly software for the SAP R/3 Systems. It is meant to provide state-of-the-art knowledge on user interaction design.

Audience

- Developers working with the ABAP Development Workbench
- User Interface Designers
- Info Developers
- SAP partners and customers
- agencies, such as TÜV, which provide certification in matters of ergonomics
- the public, including professional colleagues from the field of user-centered design

A Brief Note on the History of this Style Guide

This style guide reflects the state of R/3 release 4.6. It includes the restructured and reworked contents of the previous *SAP R/3 Style Guide (Pocket Version and Updates)*. The guidelines have been brought up-to-date, and new norms for the SAP user interface as well as new articles covering current design problems have been integrated; parts of them had been offered as an appendix up to now.

The previous *SAP R/3 Style Guide (Pocket Version and Updates)* is based on the book *Getting Started with the SAP User Interface*. This book is a compressed version of the official *SAP R/3 Style Guide*, which is part of the R/3 documentation. Both style guides reflect the state of R/3 release 3.0 (the official style guide includes some minor updates).

Note that the official SAP R/3 Style Guide has been discontinued; from now on this style guide is the only usability resource for the R/3 system. Since the reference lists are the most frequently requested section of the R/3 Mini Style Guide, they are now in a separate guideline document, the *SAP Reference Lists* to be found on the *SAP Design Guild*.

The **SAP Reference Lists** offer:

- R/3 function names
- Access mechanisms for R/3 functions: F-keys, icons, accelerator keys
- R/3 Terminology Comparison Lists (English - German)
- R/3 menus
- R/3 status icons
- R/3 icons
- R/3 text guidelines (English, German)

Note: If you need to access the previous version *SAP R/3 Style Guide (Pocket Version and Updates)*, please go to *Outdated: Former Guidelines and Papers* in *Resources* on the *SAP Design Guild*.

This guideline can be found in *Resources* on the SAP Design Guild Website (www.sapdesignguild.org).

Enhancing the SAP R/3 User Experience

Access Additional Information

HCI Bibliography: Human-computer Interaction Resources

http://www.hcibib.org/

Books on User Interface Design in General

Beyer, H. & Holtzblatt, K. (1998). Contextual design. San Francisco: Morgan Kaufmann.

Cooper, A. (1995). About face: the essentials of user interface design. IDG Books Worldwide.

Eberleh, E., Oberquelle, H. & Oppermann, R. (Eds.) (1994). Einführung in die Software-Ergonomie. Berlin, New York: de Gruyter.

Fowler, S. L. & Stanwick, V. R. (1995). The GUI style guide. Boston: AP Professional.

Helander, M., Landauer, Th.K. & Prabhu, P.V. (Eds.) (1997). Handbook of human-computer-interaction. Second, completely revised edition. Amsterdam: North-Holland.

Hackos, J.T. & Redish, J.C. (1998). User and task analysis for interface design. New York, NY: John Wiley.

Shneiderman, B. (1987). Designing the user interface. Reading MA: Addison Wesley.

Wandmacher, J. (1993). Software-Ergonomie. Berlin: de Gruyter.

Zeidler, A. & Zellner, R. (1994). Software-Ergonomie. München: Oldenbourg.
Style Guides

Style Guides in Particular

Apple Computer, Inc. (1992). Macintosh human interface guidelines: The Apple desktop interface. Reading, MA: Addison-Wesley.

IBM Corporation (1992). Object-oriented interface design: IBM common user access guidelines. QUE.

Microsoft Corporation (1995). The Windows interface guidelines for software design. Microsoft Press.

Open Software Foundation (1990). OSF/Motif Style Guide. Rev.1.2. Englewood Cliffs, NJ: Prentice Hall.

Sun Microsystems, Inc.(1990). Open look: Graphical user interface application style guidelines. Reading, MA: Addison-Wesley.

Journals

Communications of the ACM New York, NY: Association for Computing Machinery. (http://www.acm.org/cacm/)

Human-computer interaction. A journal of theoretical, empirical, and methodological issues of user science and of system design. Mahwah, NJ: Lawrence Erlbaum. (http://www.erlbaum.com/Journals/journals/HCI/hci.htm; http://www.parc.xerox.com/istl/projects/HCI/)

International journal of human-computer interaction. Mahwah, NJ: Lawrence Erlbaum. (http://www.erlbaum.com/Journals/journals/IJHCI/ijhci.htm)

 top

Source: SAP R/3 Style Guide

Structuring the Application - Screen Changes vs. Screens with Multiple Areas

At the beginning of the design process, you need to decide about the basic type of your application. There are two choices, though mixtures between them are quite common:

- Applications with **screen changes**
  - No or only few different areas on one screen/page
  - Provides simple, sequential navigation
- Applications using **screens with multiple areas**
  - Few or no changes of the main screens/pages, several areas with lot of interaction between them
  - Provides stable context

Until the Enjoy Release, transactions in the SAP System have been characterized by relatively static screen sequences. Typically, users had to enter an application via an entry screen, only to be confronted with series of deeply-nested data entry screens that had to be handled consecutively. See Structure of R/3 System.

Starting with the Enjoy Release, a new design pattern has emerged. For certain types of applications, the sequences have been replaced by so-called single-screen transactions, where all transaction areas can be accessed without changing screens. This design represents a thorough overhaul of the existing applications, and thus requires some new and sometimes unusual redesign techniques.

Although in the history of designing R/3 transactions there has been a change towards single-screen transcations, both paradigms are still valid and both paradigms will be still in use. Which paradigm is the suitable one depends on the tasks and the users of your apllication as well as the structure and volume of the data. The following tables provide criteria for selecting the basic type or structure for an application.

Tasks

Criterion	Screen Changes	Screen with Multiple Areas
Duration of Processing	long	short
Changes of Processed Objects	few	often
Detail View	seldom needed	often needed
Views	only one view at a time needed	views needed in parallel (e.g. overview and details needed in parallel)

Users

Criterion	Screen Changes	Screen with Multiple Areas
Casual User	guidance through wizard, simple rules for screen sequences	only if simple and understandable

Expert	yes, if guidance is useful	complex screens possible/needed

Data

Criterion	Screen Changes	Screen with Multiple Areas
Structure	flat (list)	hierarchical data structure (e.g. overview and details)
Volume	wide tables, large number of fields (forms)	narrow tables, small number of fields (forms)

 top

Source: **SAP R/3 Style Guide**

What is the "New Paradigm"?

Until now, transactions in the SAP System have been characterized by relatively static screen sequences. Typically, users had to enter an application via an entry screen, only to be confronted with a series of deeply-nested data entry screens that had to be handled consecutively.

Starting with ENJOY, a new design pattern has emerged. For certain types of applications, the sequences have been replaced by so-called single-screen transactions, where all transaction areas can be accessed without changing screens. This new design represents a thorough overhaul of the existing applications, and thus requires some new and sometimes unusual redesign techniques.

This approach both improves navigation for the enduser and eliminates the forced sequence of the dialogue across multiple screens. There are still situations, however, in which the processing object has to be selected in a separate screen, such as for complex selections (queries) or situations in which there is no alternative to placing many fields or select options on one screen to determine the processing object.

Goals of the New Design

Combine entry and data screens into one window to simplify navigation and retain the context for the user.

Facilitate switching between *Create*, *Change*, and *Display* (for example, set a "standard" mode according to user authorizations). In some cases, there may be no need to differentiate between *Change* and *Display*.

- *Change*, (*Display*) [select object, then choose function]
  - If possible, let user select object from list
  - Double click chooses user's default function (*Change* or *Display*)
  - Put *Change* and *Display* on context menu
  - Possible option: *Different object* with a dialogue box
- *Create*; *Copy from*
  - Possibilities:
    - Choose function, then select object
    - Select object, then choose function
  - Possible: *Copy from* presents a dialogue box
  - Options: System remembers most recently copied reference(s)
  - Presents most recent reference as default
  - Presents drop down list

Make any object (such as invoices or purchase orders) directly accessible

- User selects objects from list / tree
  - Position on screen: Left side
  - **Tip**: Provide suitable views for object selection (for example, the ten objects last edited)
- User identifies object by entering values in fields
  - If appropriate. Example: User enters document number
  - Position on screen: Top left corner, or across the top
- User searches for object
  - dialogue box with query and ranked list
  - Ranked list as additional view in the tree

Retain the context after saving an object, that is, the user stays on the application screen

What does the "New Paradigm" mean in Detail?

Screen Areas

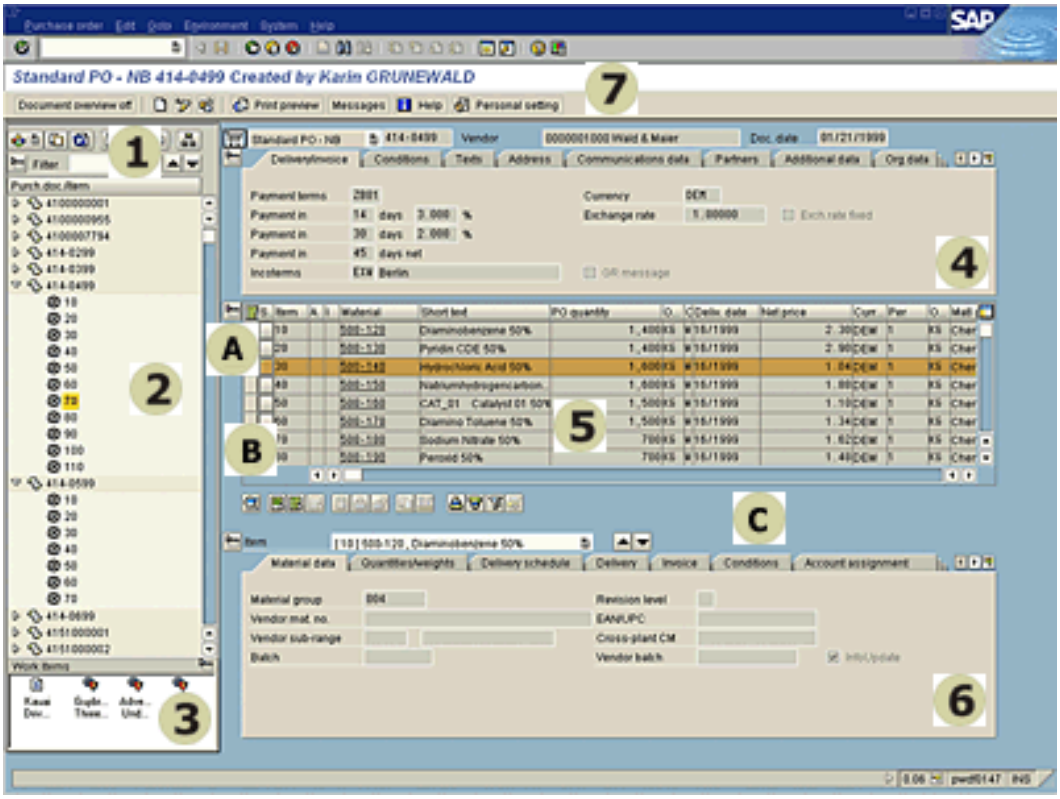


Figure 1: Screen areas within the new design

Click on an area to learn more about it! Click here for a larger version of the screenshot.

The (clickable) image above shows a sample window consisting of seven separate but interrelated areas. Each area serves a specific purpose. The content and naming of these areas depend on the user's current task. The following table provides an explanation of the specific sample functions.

1. Filter, search, and other tree-related functions
2. Object selection, in this example: using a tree
3. Work items: Used for storing "remembered" or "prestored" objects and for displaying workflow items
4. Header area containing data identifying the object
5. Items (for example, of an order)
6. Details for each of the items
7. Application Toolbars

A central goal of the new design is to facilitate user focus on the current task. Depending on the task at hand, users can concentrate on a specific area, while more or less ignoring irrelevant details. The following two methods support this task-oriented user guidance. You can use:

A Expand / Collapse of Areas

B Splitter

C Separating and Connecting the Areas

top , more...

Source: SAP R/3 Style Guide

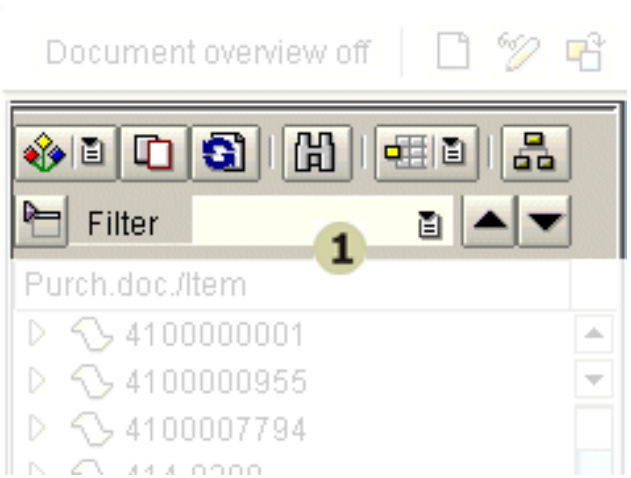
The Screen Areas: A Closer Look

1 2 3 4 5 6 7 A B C

The new single-screen transactions have integrated numerous areas of an SAP transaction into a single window. These areas are described in greater detail below, using a sample transaction.

1 Filter and Search Area

The area for searching and/or filtering data can be used both to reduce the number of entries displayed (for example, in a tree) and to select a specific entry quickly.

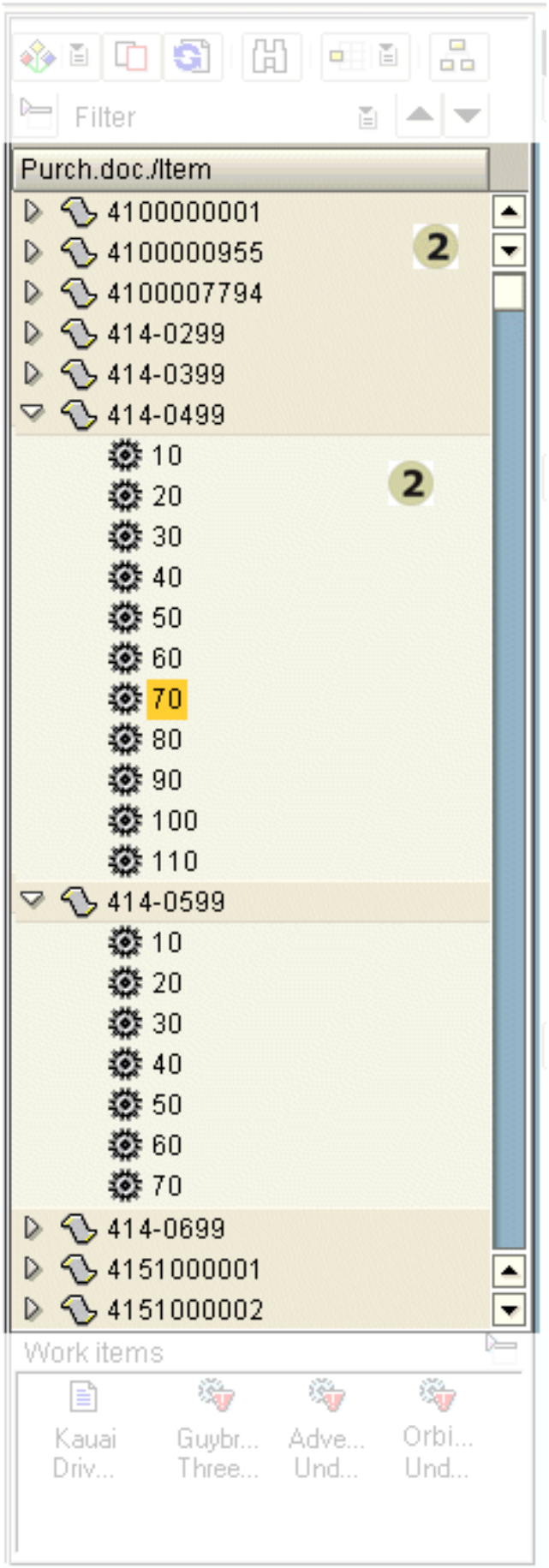


In addition to the filter/search function, this area can contain elements such as the toolbar of an ALV tree.

top, Back...

2 Object Selection

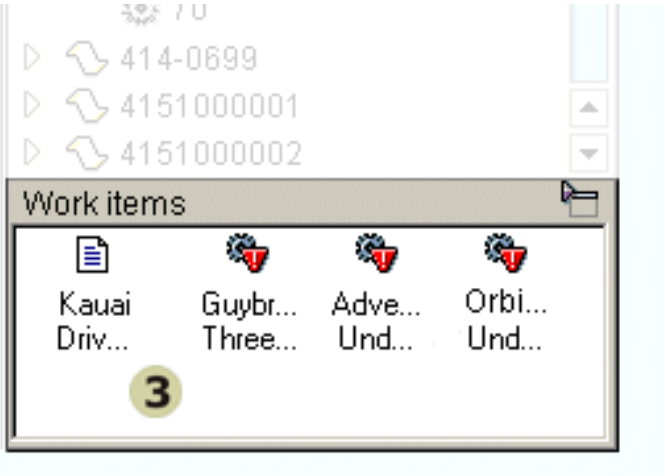
The object selection area is used to find and select entries quickly within a larger quantity of related elements. Depending on the underlying data structure and the user's specific task, either list- or tree-like displays should be used.


Filter
Purch.doc./Item
4100000001
4100000955
4100007794
414-0299
414-0399
414-0499
10
20
30
40
50
60
70
80
90
100
110
414-0599
10
20
30
40
50
60
70
414-0699
4151000001
4151000002
2
2
Work items
Kauai Driv...
Guybr... Three...
Adve... Und...
Orbi... Und...

top, Back...

3 Work Items

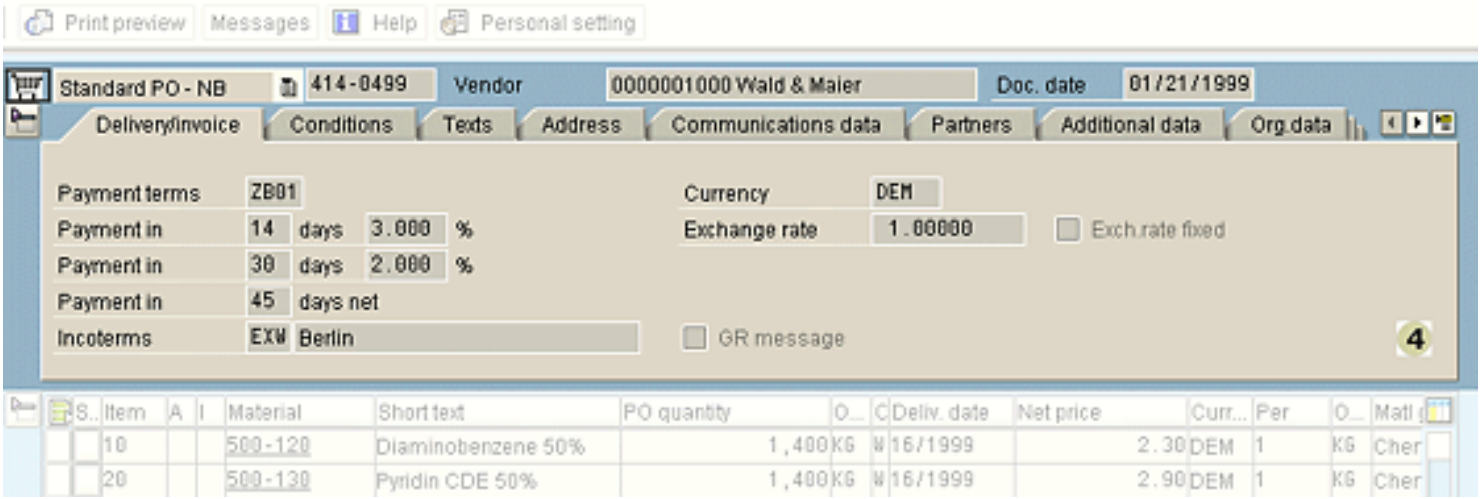
The work item area can contain current or temporarily available objects. These could be "remembered" or "prestored" objects that have to be edited later. It might be reasonable to incorporate items coming directly from a workflow as well.



top, Back...

4 Header Area

The header area contains information that unambiguously identifies an object. Additionally, it may display data from the master record which is relevant to the task. Furthermore, the header area can include entry fields for selecting another object, for example through direct entry of object values (see the following example).

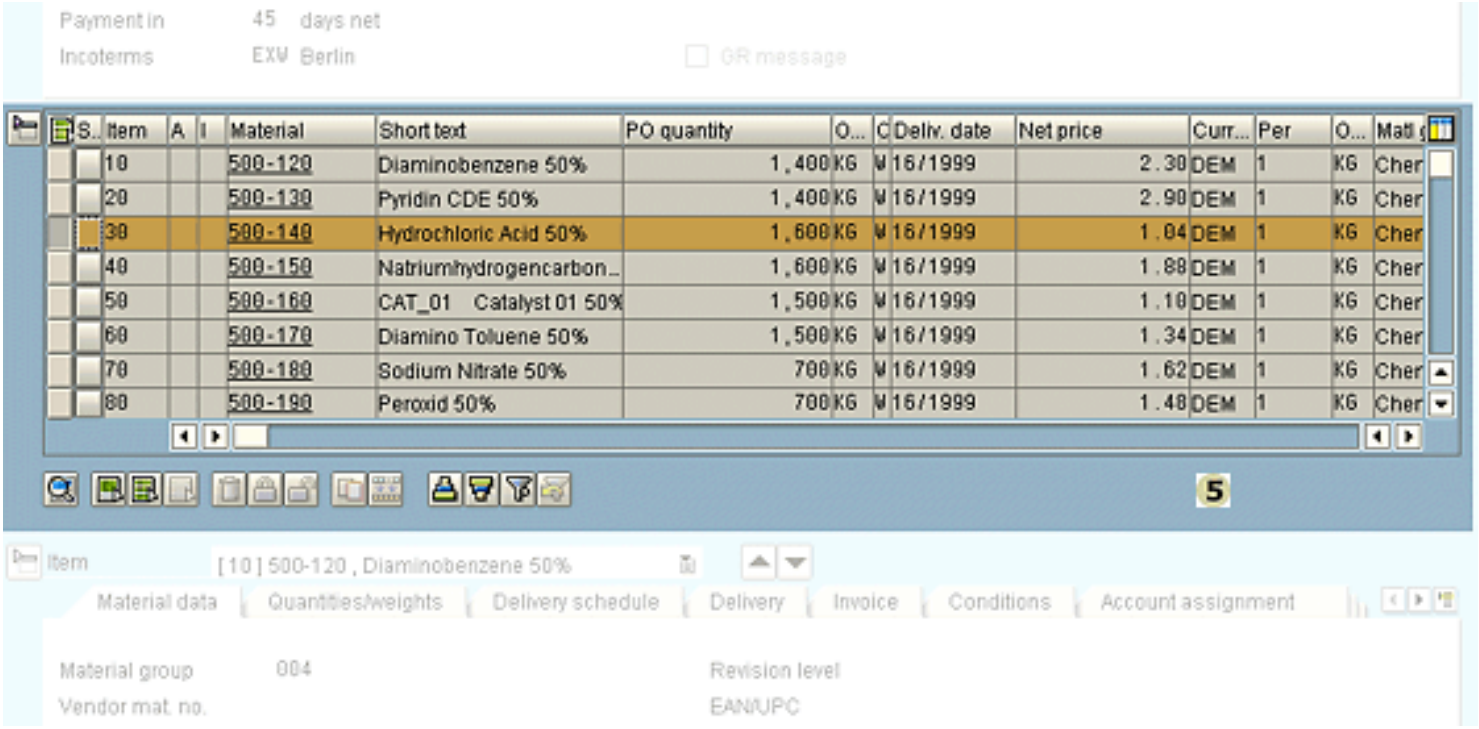


Including these fields might lead to situations that are no longer unambiguous (to the system). For example, the user can change data in the items or detail area and subsequently enter a different object identifier (without saving the changes first). In this case, the application can no longer "decide" whether the item changes are intended for the "old" or the "new" object.

top, Back...

5 Items Area

The items area contains a list of subobjects related to the object specified in the header area. These objects could be the single items belonging to an order, for example. This area aims to provide a compact overview of these single items. This is typically achieved by using a table control or an ALV grid control.

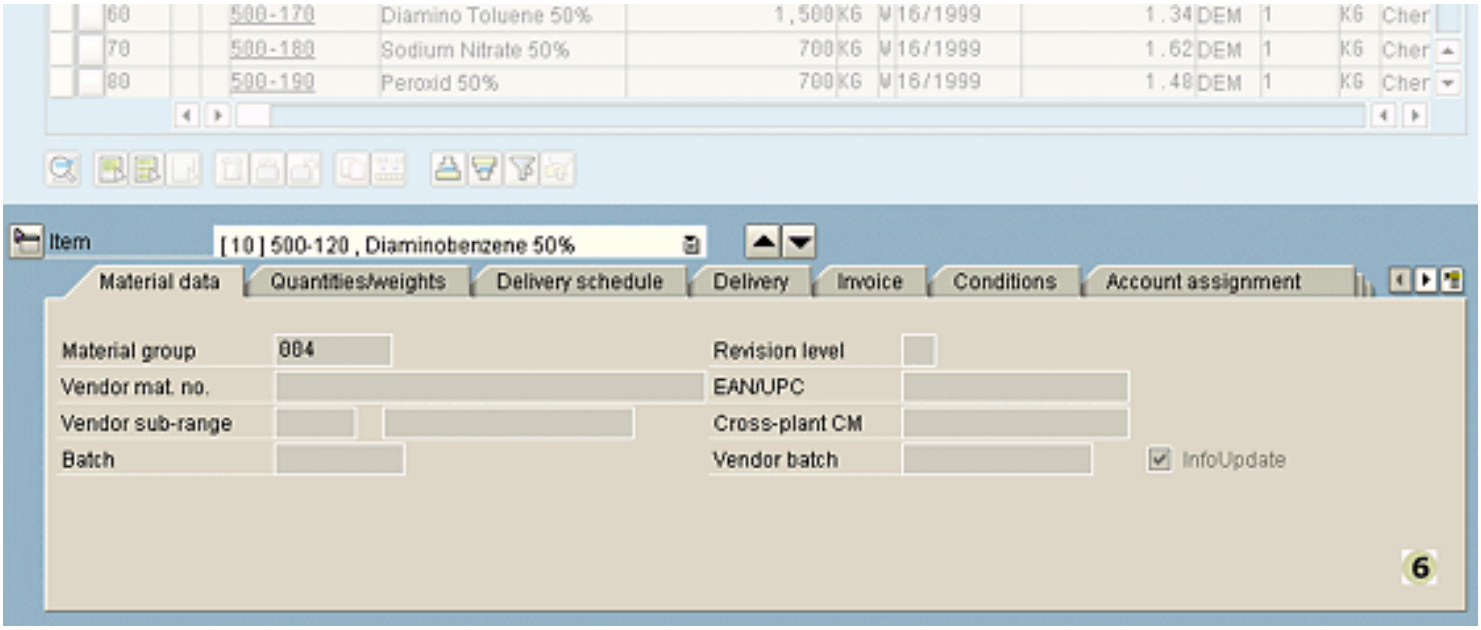


Changing items always causes the data displayed in the detail area to change accordingly. This means that the item and detail areas are always interrelated.

top, Back...

6 Detail Area

The detail area is used to display and change larger amounts of data associated with each item. This area usually has its own buttons for navigating between items. The navigation itself is reflected in the item area.



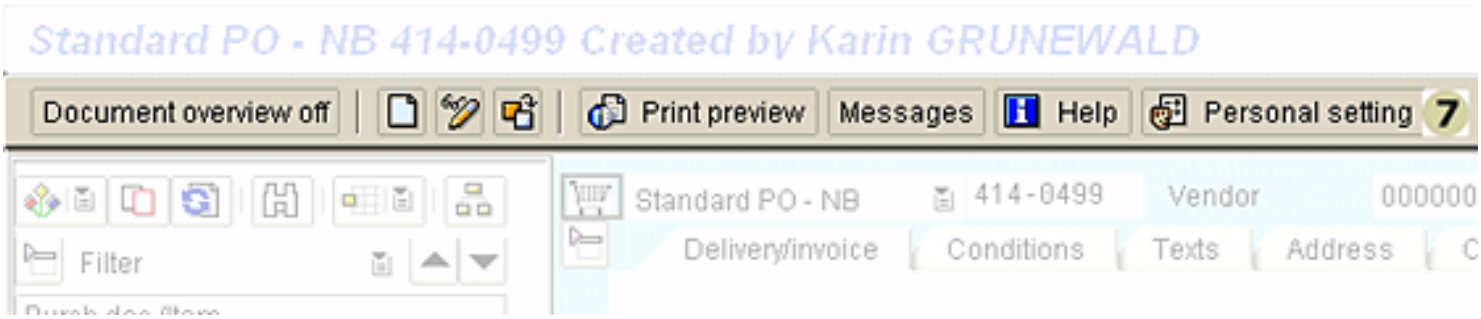
Because the detail and item areas are so tightly coupled, "precedence of navigation" rules must be implemented.

Note: Scrolling does not alter the explicit selection or change the details areas

top, Back...

7 Application Toolbars

As a consequence of the single-screen design, the application toolbar often only contains a small set of functions related to the entire selected object or the entire screen. These could include functions for displaying or hiding the object selection area, or for "creating with reference" or "remembering."



In an application that can do without screen changes, the navigation buttons *Back*, *Exit*, and *Cancel* may be assigned to different functions. The following table provides an overview of the current recommendations:

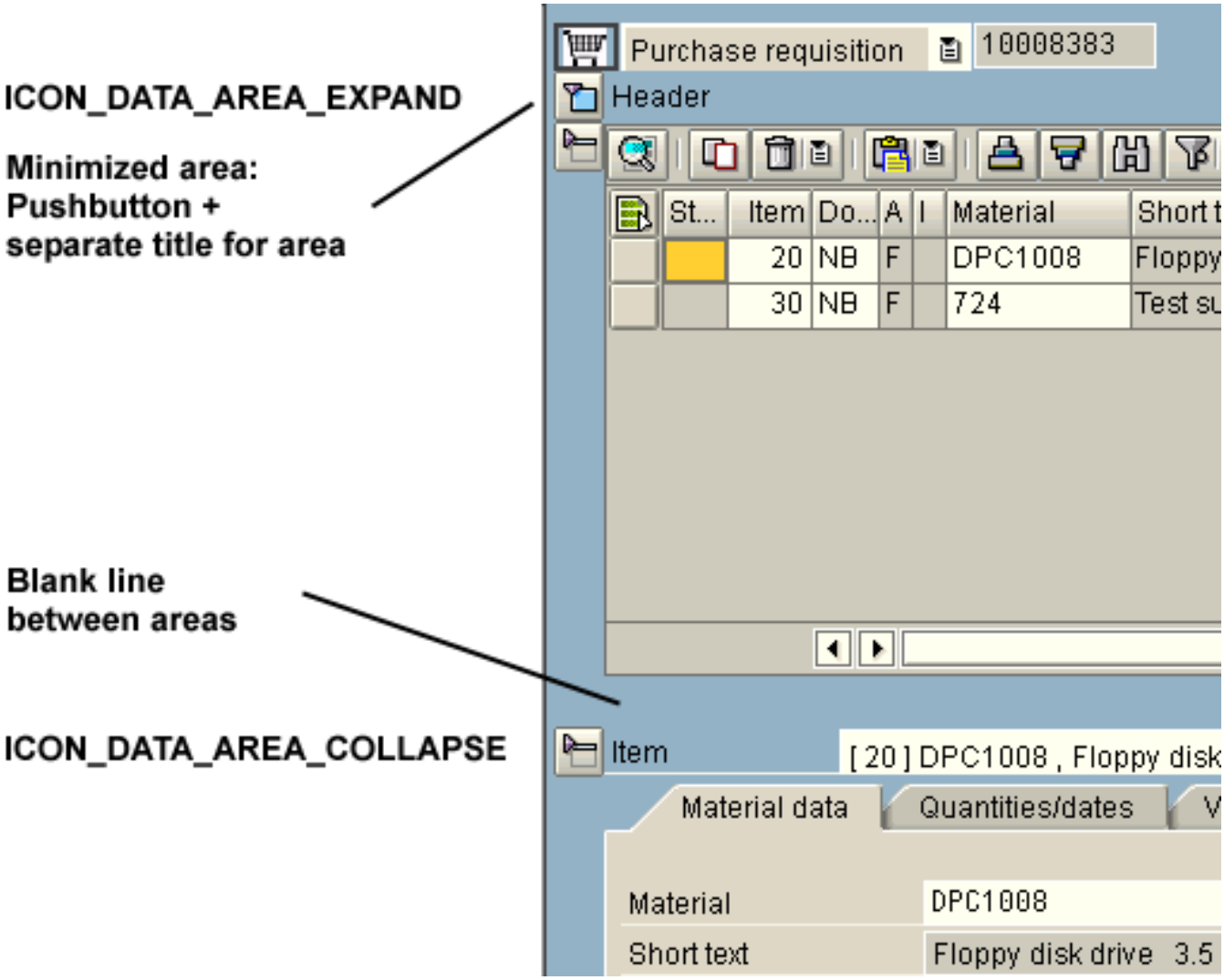
Icon	Name	Meaning
	Back	Exits the application (after a confirmation message) and returns to the next higher level.
	Exit	Exits the application (after a confirmation message) and returns to the application level.
	Cancel	Empties the data entry fields (after a confirmation message) and enables the fields for object selection

Most of the available functions will be placed directly in the appropriate areas and will thus be available to the user where they are most needed.

top, Back...

Expand/Collapse of Areas

Expand and *Collapse* buttons are especially useful for horizontally subdivided areas. For example, at the push of a button you can resize the header area to a single line, yielding more space for the other areas. With this technique, users can choose between different layout combinations reflecting the focus and information need of the current task.



top, Back...

B Splitter

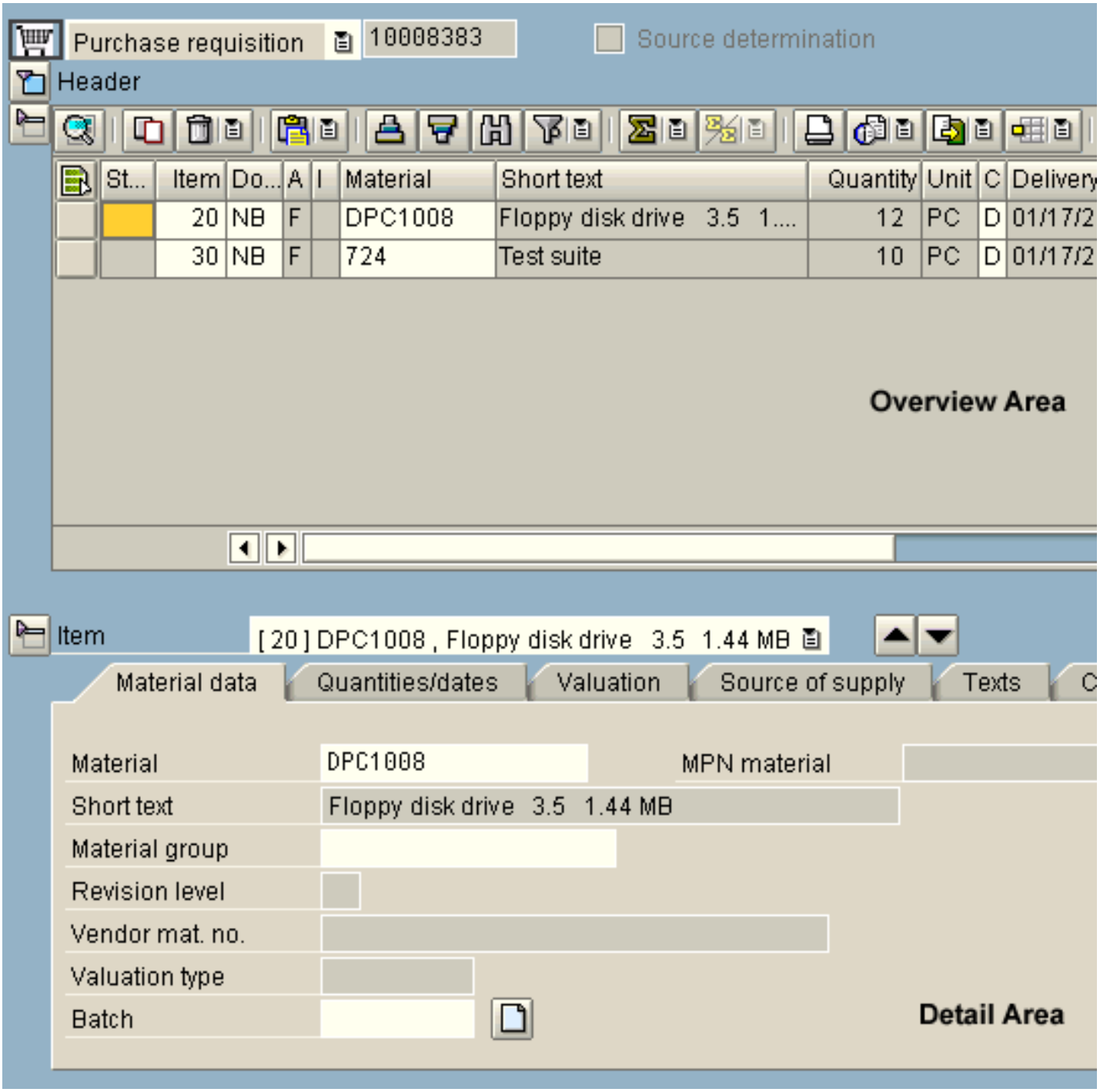
If a control (for example, for object selection, such as a tree or ALV grid control) is "docked" to a given window, splitters can be used for stepless resizing of the area the user is focusing on. A disadvantage of the splitter is that this very narrow area is difficult to handle, especially for older users.

top, Back...

C Separating and Connecting Areas

- Separating the areas (subscreens)
  - Use blank lines between areas
  - Do not put a group box around tabstrip controls or table controls with table-related functions (except to group one of those elements with other elements)
- Connecting the items area and details area
  - *Choose detail* is on F2; page in tab, choose in table
  - *Previous* and *Next* in Details area moves selection mark line-by-line in table control (so does scrolling in Details area)

- Optimize *Resize* behavior of areas
- Scrolling does not alter the explicit selection or change the Details area
- Selected tab page is "remembered"



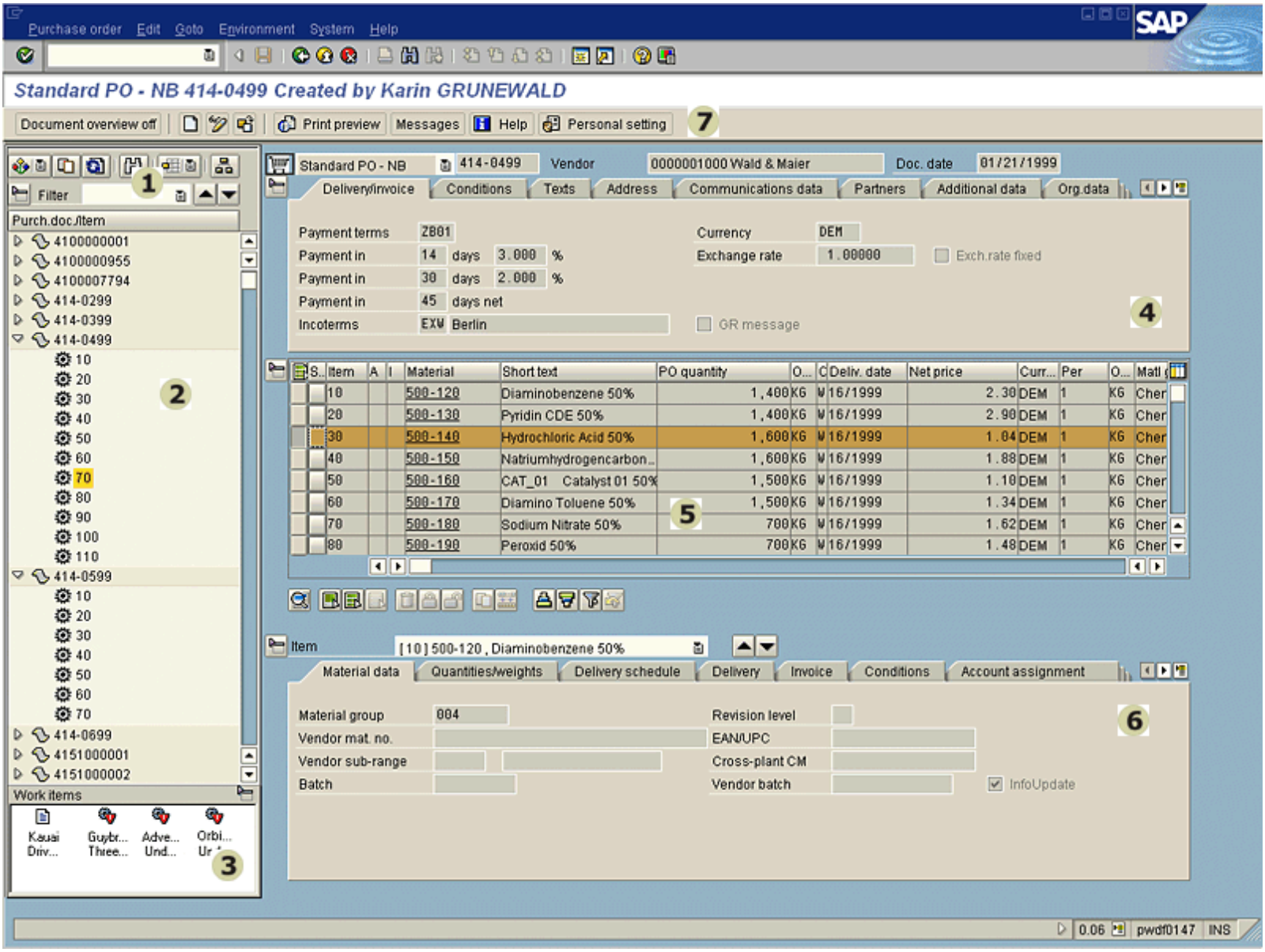
Guidelines for Interaction by Selecting Items in Overview Area

- Where you can select only **one item**
  - When selected explicitly (item selection element on)
    - Initially: Details empty
    - *Previous item, Next item* goes to anywhere in table and selects the destination item
  - When selected implicitly (cursor in row)
    - Detail shown depends on cursor position: Set explicit selection in program
    - *Previous item, Next item* goes to anywhere in table
- Where you can select **multiple items**

    - When selected explicitly (no implicit details)
        - *Previous item, Next item* goes to previous or next selected row in table
- In **details** screen:
    - For context: top line in details:
      Item <Text> <*Previous* pushbutton> <*Next* pushbutton >
    - Icons: Use the triangles pointing up and down
    - Disable the *Previous* or *Next* pushbutton, as appropriate, when first or last item is current
    - You could possibly have a *Create* icon pushbutton in the Details area (Row 1, right); or other pushbuttons, *Delete,* for example. This means users can enter data right in the Details area.

 top , Back...

Source: SAP R/3 Style Guide


Purchase order Edit Goto Environment System Help
SAP
Standard PO - NB 414-0499 Created by Karin GRUNEWALD
Document overview off
Print preview
Messages
Help
Personal setting
7
1
Filter
Purch.doc./item
4100000001
4100000955
4100007794
414-0299
414-0399
414-0499
10
20
30
40
50
60
70
80
90
100
110
414-0599
10
20
30
40
50
60
70
414-0699
4151000001
4151000002
2
Work items
Kauai Driv...
Guybr... Three...
Adve... Und...
Orbi... Ur
3
Standard PO - NB
414-0499
Vendor
0000001000 Wald & Maier
Doc. date
01/21/1999
Delivery/invoice
Conditions
Texts
Address
Communications data
Partners
Additional data
Org.data
Payment terms
Z001
Payment in
14 days 3.000 %
Payment in
30 days 2.000 %
Payment in
45 days net
Incoterms
EXW Berlin
Currency
DEM
Exchange rate
1.00000
Exch.rate fixed
GR message
4
S.. Item A I Material Short text PO quantity O... C Deliv. date Net price Curr... Per O... Matl
10 500-120 Diaminobenzene 50% 1,400 KG W 16/1999 2.30 DEM 1 KG Cher
20 500-130 Pyridin CDE 50% 1,400 KG W 16/1999 2.90 DEM 1 KG Cher
30 500-140 Hydrochloric Acid 50% 1,600 KG W 16/1999 1.04 DEM 1 KG Cher
40 500-150 Natriumhydrogencarbon.. 1,600 KG W 16/1999 1.88 DEM 1 KG Cher
50 500-160 CAT_01 Catalyst 01 50% 1,500 KG W 16/1999 1.10 DEM 1 KG Cher
60 500-170 Diamino Toluene 50% 1,500 KG W 16/1999 1.34 DEM 1 KG Cher
70 500-180 Sodium Nitrate 50% 700 KG W 16/1999 1.62 DEM 1 KG Cher
80 500-190 Peroxid 50% 700 KG W 16/1999 1.48 DEM 1 KG Cher
5
Item
[10] 500-120 , Diaminobenzene 50%
Material data
Quantities/weights
Delivery schedule
Delivery
Invoice
Conditions
Account assignment
Material group
004
Vendor mat. no.
Vendor sub-range
Batch
Revision level
EAN/UPC
Cross-plant CM
Vendor batch
InfoUpdate
6
0.06
pwdf0147
INS

(Re)designing Entry into the Applications

Explicit Selection of the Processing Object from a List / Tree | Direct Input of the Object | Omitting Selection of the Processing Object in a Single-Screen Application | Navigation in Single-Screen Applications

These recommendations apply to the user interface design for **selecting the actual processing object** within an application, e.g. specifying an order number to process an order, specifying a vendor to change the vendor data, etc.

In this solution - which fits the new design paradigm of "single-screen applications" - the processing object and the processing type are chosen or specified in the same screen where the object is subsequently processed.

This approach both improves navigation for the enduser and eliminates the forced sequence of the dialogue across multiple screens. There are still situations, however, in which the processing object has to be selected in a separate screen, such as for complex selections (queries) or situations in which there is no alternative to placing many fields or select options on one screen to determine the processing object.

The Situation for the Enduser

Once the user has chosen the application (transaction), such as "Purchase order", he frequently also has to specify the process object first, or even during the processing of subsequent objects. Different processing types can be used with the processing object, such as the usual "Create", "Change", and "Display", as well as more specific functions like "Delete" or "Lock" the entire object.

The Conventional Solution

Under the conventional solution in the R/3 System, the object for processing is selected in a separate initial screen (primary window). The processing type (Create, Change, Display) is selected previously from the menu level (work area menu). This forces the user into a modal dialogue with screen changes. Changing the subsequent processing object always requires an additional step via the initial screen, while the processing type is changed through the object menu, which also goes via the initial screen.

The Future Solution

New technical features of the R/3 GUI (subscreens, split bar, events, control containers, trees, etc.), as well as increased screen resolutions and a greater default size, allow a more modern solution that

- Does not require an additional screen change to select the processing object
- Allows direct selection of the processing object through a list or tree
- Links the selection of the processing type with object selection

These recommendations take the following requirements into account:

- Different fields must be filled depending on the processing type
- A new object can be created through a reference to an existing object
- Error and save dialogues must work correctly
- The "Cancel" function resets the input data, but remains on the screen

 top

Source: SAP R/3 Style Guide

 Menus

R/3 Menus

The following overview describes the basic hierarchy of the menu bars at the three levels of the R/3 System, the main menu level, the application level, and the task level. The task level itself can have a hierarchical structure, too. Structure the menu bar uniformly within a task to ensure consistency. Therefore, do not remove functions that are not used from the menu bar and the pull-down menus but set inactive only.

Note: The *System* and *Help* menus are always available in the R/3 System. They are appended automatically to the right of the individual menus at the respective level. The same applies to the *Layout* menu, which appears to the right of the menu bar as icon. It allows the user to change local settings like fonts or interface colors.

Main Menu Level

In the menu bar at the main menu level, the R/3 System displays all application areas available. The following application areas are currently available: Office, Logistics, Accounting, Human resources, Information systems, Tools.

Application Level

At the application level, object classes or tasks = transactions of an application area are listed in the menu bar.

Task Level

At the task level, the user edits data. The structure of the menu bar and the corresponding pull-down menus are largely standardized. In the following you find an overview of the individual menu bar options in the defined sequence from left to right, together with a short definition and obligatory or typical pull-down menus.

Keys

Bold = pull-down option: obligatory

Normal = pull-down option: optional

--- = here you should insert separator lines

<Object> (<Objekt>)

Put actions which affect the entire object here; they receive the name of the object that has been selected at the next higher level -- obligatory

1) Initial screens:

- Actions for choosing another object- Aktionen zur Auswahl eines anderen Objekts

Other <object> (Anderes <Objekt>)

Create (Anlegen)

Change (Ändern)

Display (Anzeigen)

Copy from... (Vorlage kopieren)

- Backup actions (Sicherungsaktionen) ------

Save (Sichern)

Hold (Merken)

Generate (Generieren)

- Other actions (for example import/export functions) (Sonstige Aktionen (z.B. Import-/Exportfunktionen)) ------

Print (Drucken)

- Closing actions (Abschlußaktionen) ------

Delete (Löschen)

Exit (Beenden)

2) Data screens:

- Actions for choosing another object (Aktionen zur Auswahl eines anderen Objekts)

Other <object> (Anderes <Objekt>)

Copy from... (Vorlage kopieren)

- Backup action (Sicherungsaktionen)

Save (or Post)Sichern (oder Buchen)

- Other actions (for example import/export functions) (Sonstige Aktionen (z.B. Import-/Exportfunktionen))

Print (Drucken)

- Closing actions (Abschlußaktionen)

Exit (Beenden)

Edit (Bearbeiten)

Put actions which edit the current object component here - obligatory

- Select actions (Markier-Aktionen)

  Select all (Alle markieren)

  Deselect all (Alle Mark. löschen)

  Select block (Block markieren)

  Choose (Auswählen)

- Editing action (Editier-Aktionen) ------

  Cut (Ausschneiden)

  Copy (Kopieren)

  Paste (Einsetzen)

  Move (Verschieben)

- Other actions (Sonstige Aktionen) ------

  Insert line (Zeile einfügen)

  Delete line (Zeile löschen)

  Sort (Sortieren)

  Delete (Löschen)

- Reset actions (Rücknahme-Aktionen) ------

  Cancel (Abbrechen)

Note: If the selection actions appears in a cascading menu at the second level, use the term "Selections" on the first level:

```
Selections -> Select all
              Deselect all
              Select block
```

Menu functions "Other Actions"

You may place here functions which affect an object component of the entire object, for example, the functions *Delete* and *Print*. However, if these functions refer to the object as a whole, include them in the menu *<Object>*.

Goto (Springen)

Put object components of the current object here - obligatory

Overview (Übersicht)

Header (Kopf)

Next <object component> (Nächstes <Teilobjekt>)

Previous <object component> (Voriges <Teilobjekt>)

Other <object component>... (Anderes <Teilobjekt>)

Back **(Zurück)**

Only the function "Back" is an obligatory entry, all other entries are examples.

Extras (Zusätze)

Put functions which supplement the current object/object component and are not permanently required here - optional

View (Sicht)

Put functions which display the current object/object component in different views here - only if required

Settings (Einstellungen)

Put functions which allow the user-specific defaults of parameters here - only if required

Utilities (Hilfsmittel)

Put functions which not only edit the object but are valid for all objects here - only if required

Your Own Additional Menus (Zusätzliche eigene Menüs)

only if required

Environment (Umfeld)

Put tasks or reports of other application areas here - optional

<Application area> (<Arbeitsgebiet>)

Put menu bar options at higher-level application level here - optional

top

Source: SAP R/3 Style Guide

Redesigning Entry into Applications

Explicit Selection of the Processing Object from a List / Tree

The objective here is to allow the user to select the processing object from a list or a tree on the left (as docking control with splitter or as dynpro element). In this approach, the processing objects can be displayed in different views.

To accomplish this, an area is defined on the left side of the window (screen) that allows the processing object to be selected directly from one or more lists or from a tree (object selection area). Note that the docking control does not necessarily have to be used.

The following example illustrates a single-screen transaction. The processing object can be selected from the tree on the left side. An object is created by triggering the "Create" function in the application toolbar and then entering a vendor (not required) in the corresponding fields (which are ready for input) in the initial area to the right.

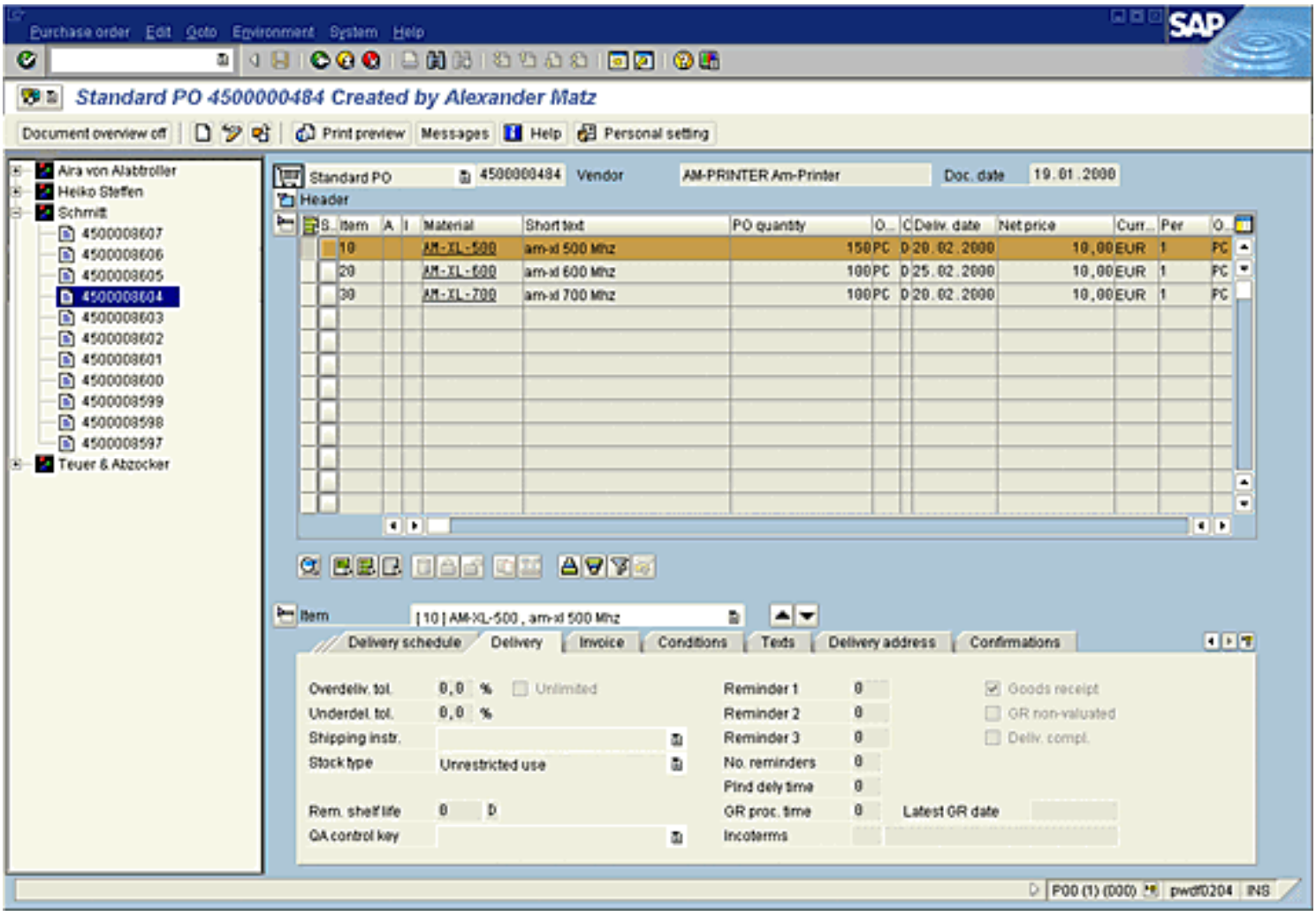


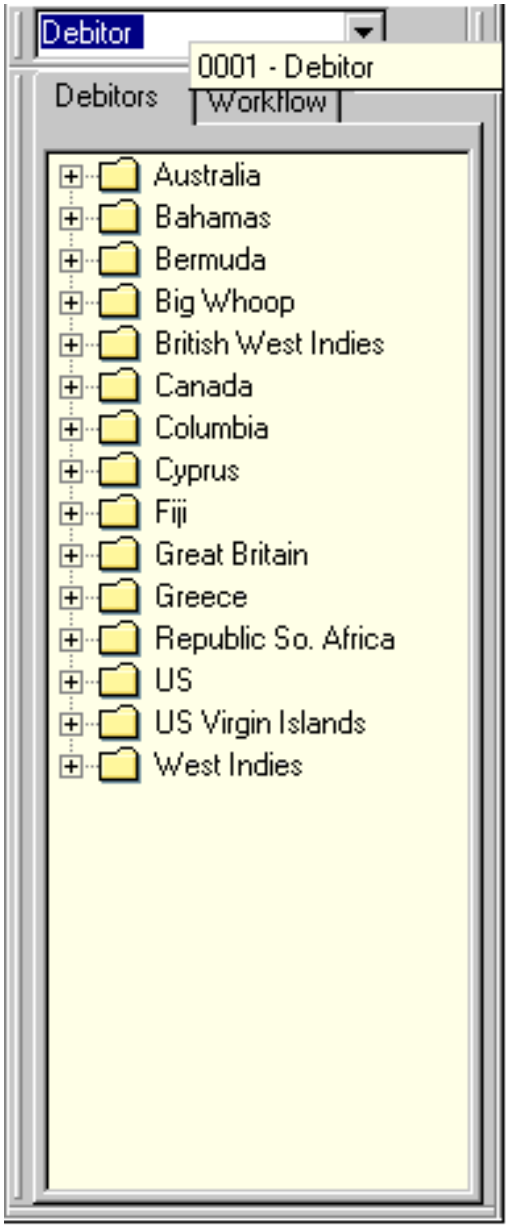
Figure 1: Explicit Selection of the Processing Object from a List / Tree
Click image to enlarge it!

Function Selection

Functions that can be applied to objects in the list, such as "Change", "Display", etc., must appear in the context menu (right mouse button menu) in addition to the "Purchase order" object menu. A default function (see below) must be assigned and appears in bold type in the context menu.

A **double-click** on an object in the list triggers the user-specific default function - that is, "Change" or "Display" (assuming "Create" was not selected). The prior processing status (if any) can be taken into account. For example, if the user is in "Change" mode, then double-clicking on the newly selected object will also open it in "Change" mode, regardless of whether "Change" is the default function or not.

Note: Another area - in initial area above the data input area - can be provided to enter the object directly. In particular, this area may contain key fields that have to be filled when you "Create" an object. It would also be feasible to have the "Create" function - particularly in the "Create with reference..." (icon_create_object) case - display a dialogue window in which the object has to be specified. As shown in the above example, "Create" can appear in the application toolbar.



Setting a Filter to a List

Input fields can be placed above the list/tree area (currently only technically possible if you do not use split bars). The values in the fields filter the objects displayed in the list or tree. After filtering, the list display should retain the context by also displaying the next-highest node. The filter is activated with a "Filter" pushbutton (with a corresponding icon), which appears to the right and below the lowest filter field. Of course, you can also supply a field to enter an interval (compare to selection screens), or use the "More" button to display a dialogue window in which the complete query can be entered (the generated hit list can be used as a new, user-specific view in the list). The values entered in the filter fields should also be persistent for each respective user.

The adjacent diagram illustrates one option for selecting predefined filter conditions from a drop-down listbox plus views on tabstrips.

Finding a String in a List

You can also provide a "Find" function (input field plus pushbutton with the corresponding icon), which allows the user to enter a search string to position the cursor on the corresponding entry.

Different Views of the Objects

To visualize different views of the processing object, the list or tree can be set in a tabstrip. The tab strips allow the user to toggle the view (see diagram). Alternatively, you can also use a drop-down list box to select a view.

Another option is available when the ALV tree is used. This element allows you to create a toolbar with its own defined functions in a new status. Different views can be implemented through a menu button (compare diagram when tree is hidden). It is also possible to have the user generate their own view - through a query for example - which is then dynamically added as a menu button in the menu.

Note: If the docking control is used, you can currently only switch between views by placing an appropriate pushbutton (with text) in the application toolbar and/or defining one pushbutton for each view (the pushbutton of the selected view is displayed in gray).

Template Objects

In addition to the possibility of entering a template in a separate dialogue window, the templates can also be added directly to the left-hand list, or be displayed when the "Create with reference" function is selected.

Last Used or Flagged [Objects]

It is worth considering offering an additional view, "Last used [objects]" or "Flagged [objects]", that contains the last N (e.g. 20) used/flagged objects. These objects are also available for selection. If it does not cause the tree to become too large, you can also add these objects to a new node "Last used [objects]" or "Flagged [objects]".

This list should be independent of the respective user and must currently be programmed manually.

Hiding the Tree / List

The first button in the application toolbar should be the "[View] on/off" function. [View] describes the display in the list/tree, such as "Purchase orders". Pressing the button hides/displays (toggle) the left-hand area (Text button are better than icons).

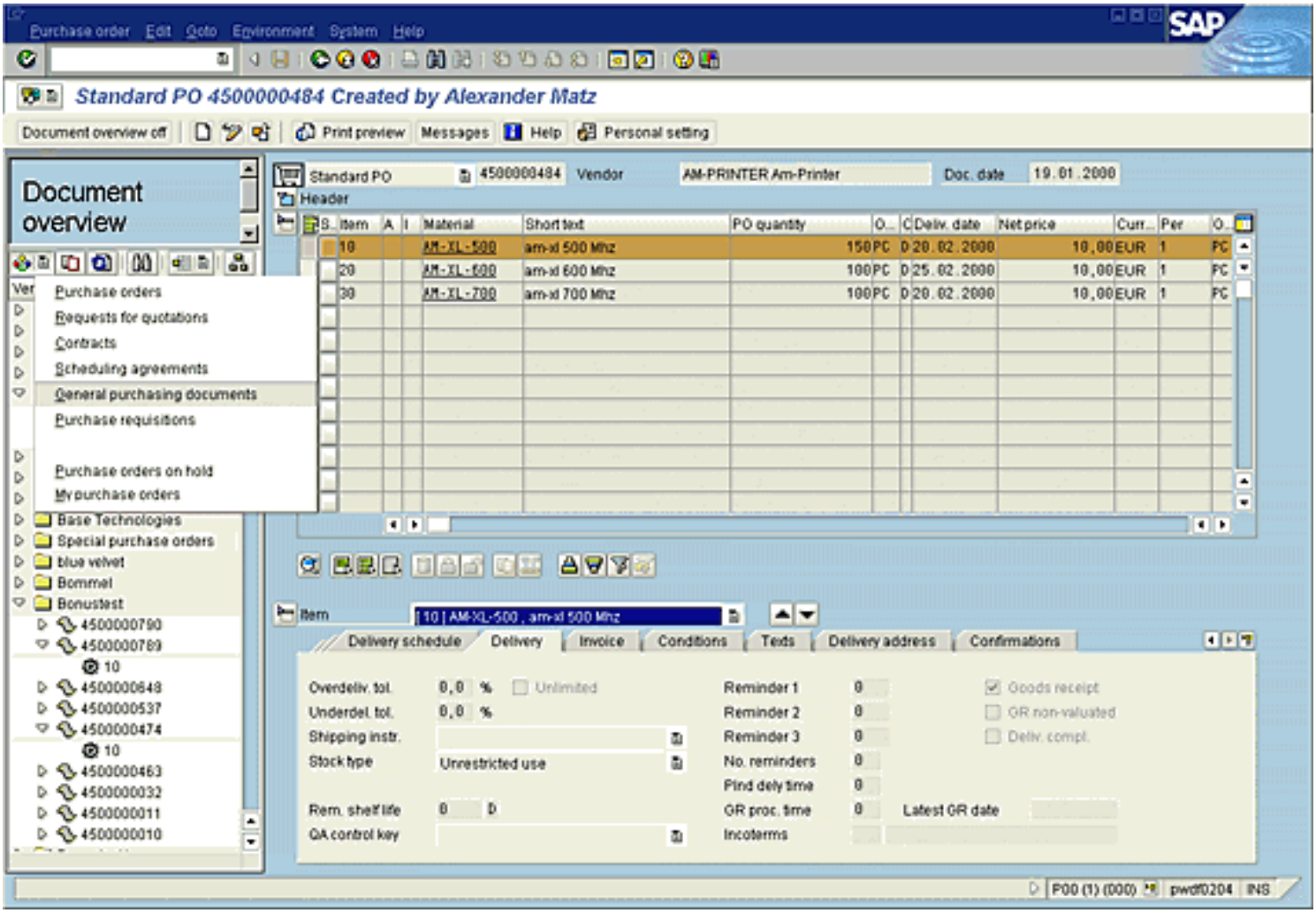


Figure 3: Object Selection with Show/Hide Toggle, ALV Tree, and Different Views
Click image to enlarge it!

 top

Source: **SAP R/3 Style Guide**


Redesigning Entry into Applications

Direct Input of the Object

If the user does not have a chance to select the processing object from a list, the object can be entered in the upper area of the screen (initial area).

The option to select a processing object and a processing type becomes a component of the normal screen where the object is also processed.

The next example illustrates a simple solution for the following case:

- The functions "Create", "Change", and "Display" (in that order) appear to the right of field "Class" and "Class type", next to the input fields. The object-function approach is used here: the class and class type are entered first, and the function is then triggered
- The object can also be created with a template; implementation: pushbutton "Create with reference" (icon!), which appears to the right of the other functions, displays a dialogue window in which the complete object with the template is specified
- Other optional key fields, like "Change number" and "Valid from", can appear in the initial area, but are placed under the functions for selecting the processing type
- No illustration: If "Create" uses other key fields than "Change", then these fields are placed below the others, separated by a blank line. The "Create" function moves down and to the right, accordingly. A different group frame can be placed around these key fields and the "Create" function to emphasize that they belong together.

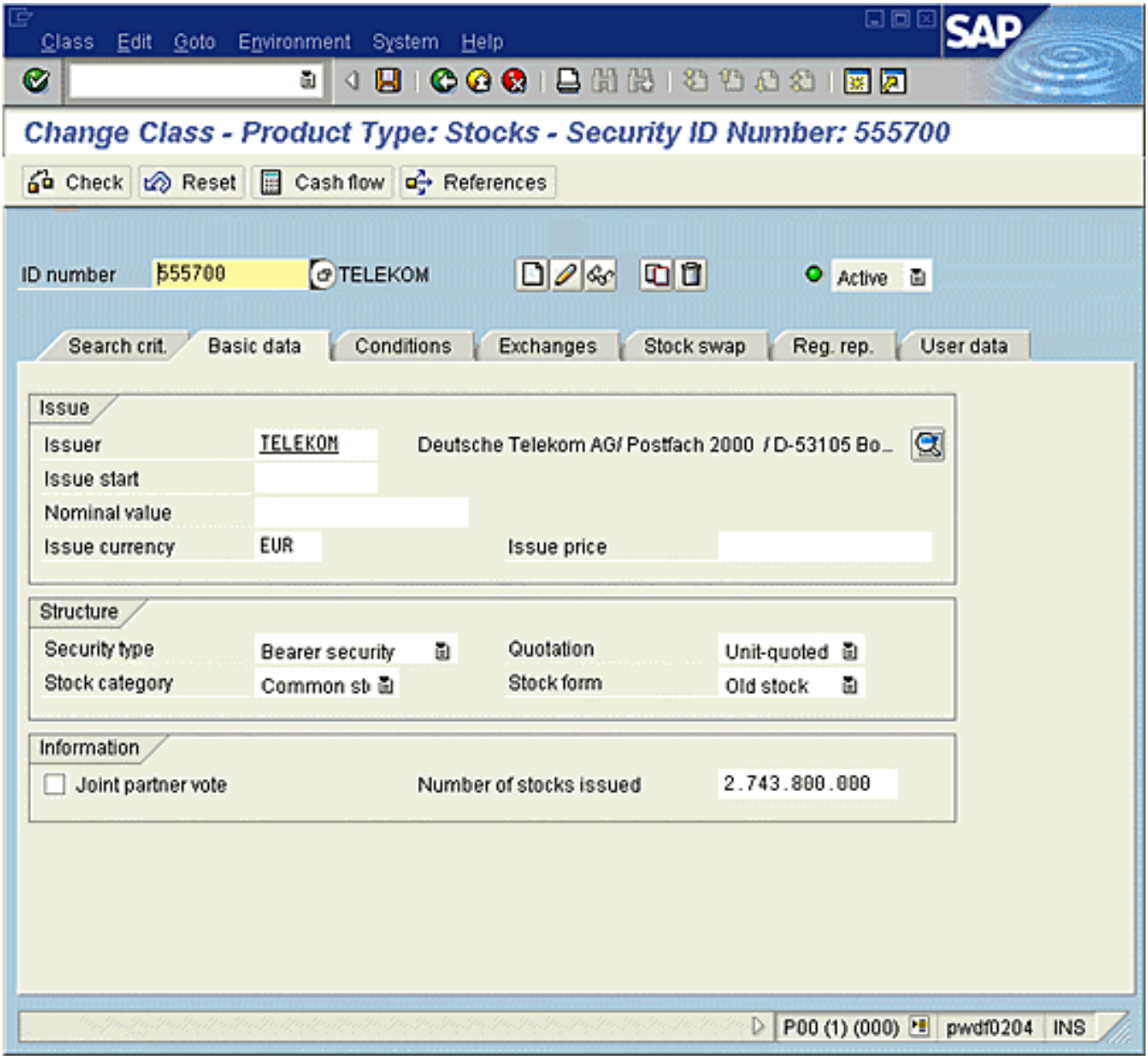


Figure 1: Direct Input of the Object
Click image to enlarge it!

The following example illustrates a somewhat different solution (compare diagram below; this diagram has been modified for the recommendations):

Because directly assigning the "Display", "Change", and "Create" functions would make it unclear which fields have to be filled first (different combinations for key entry are possible), all the functions have been consequently placed in the application toolbar. The user has to know, however, that the "Account group" field must be filled before "Create" can be used. Still, the blank line between company code and account group is a signal that these two fields are not related.

Merging the "Create" and "Create with reference" functions would be possible, and would save two lines. However, the "Create" function would always call a dialogue window, which would force the user into a modal dialogue. Which solution is better depends on how frequently the "Create" function is used.

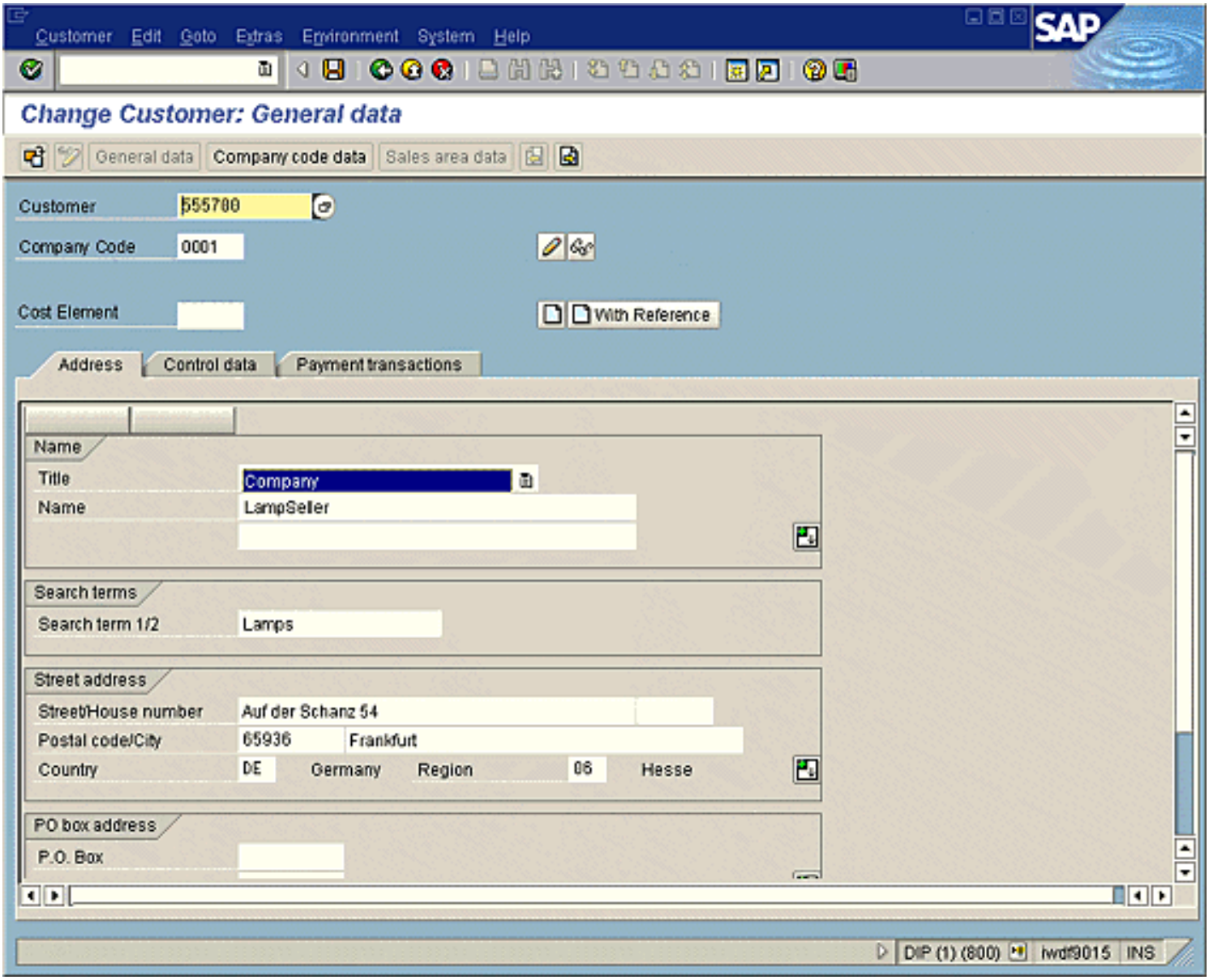


Figure 2: Alternative solution
Click image to enlarge it!

Questions and Answers

- How is the initial area separated from the data input area?
  - A blank line should be used to separate the two areas.
  - When the appropriate Basis elements are available: use a split bar to separate the areas; the individual screen components should be implemented as subscreens within a control container.
- What appears in the lower data input area when no object has been selected yet?
  - Whenever possible, the default data input area with all input elements should be displayed; if the processing object has to be selected first, the input elements are initially inactive. Otherwise, these fields are ready for input. If no other possibilities exist, the area can initially be left blank (blank tab strip)
- Location of focus, and what does the Enter key do?
  - A specific processing type should be assigned the default, depending on the user role. If the user enters the processing object in the upper initial area and presses the Enter key, the default processing type is triggered. Of course, the user can also select other functions.
- How can different authorizations be handled?
  - All processing types that are possible within the application are displayed using pushbuttons. If a user is not authorized to perform certain processing types, the corresponding functions are hidden (at the present time, functions in the application toolbar can either be hidden or set to inactive; when in doubt, please set them to inactive).

- How should the functions be arranged?
  - If the fields to fill are clearly identified, the processing functions should appear to the right of the respective field groups, next to the respective lowest row of a group; otherwise, it is better to place the functions in the application toolbar.

It Is Better to Select a Value than to Enter an Object Key

In the current initial screens, the selection/entry of the processing object is almost always performed by entering (combined) keys, and not through the selection via comprehensible, self-explanatory texts - i.e. the user types in the key.

It is better to allow the user to select the object from a list with self-explanatory texts. The Basis element drop-down listbox can be used for this purpose. If enough space is available on the screen, the objects can also be displayed directly in a list (e.g. table control). If there are seven or less fixed expressions, you can also use the radiobutton element.

Whenever possible, you should assign sensible default values to the fields. If a value is fixed for a certain user, then the corresponding field should be hidden/set to read-only at runtime.

Note

- The processing functions must be assigned to the (appropriate) F keys in the menu painter.
- If a user is not authorized to perform a function, that function is hidden.
- When the focus is in the initial area, the context menu should only contain those functions that affect the initial area (please follow the corresponding recommendation on context menus).
- Assign a sensible default function to the Enter key: The Enter key must be assigned one of the displayed functions. This default assignment should be user-specific (or user role-specific) and defined in customizing.
- In addition to the application name, the title must always also display the selected function (or default function), such as "Change Purchase Order".
- In addition, the application can be programmed such that the last processed object is displayed automatically when the transaction is started.

Remember

If there is no difference between "Create", "Change", or "Display" - that is, there is only one default function that can be executed for the object - then assign it to the Enter key and visualize it with the ICON_OKAY pushbutton.

The diagram below illustrates the input options for the above object with subsequent triggering of the Enter key. Of course, the object can also be selected directly from a tree (ALV tree).

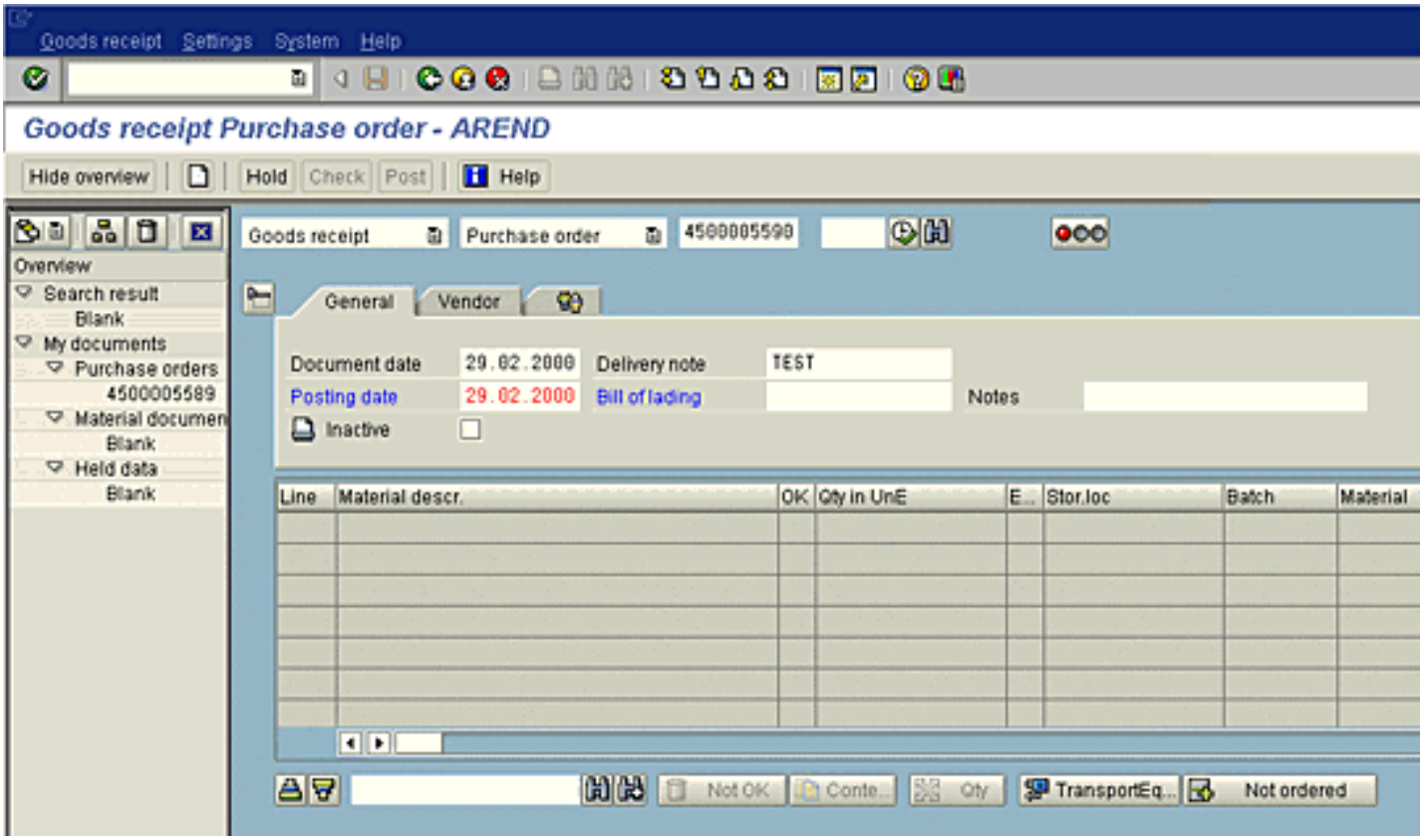


Figure 3: Input options for object with subsequent triggering og the Enter key
Click image to enlarge it!

Entering Additional Details and/or Template/Reference

There are three possible alternatives for implementing and positioning additional input elements that do not fit in the default initial area:

1. Use an additional pushbutton to open a dialogue window
   The "Details..." pushbutton or "[create icon] or [Create with reference... icon]" displays a dialogue window where additional entries can be made.
2. Use a pushbutton to expand the initial area
   A "Details" pushbutton with the "Expand" icon (ICON_Data_Area_Expand) is placed under the input fields (left-aligned). This function adds additional fields - which are surrounded by a group frame - to the initial area. When the expanded area is visible, the pushbutton changes to "Collapse" with the collapse icon (ICON_Data_Area_Collapse).
3. Split the initial area and data input area with a split bar; in this case, the additional fields are initially hidden and must be expanded to be visible
   The additional input area should be surrounded with a group frame and an appropriate header.

You must decide on a case-by-case basis which approach is best. The disadvantage of the third alternative is that the user does not immediately see that the area can be expanded; but the status can be recorded. The second alternative is more explicit for the user and is therefore the preferred solution. The first alternative only makes sense when multiple fields are present but they are only rarely used.

Changing the Processing Object During Processing

If the application has a single key field with internal number assignment (see below) that uniquely identifies the object, the following sequence can be added to the upper right of the screen; it allows a different object to be selected during the processing of the current object:

[Field label] [Input field] [Pushbutton with icon for a different object]

The pushbutton with the "Other object" function can also appear in the application toolbar. When this function is triggered, a dialogue window is displayed in which the user can specify the object to process.

This solution allows the fields in the input area to be set to read-only when an object is already in processing, but the user can still switch to a different object.

top

Source: SAP R/3 Style Guide

Redesigning Entry into Applications

Omitting Selection of the Processing Object in a Single-Screen Application

There are applications in which the enduser sees no explicit difference between selection and processing of the object. In such cases, the screen should be designed in accordance with the enduser's workflow. This means that any required key appears on the screen in the location demanded by the workflow. No differentiation between the different processing types is necessary before processing is started.

In the example below, however, there are four different options for completing processing of the object (parking, flagging, posting). The corresponding functions have been placed in the application toolbar.

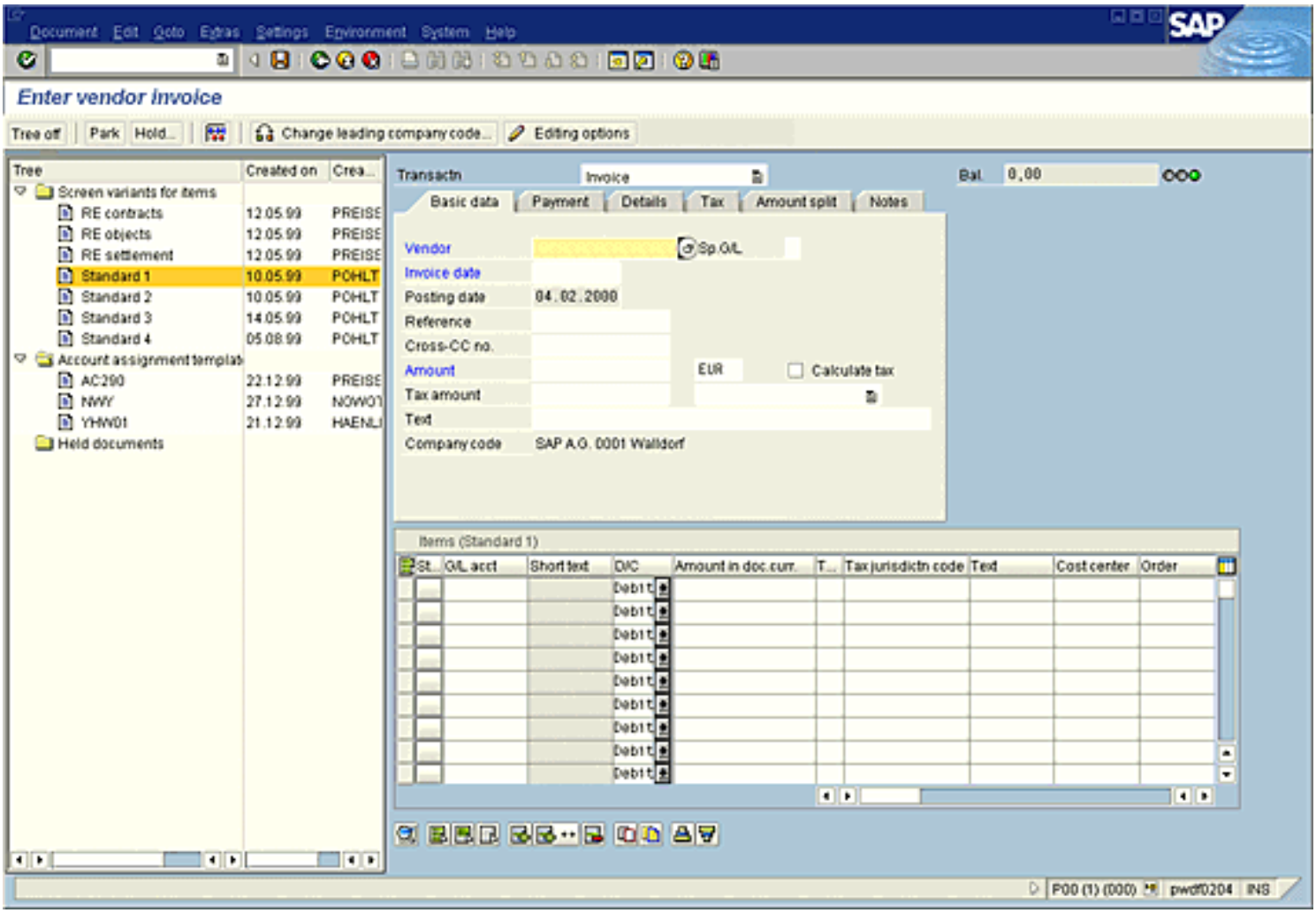


Figure 1: Example for separation between initial area and data input area omitted; key: vendor; select templates from a left-hand object selection area (this diagram has been modified for the recommendations)
Click image to enlarge it!

"Saving" the Processing Object and Changing the Processing Type

The processing object is saved through an explicit trigger of the "Save" function, as before. The contents of the data fields and - where sensible - the fields for selecting the processing object remain. The data fields are initially set to read-only, while any fields for entering the processing object are released (R/W). The application remains on the screen. The user can then enter a different

processing object and has to select

 top

Source: **SAP R/3 Style Guide**

Redesigning Entry into Applications

Navigation in Single-Screen Applications

- Within one screen: *Undo – Cancel – Back*
  - What does user do when there is an error? Must be able to *Cancel* (instead of *Undo*)
  - *Back* retains data (as previously)
  - Confirmation dialog if data may be lost
- From application to desktop to application
  - *Exit* or *Back* exits the application (after a confirmation message) and takes the user to SAP Easy Access.
  - Choose different application when in an application (*Favorite)*
  - Cancel empties the data entry fields (after a confirmation message) and enables the fields for object selection.
  - **Note:** Cancel does not leave the application.

Navigation with F12 and F3

Screen areas in single-screen applications:

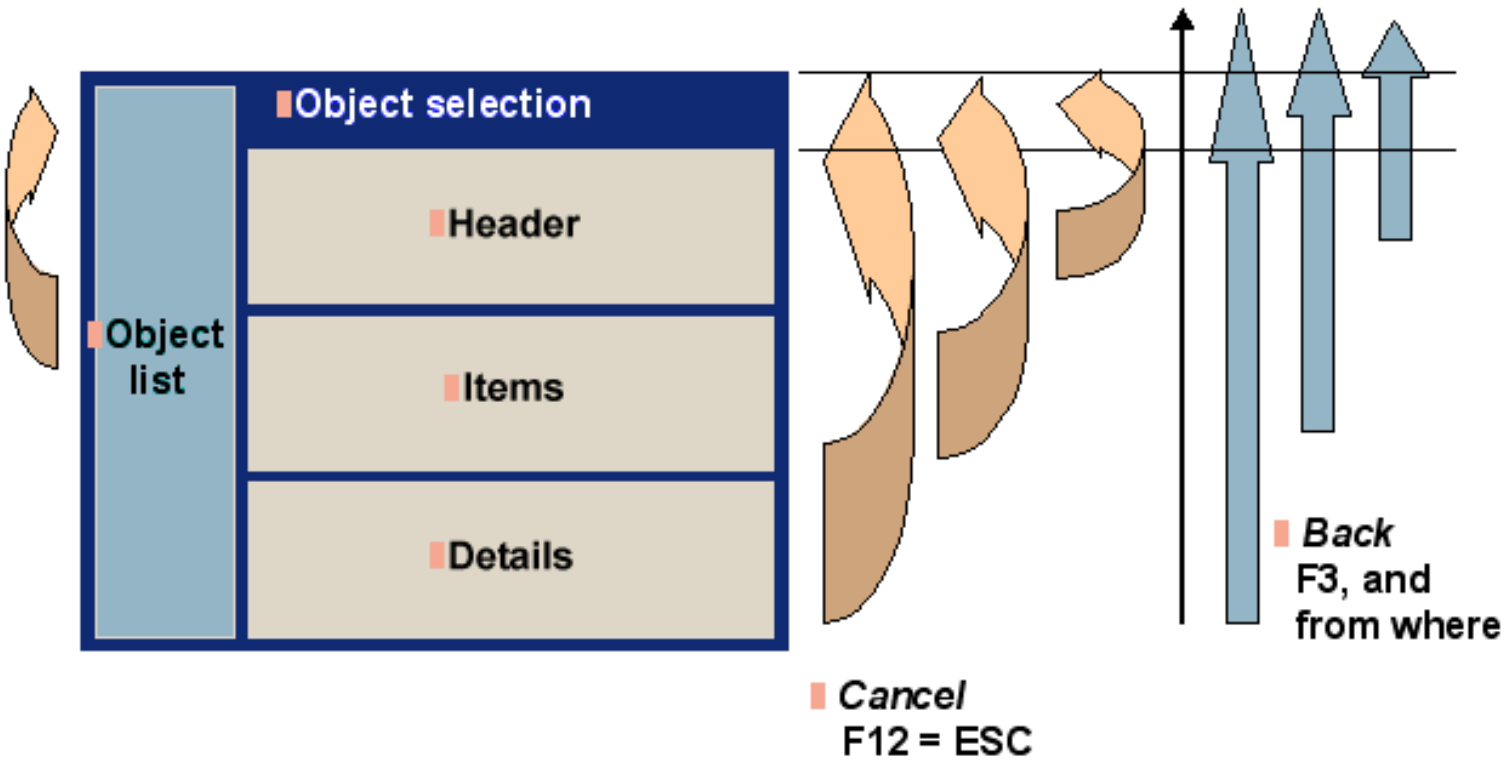


Figure 1: Navigation with F12 and F3

top

Source: **SAP R/3 Style Guide**

Structure of the R/3 System

After logging on to the R/3 System the user sees the SAP Easy Access User Menu. This is the user-specific point of entry into the SAP System. The user menu contains only those items – such as transactions, reports, and Web addresses – you need to perform your daily tasks. If a user menu was defined by your system administrator, it appears when you log on to the system. If he opens the SAP standard menu by choosing "SAP menu" he will have a complete overview of the SAP System.

Regardless at which entry point you start from, you have to open an application in the navigation area. What do these apllications look like? How is the R/3 System structured and what types of screens may he or she expect?

R/3 tasks usually consist of a number of screens that differ in their purpose and look. These different screens are called "screen types". Such types can for example refer to the function that a screen fulfills within a task. They can also refer to its structure, its main processing mode or its level in a hierarchy of data. Giving the screens characteristic names also helps to talk about screens and their special needs and features.

General R/3 Screens

Now we describe and assign names to screens that come up consistently in the R/3 System at task level. This should help you in classifying the screens of your tasks and in assigning proper screen titles to them.

In many cases a simple distinction between "initial screens" and "data screens" will already do for the guidelines presented here.

The following definitions of "Detail Screen", "Data Screen" and "Entry Screen" do only slightly differ. You use either one depending on which aspect you want to place special emphasis.

Initial Screen

First screen of a task from which the user requests the object he or she wants to edit or create. On some initial screens the user may also choose the screens to be displayed or the type of processing. There are certain differences between business tasks and tool tasks, as we shall see later.

Data Screen

Screen within a task where data of a particular type for an object or object component is displayed or changed. Here, the emphasis is on the type or the **contents** of the data. The following screens are subtypes of "data screens".

Entry Screen , Single/Collective Entry Screen

An entry screen is a screen within a task that is mainly used to enter data for an object. "Entry Screen" is the general term for "Single Entry Screen" and "Collective Entry Screen". The latter are screens where the user enters data of one or several object component(s).

On entry screens, the emphasis is on the **design** of the screen: The user should be able to **enter** data as efficiently as possible. However, the user should also be able to change data. Although the title bar says "Entry screen", this also includes the change functionality.

Additional Data Screen

Screen within a task that contains additional data on an object or object component and which is generally called by choosing a

menu option from the "Details" menu (see below) on a data screen.

Detail Screen

Screen within a task where data of one object component is displayed in detailed form.

Overview Screen

Screen within a task that displays object components (mostly items) of an object grouped together in table form. For each object component, there is usually also a detail screen.

Result Screen

Screen of a report that displays the result of the report and which can possibly be used as the basis for further evaluations.

Selection Screen

Screen of a report clearly identified as a prompt on which you specify the criteria to define the information you want in the report.

The Screen Types for Navigation

R/3 tasks are structured hierarchically. The user navigates between these screens to enter, view or edit his or her data. To simplify talking about navigation, some more abstract screen types have been defined.

The Screen Calling the Task, Initial Screen

Usually, the user passes from the system level to the application level where he or she selects a certain task. The screen on the application level is called "the screen calling the task". This type of screen may also be the screen of a task that calls another task.

When the user gets to the desired task, he or she enters the "initial screen" that was described above.

Object Component Screen, Subcomponent of Object Component Screen

From the initial screen the user may pass a sequence of data screens. These screens may show object components ("object component screen") or may even branch to subcomponents of the object components ("subcomponent of an object component screen"). The object components may be freely accessible, or the user has to pass them in a predefined sequence. They may also be accessed via an "object component list screen", described below.

Object Component List Screen

The "object component list screen" is a kind of overview screen, showing a list of all object components. Object component list screens may be nested hierarchically, if the data are very complex.

 top

Source: **SAP R/3 Style Guide**

Moving around in the R/3 System

Comparison of Exit, Back and Cancel | Navigation between Screens

Inspite of the "New Paradigm" with it's navigation in single screen applications, for different reasons the user has to move from one screen to another. This moving between screens is called moving around or navigation. As in the real world, the user needs some kind of orientation and guidance while moving around the SAP System. Therefore it is important to use the means for navigation correctly. You use screen titles to tell the user where he or she is and what he or she is doing. Use standardized navigation functions to provide predefined ways of moving around in the SAP System

We split the discussion into two parts. First, we discuss some general issues and the elementary navigation functions *Back*, *Exit*, and *Cancel*. These functions are shown in the standard toolbar and are active throughout the system. They provide the standard means for exiting a task - may it be a normal or an abnormal termination - and going back to higher or safe levels. Then we will go into general navigational issues and show how users should be able to move between typical R/3 screens.

General Issues

There are different levels in the R/3 System, lots of types of data screens, too, and as a complement, a bunch of navigation functions. Therefore, it is important that you know, when to use which function and which paths between screens are to follow. We shed some light on these issues in this and in the following chapter.

Overview of Elementary Navigational Paths and Functions

Let us start with a simple graphic. It does not by far tell the whole story. However, it gives you a first impression on navigation in the R/3 System.

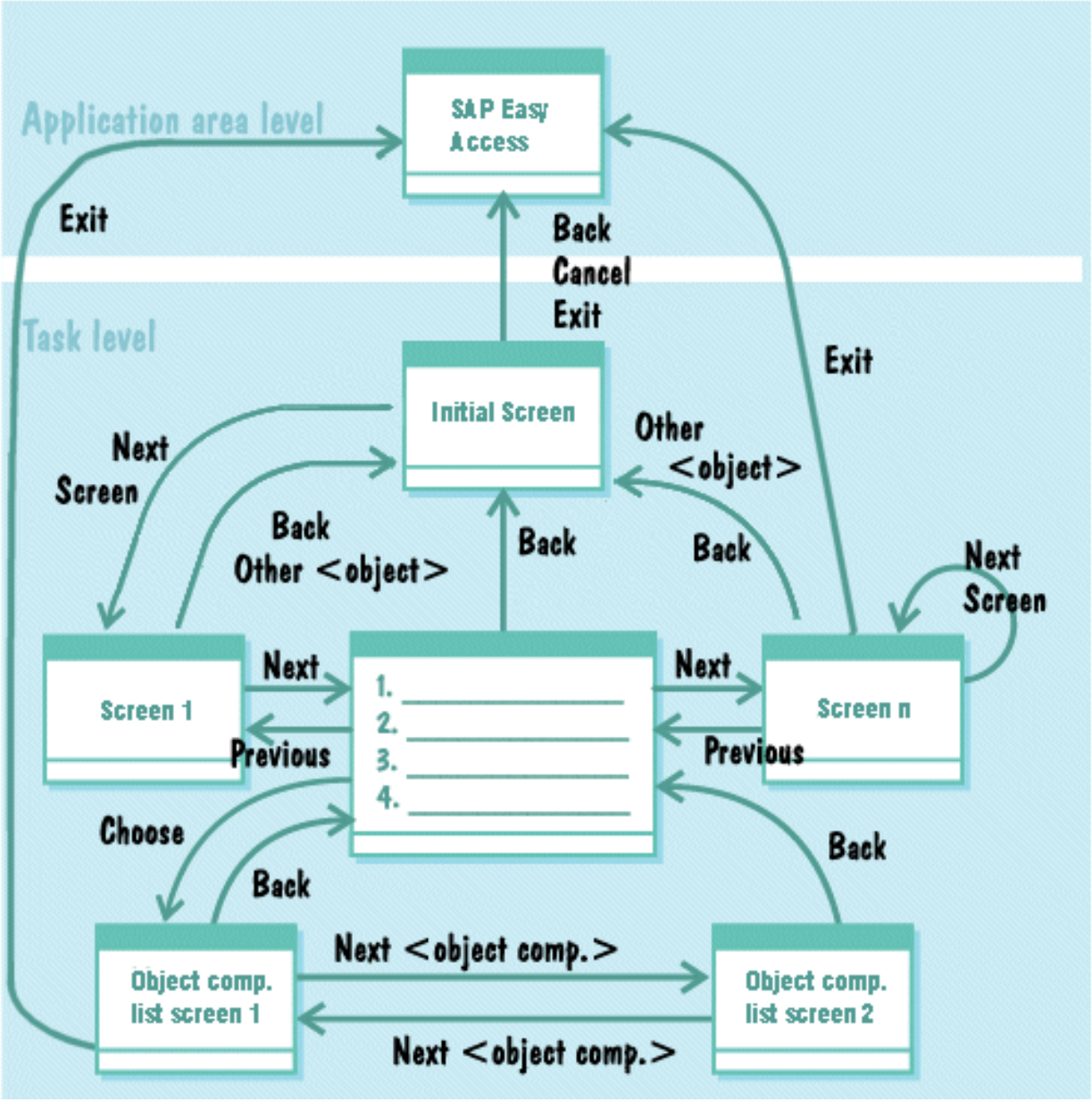


Figure 1: Overview over the most important navigational paths in the R/3 System

Figure 1 shows the three levels of the R/3 System and some typical R/3 screens (shown as rectangles). The labeled arcs between screens on the same or on different levels show some of the primary options for moving around.

Additional options are often available as well. In actual tasks, the options not shown can be even more useful than those shown in the graphic. You should therefore refer to the corresponding functions and look for further options.

The following overview classifies the R/3 navigation functions according to what they are doing:

- Elementary navigation functions: the most important functions for back navigation are the functions *Back*, *Exit* and *Cancel*; for forward moves it is *OK (Continue)*.
- Navigation in screen sequences: *First screen*, *Previous screen*, *Next screen*, *Last screen*, *Previous <object component>*, and *Next <object component>*.
- Navigation in hierarchies: *<Object component> up/down/left/right*.
- Navigation to objects or object components: *Other <object>*, *Other <object component>*, *Choose*, and *Open*.

Navigation Functions in the Standard Toolbar

Some of the most often used navigation functions appear in the standard toolbar. These are the already mentioned functions *Back*, *Cancel*, *Exit*, as well as the scroll functions *First page*, *Previous page*, *Next page* and *Last page*. The user can access and execute these functions easily with the mouse.

Figure 2: Navigation functions in the standard toolbar

top

Source: SAP R/3 Style Guide


Moving around in the R/3 System

Comparison of Exit, Back and Cancel

To clarify the differences between the three basic navigation functions *Back*, *Exit* and *Cancel* let us start with simple examples.

Example 1: Exit

The user has been working on a task and has finished entering data. She wants to leave the application and move over to a different task. In this case, she uses the function *Exit*. This function reminds her to save her entered data (if she had not already done this) and then returns her to the application level.

Example 2: Cancel

The user has been working on a task and has already entered some data. When she moves to the next screen she realized that she has to enter data which she does not know and has to ask for. Therefore, she decides to cancel the editing and get the required information. This time she chooses the function *Cancel*. *Cancel* informs her that she will lose all her data. She may, however, cancel her canceling and go on with the editing. This time, going on with editing makes no sense. So she accepts losing her data and is transferred to the initial screen of her application (note: this is an example, the actual jump target of *Cancel* depends on the context).

Other examples for uses of *Cancel* are error conditions where inconsistent data have been entered which cannot be saved or dialogue boxes that have been erroneously initiated.

Example 3: Back

The name of the function *Back* sounds simple, but this function may not quite be what you and the users expect.

The user edits a document that consists of header data and some more groups of data. All groups have to be entered on separate screens. When the task requires that all groups of data have to be entered, it is useful to put the data screens in a fixed sequence. This time, however, the user wants to access only some of the screens. She does so from a main screen via the *Goto* menu. After finishing data entry for a group she returns to the main screen via the function *Back*. As with *Exit*, *Back* allows her to save the entered data, if she had not already done this.

It is important to note that the function *Back* is not to be used in a fixed screen sequence, as many users would expect. Here, you use the functions *Next screen* and *Previous screen* for moving between screens! *Back*, however, always moves "up" in an application hierarchy. Sometimes this is the screen where the user came from, but not always!

Another problem for users may be that in some cases all three functions return them to the same screen, for instance to the initial screen or to the application level. So, they may come to believe that these functions all do the same. This, however, is not the case. You as developer can help users by implementing these functions correctly!

The three functions *Exit*, *Cancel* and *Back* serve different purposes and have quite a different behavior with respect to where the user is transferred and whether he or she may save the entered data or loses them.

Graphical Comparison

We now move on to a more technical comparison of the three navigation functions *Exit*, *Cancel* and *Back*. First we compare the processing within the three navigation functions, especially with respect to when error checks are done and when safety prompts

appear.

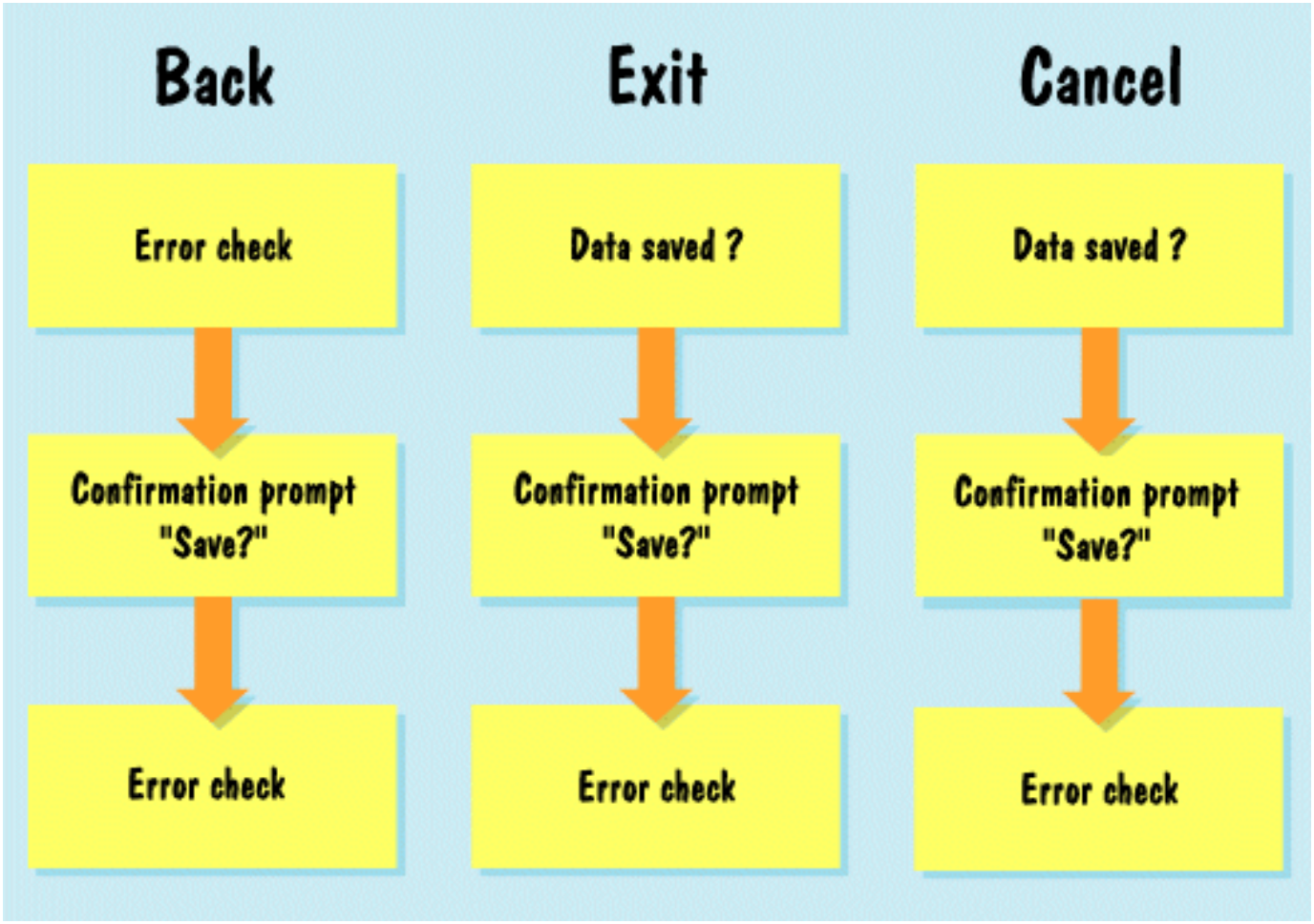


Figure 1: Graphical comparison of Back, Exit and Cancel

Our graphics reveal that with *Back* and *Exit* the error check is performed at different events: With *Back*, the check is performed at the beginning of the processing, whereas with *Exit* it is performed at the end. With *Cancel*, by contrast, no check is performed at all. In addition, the safety prompt with *Cancel* does not provide the chance to save the data, as data cannot be saved with *Cancel*. Therefore, it informs the user about the possible loss of data.

Table Comparison

The following table summarizes the most important information on the three navigation functions *Back*, *Exit* and *Cancel* that you as developer need. Among others, it aids you in designing the safety prompts.

Criterion	Back (*Zurück*	Exit (*Beenden*)	Cancel (*Abbrechen*
effect	exits object component	exits task (application)	exits processing
jump target	returns to previous level	screen calling the task	if possible, a close and safe screen
save option (dialogue window)	Yes	Yes	No
checks	Yes	Yes	No

sequence of save/check	checks first, save option later	save option first, checks later	---
example title of dialogue window	Back	Exit G/L document	Cancel editing
diagnosis text in dialogue window	No diagnosis text	No diagnosis text	Unsaved data will be lost (e.g.)
example text in dialogue window	Do you want to post the held document? **or** Do you want to post the held document 4712?	Do you want to post the held document? **or** Do you want to post the held document 4712?	Unsaved data will be lost. --- blank line --- Do you want to cancel the editing? (1st line is diagnosis text)
pushbuttons in dialogue window (default answer underlined)	Yes, No, Cancel	Yes, No, Cancel	Yes, No
function module	popup_to_confirm_ step **or** popup_to_confirm_ with_ value	popup_to_confirm_ step **or** popup_to_confirm_ with_ value	popup_to_confirm_ loss_of_ data
action menu	*Goto* (*Springen*)	*<object>* (*<Objekt>*)	*Edit* (*Bearbeiten*)
F-key	F3 (V3)	F15 (V15)	F12 (V12)

Table 1: Summary of the three navigation functions Back, Exit and Cancel

top

Source: **SAP R/3 Style Guide**

 Moving around in the R/3 System

Navigation between Screens

This chapter presents the "whole story" of navigation between hierarchically organized screens within a task. For each basic screen type we present the options for moving around. There are from-options to hierarchy levels above and below and within the same hierarchy level. There may also be to-options, that is, navigation options to the current screen.

Navigation: The Screen Calling the Task

From: Moving to Subordinate Hierarchy Levels

- By choosing a pull-down menu item, a function key, a pushbutton, or a fastpath.

To: Moving to "the Screen Calling the Task"

- Relative branches to "the screen calling the task".
- *Exit* always jumps to "the screen calling the task".

Navigation: The Initial Screen

From: Moving to Superordinate Hierarchy Levels

- *Exit* , Back and Cancel jump to "the screen calling the task" (exits the editing).

From: Moving to Subordinate Hierarchy Levels

- **Unrestricted access**: If the object components are contents-related groups of data and can be defined without restrictions (see below), you can define freely navigation options for selecting a group.
  Branches that cannot be carried out because they are object-dependent result in an error message. *Continue* can carry out a default branch.
  Try to provide pushbuttons in the application toolbar for accessing the views. If too many different views are possible, offer the navigation options used less often only for selection via pull-down menus.
- **Predefined navigation sequence**: If the next screen is predefined, for example, the first screen of a sequence of screens, the user can get there by using *Continue*. However, avoid predefined screen sequences in R/3!

To: Moving to the Initial Screen

- Relative jumps to the initial screen.
- *Other <object>* always goes to the initial screen.

Note: The standard functions *Create*, *Display*, *Change*, *Edit*, *Single processing*, *Collective processing*, *Enter*, *Single entry* and *Collective entry* always navigate to the initial screen as well. In the following, this is not pointed out separately.

Navigation: The Object Component List Screen

From: Moving to Superordinate Hierarchy Levels

- *Other <object>* jumps to the initial screen.
- *Back*: If the object component list screen was directly accessed from the initial screen *Back* also returns to the initial screen.
- *Exit* always jumps to "the screen calling the task".

Returning to the initial screen exits the editing of the current object.

From: Moving at the Same Hierarchy Levels

- **Nested lists/tables**: Previous list via *Previous <object component>*, next list via *Next <object component>*

From: Moving to Subordinate Hierarchy Levels

- **Only one view**: If there is only one screen for every line of the list, the user can choose an entry by positioning the cursor on the required line and by choosing the function *Choose* (F2).
- **Several views**: If several views are possible for every line, the user selects an entry by positioning the cursor on the required line. The user then chooses the view by activating a corresponding function. Here, the function *Choose* (F2) might select the most often used function (default function).

Navigation: The Object Component Screen

From: Moving to Superordinate Hierarchy Levels

- **Standard navigation**: With *Back*, you get to the screen that is exactly one level higher in the hierarchy.
  With *Other <object>*, the user always reaches the initial screen.
  With *Exit*, the user reaches "the screen calling the task".
  If data is not yet saved, a safety prompt must be displayed.
- **Saving when making fast entries**: In some cases (for example fast entry), it can be useful to link the function *Save* or *Post* with a simultaneous navigation to the initial screen.
- **Navigation in case of inconsistent data**: If navigation to a superordinate hierarchy level is actually impossible (e.g., because data is still inconsistent), you have to support *Cancel* anyway. In this case, however, data entered by the user will be lost.

From: Moving at the Same Hierarchy Levels

- **Unrestricted access**: If it was possible to choose the object component for processing without restrictions via various functions, then use the same functions also for navigating at the same hierarchy level.
- **Access via object component list screen**: If the access to the object processing occurred via the object component list screen, then after having processed an object component the user can branch as follows: with *Previous <object component>* to the object component's predecessor (according to the list), and with *Next <object component>* to the object component's successor (according to the list).
  With the function *Other <object component>*, the user can branch directly to any object component on the list. You need not support this, however. With *Other <object component>*, it is not possible to branch to an object component outside of the list.
- **Preselection**: If a selection has already been made by selecting something from the object component list, the functions only branch between these selected lines.
- **Nested lists/tables**: If the displayed screen in turn consists of a list/table via which the user can branch to a "component/object component" screen, then *Previous <object component>* and *Next <object component>* only navigate at the current level.
- **Changing the view for object components**: Sometimes you can change the view on subordinate screens, that is, the user can look at yet further screens to an object component. Then after *Previous <object component>*, *Next <object component>* he should see again the original view for the next displayed object component with which he initially accessed the display of the object components.
- **Predefined sequence**: Previous screen via *Previous screen*, next screen via *Next screen* or *Continue*.

From: Moving to Subordinate Hierarchy Levels

If the subordinate hierarchy levels can be defined freely, any functions can be defined (see initial screen); if a choice was made via a list, then a distinction is to be made as on the object component list screen (see there).

Navigation: The Sub-Component of Object Component Screen

From: Moving to Superordinate Hierarchy Levels

- **Standard navigation**: With *Back*, the user returns to the object component screen, with *Other <object>*, to the initial screen, and with *Exit* to "the screen calling the task".
- **Navigation when data is inconsistent**: If it is not possible to navigate to a superordinate hierarchy (e.g., because the data is not consistent at this point in time), you have to support the function *Cancel* anyway. The data entered by the user, however, is lost in this case.

From: Moving at Same Hierarchy Levels

The same rules as for navigating between object component screens apply.

From: Moving to Subordinate Hierarchy Levels

If further hierarchy levels exist, the same rules as for navigating from an object component screen to a sub-component of an object screen apply.

 top

Source: SAP R/3 Style Guide

Designing Screens

Overview of Controls | A Typical R/3 Window

Before we are going to design screens, we introduce the concept of windows to you and give you a general overview on the interface elements that are used in R/3 windows. Finally, we show a typical R/3 window with its interface elements.

Windows

Up to now we talked very generally of "screens" and "interface elements". R/3 screens are parts of *windows*. Windows are the primary interface element of modern graphical user interfaces. They cover a rectangular area on the computer screen and represent certain tasks or applications running on the computer. The user may move windows around the computer screen, size, stack, activate, or deactivate them.

When we talk of screens in the context of the R/3 System, we usually mean the *work area* of a window that is its main part. You as the developer place interface elements into the work area to fit a window to a certain task. In addition to the work area a window consists of further elements. Some of them serve for basic window handling, and some of them are specific for R/3. You may place interface elements into some of these areas, too.

There are two kinds of windows in the R/3 System, the *primary windows*, and the *secondary windows*, usually referred to as dialogue windows or popup windows. The R/3 System always resides in one main or primary window where the user's main activity takes place. In addition, dialogue windows may pop up in reaction to the user's actions to supplement the main window. Here the user may enter additional data, make choices or is informed on errors or consequences of actions. There may be more than one dialogue window at a time, but only one can be worked with. This is because in the R/3 System dialogue windows are "modal" windows, which take over control of the system.

The user may open more than one primary window in the R/3 System and exchange data between the sessions running in the various windows.

 top

Source: SAP R/3 Style Guide

 Designing Screens

Overview of Controls

The SAPGUI provides a number of standard interface elements or controls that are used in graphical user interfaces and that you can use in your applications. This document lists the interface elements used in the SAPGUI following a simple classification scheme. Each type of interface element is introduced in short first. Then the SAPGUI-specific variants of this type - shown in italics - are presented.

Elements for Input and/or Output of Data

These elements are used

- to display information the user can possibly change
- to receive a small up to a very large amount of information from the user

Output elements only display information. The user cannot change the data or can do so only under particular circumstances.

Input/output fields are text fields in the work area of a screen. The user can enter data in I/O fields or the system can display information there. *Field names* are text fields (output fields) which belong to input/output fields. They identify the function of the particular fields. Column or row headers in tables and lists serve a similar purpose. *Short descriptions* are text fields (output fields) which describe the contents of I/O fields in more detail. All these fields are defined in the Screen Painter.

Additional I/O elements in the R/3 System are the *command field* and the *status bar*. The command field allows the user to type in a command. The status bar displays system messages and informs the user about the system status.

Elements Initiating Actions

These elements initiate an action when the user executes a particular function. The action can be performed on the current screen or lead to a screen change.

Pushbuttons are fields with a textual or graphical label and initiate a particular action, if activated. They may be grouped together in toolbars or floating windows. In the R/3 System, you find pushbuttons in the *standard toolbar*, in the *application toolbar*, and in the work area of a screen at no fixed position.

Pushbuttons can be activated with the mouse (single mouse-click) or (mostly) the keyboard. Pushbuttons of the *application toolbar* and the *standard toolbar* are defined in the Menu Painter. *Pushbuttons with no fixed position* are defined in the Screen Painter.

Function keys also initiate an action, but can only be activated with the keyboard. Some function keys also appear as pushbuttons in the *application toolbar* or in the *standard toolbar*. All active function keys are displayed in the *function key menu*. To display the menu, click the right mouse button.

Menus

A menu is a list of items that the user can choose one at a time. Menus can show for example, functions or attributes.

Menu bar, Pull-down menus: We distinguish between the menu bar and pull-down menus. The menu bar is a horizontal bar just below the title bar of the window.
A pull-down menu is a submenu attached to the bottom of a menu bar item and usually hidden. It appears when the user clicks a menu bar item with the mouse. Pull-down menus are arranged vertically and may be cascaded, i.e. arranged hierarchically.

Besides using the mouse, you can activate menus with the function *Menu bar* (F10). You select the required menu option with the arrow keys and press ENTER to activate it. To access a menu directly, you can define access characters that the user has to press in combination with the ALT key. The *fastpath* is another fast access method: In the command field, the user enters a period followed by a string of access characters.

Elements for Making Selections

These elements allow the user to choose options or attributes. There are selection elements for different purposes.

Radio buttons allow the user to choose exactly one item of several alternatives. The number of alternatives is fixed and all alternatives should fit on the screen. One option is always selected. *Checkboxes* allow the user to choose as many alternatives as desired or none at all. The number of alternatives is fixed and all alternatives should fit on a screen.

Single- and multiple-selection lists contain at least two lines. The length of the list, however, is usually not fixed and all of the alternatives need not fit on one screen. In single-selection lists the user can only choose one line, in multiple-selection lists the user can choose several lines. Single-selection lists may mark items with radio buttons, multiple-selection lists use checkboxes.

Elements for Making Adjustments

These elements can be used for different purposes. A typical case is to choose a value from a value set, for example, by moving a slider on a scale. This is possible for discrete as well as for continuous values.

Scroll bars provide a means for viewing an area larger than the work area on the screen. They inform the user that there is more information available than displayed on the current screen. There are vertical and horizontal scroll bars. These allow to scroll up and down or left and right in the information.

Spin buttons let the user increment or decrement values in standard increments. They can also be used for non-numerical data, for example for choosing a month or a day of the week.

Elements for Grouping Objects

These elements are used for visual grouping of related control elements on a screen. There is just one such element in the R/3 System, the *group box*. Group boxes provide grouping of I/O fields with field names or radio buttons or checkboxes. They consist of a rectangular frame and are identified by a *group heading*.

top

Source: SAP R/3 Style Guide

Designing Screens

A Typical R/3 Window

A typical primary R/3 window consists of the following elements (from top to bottom):

- the *title bar* the text in the title bar informs the user about the current position and activity in the system
- the *menu bar* contains the headings under which the pull-down menus are grouped; clicking the mouse into a menu bar item shows a pull-down menu with a list of possible actions
- the *standard toolbar* contains pushbuttons for basic R/3 functions; it also includes the command field where the user can enter commands
- the *application toolbar* contains pushbuttons for often used functions that are specific for the application
- the *work area* contains fields, group boxes, pushbuttons, radio buttons and checkboxes, tables, lists etc.
- the *status bar* is where the system issues information to the user, and where error messages may appear

Dialogue windows lack some of these elements and are usually smaller than primary windows.

The following figure shows the elements of a typical primary and secondary window.

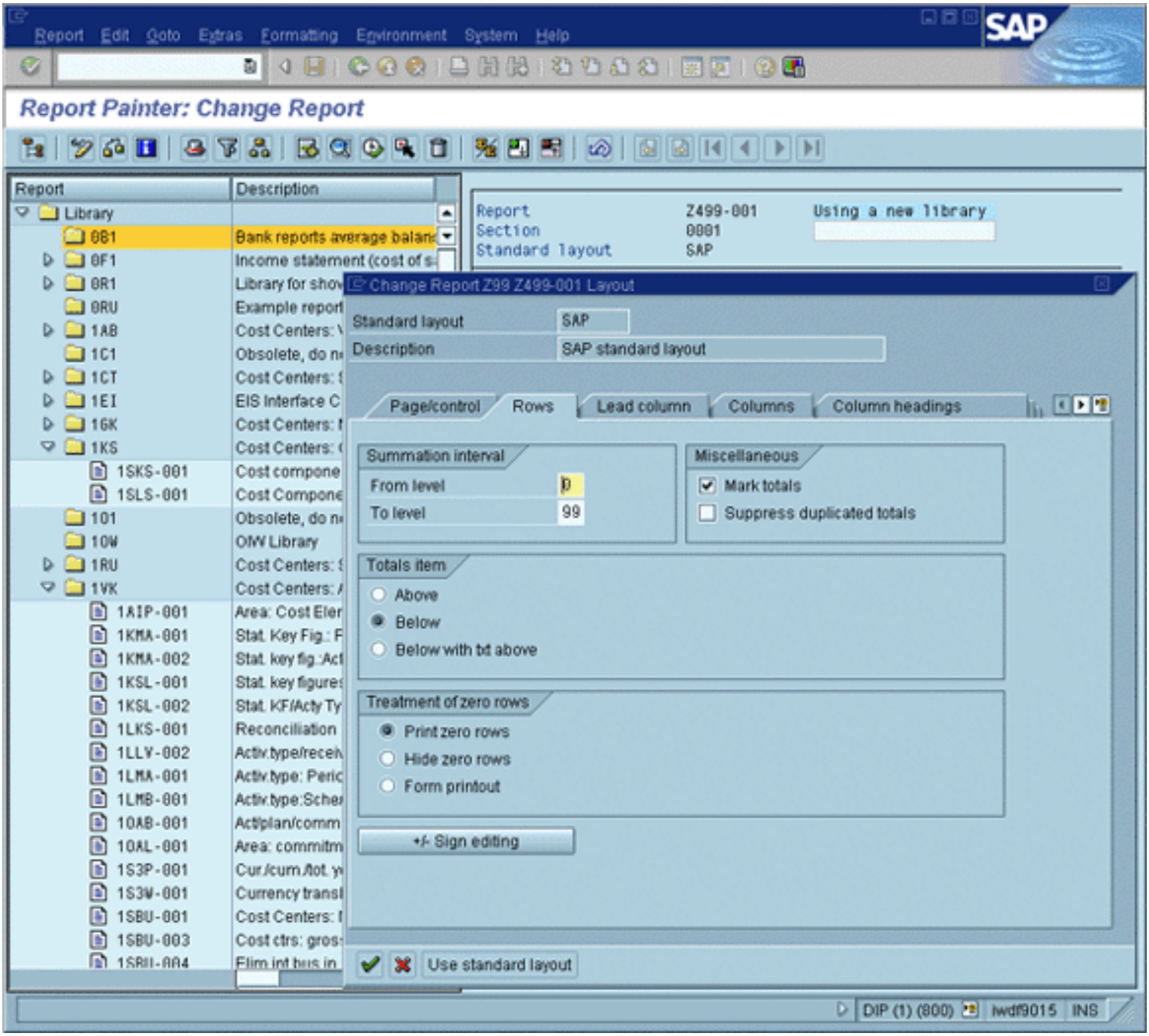


Figure 1: The SAPGUI window elements

Click image to enlarge it!

top

Source: **SAP R/3 Style Guide**

Menus

General Design Guidelines | R/3 Menus - Overview of the Levels | Menus at the Application Level | Menus at the Task Level

In the early computer times users had to issue text-based commands to the computer. These commands had parameters and lots of options and were usually rather cryptic. Users had to remember these commands which was very difficult (and still is). Menus provide users with a list of the available commands or options. The user can select the wanted command from a list and has only to recognize the command. This is much easier than having to remember it. Menus help in particular beginners and casual users to exploit the available system functionality.

That is not the whole story, however. Menus may require a lot of screen space when they list all available commands. If the list of commands is long users have to search for the commands. This may take quite a long time. There are even more problems: What can be done if the list of commands is longer than would fit on the screen? Should users scroll the menu list in this case? Where should the "real" information go when the menu occupies the screen?

One solution to these problems is the now common pull-down menu. There are lots of variants of them, but the basic principle is always the same: The functions are grouped under some - hopefully - descriptive label. Only the group names are displayed on the screen, for example at the upper border of the screen or of the application window. The functions themselves are arranged in submenus that are usually hidden.

To access a function, the user selects the group where he or she assumes the function should be. A menu pops up, and the user selects the desired function from the list of menu items. Sometimes menus are organized in even deeper levels. Then the user has to open cascading menus, maybe over several levels. This way the pull-down menu forms a hierarchical structure that comprises most or all application functions.

This leads us to some of the problems of pull-down menus:

- Functions may be buried deeply in the tree-like menu structure and difficult to find.
- Users may not know where to search for a function.
- Users may be "lost in function space": The space of available function offers lots of opportunities, but the users do not know which steps actually to take; they have no guidance from the system.

You can take some preventive measures to overcome these problems:

- Look for descriptive labels that the users understand if you group functions.
- Consider where you put functions. Ask other people, whether they would look for a function at the location where you placed it.
- Do not bury functions too deeply. They may never be found and they are cumbersome to access. Put only seldom-used functions into cascading menus.
- Do not make lists of action menus too long to avoid excessive search. Though the magical number 7 may often be too small, 25, for example, may be far too large.
- Place functions in standard locations. Users will look for the functions in these locations, because other applications provide them there, too.

Today we recommend the use of pushbuttons to offer functions which are appropriate for the task the user is currently performing. This helps in many cases to provide only slim menus while focussing on the functions which are relevant to the course of user action.

 top

Source: SAP R/3 Style Guide

 Menus

General Design Guidelines for Menus

The menu selection of a function follows the *object-action approach*: First, the user has to specify the object class, and then the corresponding action. Therefore, put the object to be processed in the menu bar and the actions that can be performed on it on the first level of the pull-down menu. If the object has for some reason to be placed in the pull-down menu and there are only few actions, write the action on the same level directly behind the object. If the action explicitly results from the context of the object selection you can omit the specification of an action.

Assigning Functions to Menu Levels

Arrange the functions for a menu bar option on as few menu levels as possible so that the users can view the alternatives in parallel. Furthermore, menus with a flat hierarchy speed up interaction if the users only work with the keyboard.

If possible, put the most important and most frequently used functions on the first level of the pull-down menu. Hide functions that are rarely used or complex on a second or third level.

In some cases, there will be many options in a menu so that some of them would have to be moved to the second or third level. If there is still space for further options in the menu bar, split the menu into two menus with less options on the first level. For the two new menu titles, choose terms that are typical for the subordinate menu options and can easily be distinguished.

Consistency of the Menu Bar and of the Pull-Down Menus

The *menu bar* should be consistent within a task. Include all menu bar options of the data screens on the initial screen as well. This avoids any inconsistency caused by the initial screen having an additional, unclear status between application level menu and task level menu.

As far as possible, the *pull-down menus* should have the same options within a task. This also applies to the entire application, if possible.

Status of the Menu Bar Options and of the Pull-Down Options

Display *menu bar options* that contain at least one selectable function in the pull-down menu as being active, otherwise, as being inactive. The pull-down menu can be opened anyway.

Display an *option in the pull-down menu* as being active if it initiates a function in the current dialogue state or if the subordinate level contains at least one active option. Otherwise, display it as being inactive. Display also options as being inactive if, in turn, there are only inactive options at the subordinate level of the pull-down menu. The pull-down menu subordinate to the inactive option can be opened anyway.

Display functions, that cannot be chosen in individual dialogue states of a task as being inactive and do not hide them, regardless of the user's authorization. This enables communication between users with different authorizations. Furthermore, changing an authorization does not require to change the menu.

If a pull-down option is displayed as being inactive, the function is neither displayed as function key nor as a pushbutton.

Additional Guidelines

- **Grouping of pull-down options**: Combine the menu options to form useful groups. Separate the groups from each other by a horizontal line.
- **Following dialogue box**: If selecting a menu option displays a dialogue box, indicate this by placing three periods after the menu option. Do not use blanks between the option name and the first period.
- **Menu selection with access characters**: The user may select a menu bar option by entering an access character on the keyboard. These access characters are defined in the Menu Painter. For a number of standard functions, the characters are predefined. Otherwise use a character that is typical for the option. This is usually the first letter or a letter which uniquely distinguishes options with the same first letter.
- **Upper/lower case**: Menu options can consist of several words. Write them in lowercase letter in English and according to the rules for upper/lowercase in other languages. The very first letter of the menu title is always capitalized.

R/3 Menus

The basic hierarchy of the menu bars reproduces the three levels of the R/3 System, the main menu level, the application level, and the task level. Structure the menu bar uniformly within a task to ensure consistency. Therefore, do not remove functions that are not used from the menu bar and the pull-down menus but set inactive only.

Note: The *System* and *Help* menus are always available in the R/3 System. They are appended automatically to the right of the individual menus at the respective level. The same applies to the *Layout* menu, which appears to the right of the menu bar as icon. It allows the user to change local settings like fonts or interface colors.

Main Menu Level

In the menu bar at the main menu level, the R/3 System displays all application areas available. The following application areas are currently available: Office, Logistics, Accounting, Human resources, Information systems, Tools.

Application Level

At the application level, object classes or tasks (= transactions of an application) area are listed in the menu bar.

Task Level

At the task level, the user edits data. The structure of the menu bar and the corresponding pull-down menus are largely standardized. The task level itself can have a hierarchical structure, too.


top

Source: SAP R/3 Style Guide

 Menus

R/3 Menus - Overview of the Levels

The following overview describes the basic hierarchy of the menu bars at the three levels of the R/3 System, the main menu level, the application level, and the task level. The task level itself can have a hierarchical structure, too. Structure the menu bar uniformly within a task to ensure consistency. Therefore, do not remove functions that are not used from the menu bar and the pull-down menus but set inactive only.

Note: The *System* and *Help* menus are always available in the R/3 System. They are appended automatically to the right of the individual menus at the respective level. The same applies to the *Layout* menu, which appears to the right of the menu bar as icon. It allows the user to change local settings like fonts or interface colors.

Main Menu Level

In the menu bar at the main menu level, the R/3 System displays all application areas available. The following application areas are currently available: Office, Logistics, Accounting, Human resources, Information systems, Tools.

Application Level

At the application level, object classes or tasks = transactions of an application area are listed in the menu bar.

For guidelines on how to design a menu bar on the application level see Menus at the Application Level.

Task Level

At the task level, the user edits data. The structure of the menu bar and the corresponding pull-down menus are largely standardized. In the following you find an overview of the individual menu bar options in the defined sequence from left to right, together with a short definition and obligatory or typical pull-down menus.

For a detailed description of the menu strcuture at the task level see Menus at the Task Level.

 top

Source: SAP R/3 Style Guide

 Menus

Menus at the Application Level

The application level is the second level of the three-level system hierarchy. Within an application, the user should be able to choose all related object classes and the corresponding actions to start a task in order to process a particular object of this class.

Below, we give a step-by-step description on how to design a menu bar on the application level.

Step 1: Include all Object Classes in the Menu Bar

When structuring the menu bar, first try to include all object classes of the application in the menu bar. Then assign the individual actions to these object classes in the respective pull-down menus.

Step 2: Create Super Classes

If there is not enough space in the menu bar (the menu bar can have up to 6 menus in addition to the system menus), group the object classes together to form super classes. Provide these super classes with mnemonic names which characterize the subordinate object classes, so that the users have no problems with allocating the individual object classes to the individual super classes.

Assign the object classes to the super classes in the respective pull-down menus. In the pull-down menus of the next level, assign then the actions to the object classes.

Step 3: Form Useful Units

If the number of object classes of a pull-down menu is more than seven (orientation value), group the object classes within the pull-down menu to form a maximum of seven useful units separated by lines.

Step 4: Repeat these three Steps, if necessary

If there is still not enough room in the menu bar, repeat the three-step procedure described several times by grouping the super classes.

Goal: Flat Hierarchy

When structuring the menus in this way, the total quantity of the pull-down options subordinate to a menu bar option should be arranged on as few hierarchy levels as possible.

 top

Source: **SAP R/3 Style Guide**

 Menus

Menus at the Task Level

The task level is the third level of the three-level system hierarchy. Within the task level of the R/3 System, the user can call a large number of individual tasks. Each of these tasks is used for processing objects of a certain object class. The menu bar is structured in the same way across all tasks so that the user can easily find his or her way around in the different tasks.

The menu bar and the pull-down menus of the SAP application functions consist of obligatory and optional entries. Optional entries are to be used and arranged in the specified form, if required. Variable entries are indicated by angle brackets.

The standard menu bar contains the following options in the specified order:

<Object> Edit Goto Extras Environment <Application area> **System Help**

The options <*Object*>, *Edit*, and *Goto* are required, *System* and *Help* are added automatically; all other options are optional.

In addition, you can include the menu titles *View*, *Settings* and *Utilities*, if required. Position them between *Extras* and *Environment* in the specified order. Further menus for special features of individual tasks are possible. Keep in mind, however, that in addition to menus automatically added by the system, you can freely define only 6 entries.

In the following we present the definition for each individual menu bar entry in the order in which they are to be placed in the menu bar, together with a short definition and obligatory or typical pull-down menus.

Keys

Bold = pull-down option: obligatory

Normal = pull-down option: optional

--- = here you should insert separator lines

<Object>

The <*Object*> menu (obligatory) contains actions the user can choose to manipulate an object as a whole. It receive the name of the object that has been selected at the next higher level This includes the functions for requesting, saving, printing, and deleting objects. *Exit* is always the last menu option.

Put actions which affect the entire object here; they -- obligatory

1) Initial Screens:

- Actions for choosing another object- Aktionen zur Auswahl eines anderen Objekts

  Other <object>

  Create

Change

Display

Copy from...

- Backup actions ------

Save

Hold

Generate

- Other actions (for example import/export functions) ------

Print

- Closing actions ------

Delete

Exit

2) Data Screens:

- Actions for choosing another object (Aktionen zur Auswahl eines anderen Objekts)

Other <object>

Copy from...

- Backup action

Save (or Post)

- Other actions (for example import/export functions)

Print

- Closing actions

Exit

Edit

The *Edit* menu (obligatory) contains actions the user can choose to edit the current object component. This includes the functions for selecting, editing, canceling, clipboard and application-specific functions. *Cancel* is always the last option.

- Select actions

  Select all

  Deselect all

  Select block

  Choose

- Editing action ------

  Cut

  Copy

  Paste

  Move

- Other actions ------

  Insert line

  Delete line

  Sort

  Delete

- Reset actions ------

  Cancel (Abbrechen)

Note: If the selection actions appears in a cascading menu at the second level, use the term "Selections" on the first level:

```
Selections -> Select all
              Deselect all
              Select block
```

Menu Functions "Other Actions"

You may place here functions which affect an object component of the entire object, for example, the functions *Delete* and *Print*. However, if these functions refer to the object as a whole, include them in the menu <*Object*>.

Note: Functions related to table controls do not need to appear in the Edit Menu. They should appear as pushbuttons directly below the table.

Goto

The processing of an object consists of a sequence of object component processing. The *Goto* menu (obligatory) contains all object components or all screens of the current object the user can navigate to by choosing the option. It is the navigation menu within the individual object components of an object. If there are two many object components, you may split the *Goto* menu into several menus (if there is room in the menu bar). They have to be named accordingly.

Back is the last option of the pull-down menu at first level. If the *Goto* menu is split into several menus, then include *Back* in the first menu from the left as the bottom option of the pull-down menu at first level.

Overview (Übersicht)

Header (Kopf)

Next <object component>

Previous <object component>

Other <object component>...

Back

Only the function "Back" is an obligatory entry, all other entries are examples.

Extras

The *Extras* menu (optional) contains functions to complete the current object or object component by supplying additional data or options and which are not constantly needed. They generally initiate the display of a dialogue box that contains additional information or options.

Goto menu vs. Extras menu

The *Goto* menu contains the main screens of the transaction that the user generally has to pass during the object processing but that are relatively independent from each other. The *Extras* menu, on the other hand, often contains the secondary screens of a main screen that are only occasionally processed or displayed. The information of the secondary screen depends directly on the main screen.

To make this dependency structure clear on the user interface, do not arrange the calls of the secondary screens under *Goto* and display additional information in a dialogue box.

Environment

The *Environment* menu (optional) contains functions the user can choose to call transactions or reports of other applications.

This menu serves two functions. The first group of functions in this menu should provide the user with information occasionally

needed for the processing of the current object. They either call a display transaction of another application or start a report without exiting or canceling the current processing. Separate this group of display functions visually from the second group of maintenance transactions of other applications that the user occasionally needs for the processing of the objects. This integration allows an integral work process. In this case, the current object processing is also only interrupted.

From the called task, the user should always be able to return to the interrupted task by using a constant function key (or *Back*).

<Application Area>

The <*Application area*> menu (optional) contains the menu options of the higher-level application area menu. This menu allows the user to alternate between the application functions of the current application. The current application function is exited or canceled. A safety prompt must therefore be issued to avoid a possible data loss.

This menu replaces the explicit exiting of the current application function and the selection of a new application function at application level. Display the current object class as being inactive, because the same functionality is already available in the menu <*Object*>.

View

The *View* menu (optional) contains the functions the user can choose to display the current object or object component in different views (for example, when switching between a one-line and two-line display of a table). The functions do not change the data of the object or the object component.

Settings

The *Settings* menu (optional) contains functions that allow the user to set user-specific transaction parameters.

Utilities

The *Utilities* menu (optional) contains functions that allow not only the processing of the current object but also cross-object processing. This includes general functions, such as *Delete*, *Copy*, and *Print*.

Help, System, Layout

The *Help* and the *System* menu are always available in the system. They are added automatically to the menu bar, and you cannot change them.

 top

Source: **SAP R/3 Style Guide**

Fields

Field Names and Short Descriptions | Group Boxes | Arranging Fields and Field Groups | Check Texts

Input/Output Fields

Input/output fields (I/O fields) are fields in the work area of a window. Here the user can enter information, or the system displays already known data in them.

Input fields are used for entering data. They may be set to read-only status. This is useful, for instance, when a task can be toggled between display and change mode. Another useful situation is when the user navigates to a subsequent screen. There the user can view the data entered on the previous screen but cannot change them.

Pure *output fields* display information. In the R/3 System, they are used as *field names*, as table or column headings, and to show *descriptive texts* (see figure 1).

Appearance

Depending on their function, I/O fields look differently:

- input fields are displayed in 3D-look and in a special color, usually in white
- if an input field cannot be changed in a particular dialogue situation (read-only field), it has the color of the window background, but remains in 3D-look
- pure output fields, however, are displayed flat on the window background and in background color

You set the corresponding flags in the Screen Painter.

Read-only and output fields can be chosen with the active control indicator (Windows: Ctrl arrow key) and with the mouse. This allows the user to request help and to position the active control indicator in a selection table.

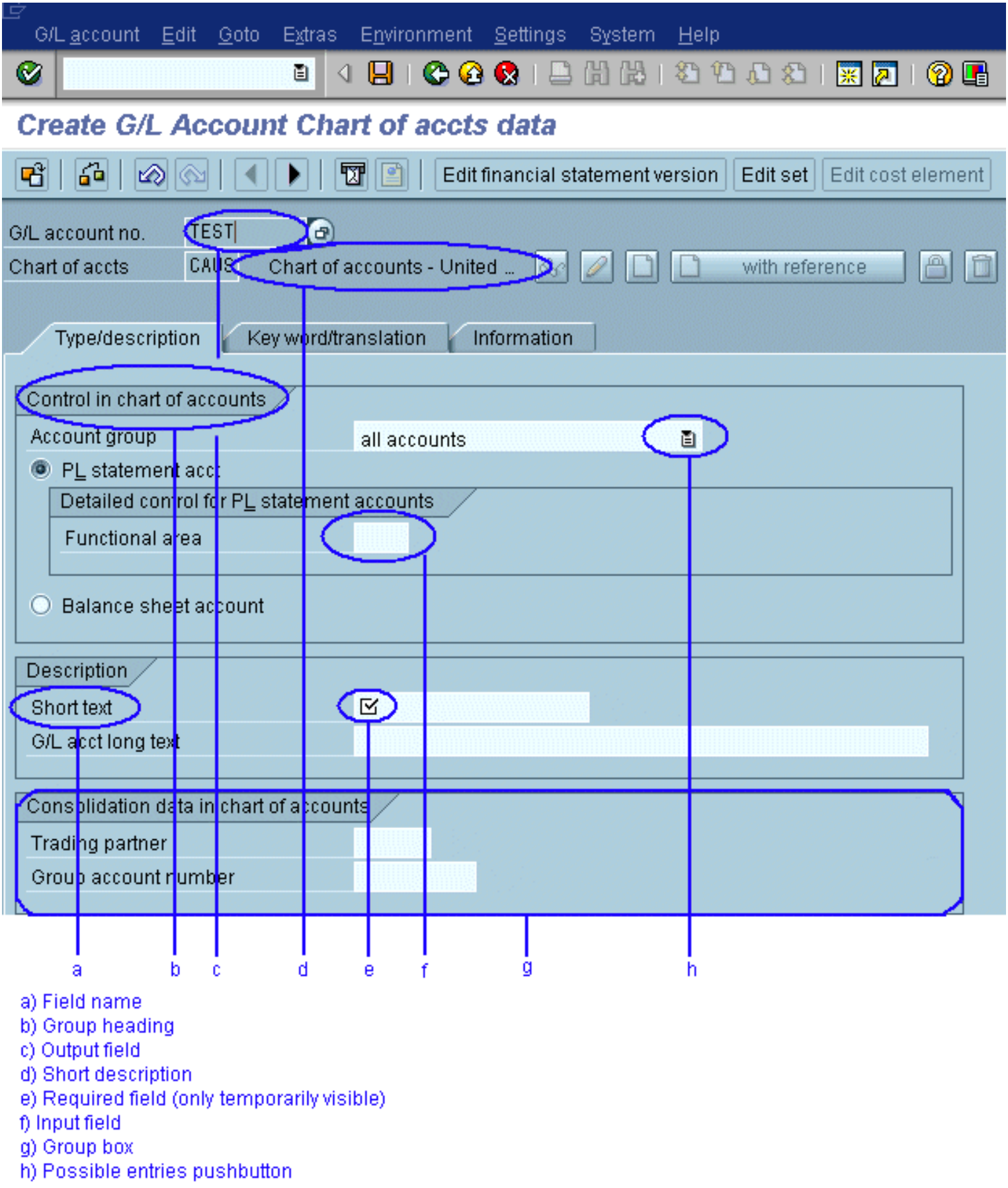


Figure 1: A typical R/3 screen with different types of fields and with group boxes

Further characteristics of I/O fields:

- **Length**: The length of an input field should indicate how many characters the user can enter. Thus, it usually corresponds to the maximum number of characters the user can enter. In certain cases, however, it is advisable to assign a "visible" length to an input/output field that is shorter than its physical length and make it scrollable.
- **Character Set**: For input data, the system standard is a non-proportional character set to indicate the length specification. Output fields are usually displayed in proportional typeface.
- **Required Fields**: Required fields are flagged by a question mark "?".
- **Error Fields**: Fields in which an error has occurred are highlighted by a different background and/or text color.
- **Fields With Possible Entries Pushbutton**: Matchcode fields and other fields with help on input values have a *possible entries pushbutton* to the right. It may be permanent or appear when the cursor is placed into the field.

Design Guidelines

Input/output fields are usually assigned to a field name and positioned to the right of it. A descriptive text may follow to the right of the I/O field. Fields can be clustered into blocks. These can be further assembled into block groups consisting of one or more blocks. A typical block group consists of a block of field names and a block of corresponding input fields.

Spacing

Separate field name and I/O field by at least **two** blanks. Separate the descriptive text from the I/O field by at least **three** blanks to leave room for the possible entries pushbutton (**Exception**: tables).

Displaying Output Values

- Do not display leading zeros in numeric fields.

Example - Do not!	**Example - Do!**
Net amount_____ 00000234,78	Net amount_____ 234,78

Exception: The leading zero for the month should be displayed.

Date **3.02.95**

- Divide values with important subcomponent to emphasize the importance of these parts. Use special characters for the separation.

Net value **12.323.411,23** Date **12.12.90**

Positioning Quantity and Currency Units

Place quantity or currency units to the right of the value. Leave only one blank between value and quantity or currency unit (see example 1).

Exception: If units are usually positioned to the left of the value, follow this arrangement in order to not confuse the user.

Check if you can include the unit in the field name (see example 2). In this case the quantity unit should appear in parenthesis. Insert a blank between field name and quantity unit and make sure that there are at least two blanks at the end of the entire field name.

Example 1		**Example 2 - alternative**	
Material	**Screws, small**	Material	**Screws, small**
Quantity	**20 kg**	Quantity (kg)	**20**

I/O Fields With Several Parts

You can assign several I/O fields to a field name if (1) you split an I/O field into several logical parts or (2) several related I/O fields are together on the screen.

Try to split the field name in accordance with the I/O fields. Separate the individual parts of the I/O field from one another by meaningful special characters. Place the special characters as close to the I/O fields as possible. Remember, however, that some special characters might have a different meaning in other countries.

```
Plant/storage location       _____ / ______
```

Number of I/O Fields per Field Name

- Every I/O field must have some sort of field name.
- If two fields are closely related, you can assign both to the same field name. The field name does not have to refer to the subdivision.

  ```
  City      __________ ___________
  ```

- If there are several I/O elements whose interrelation is not immediately clear, you should subdivide the field name accordingly. Do not use this approach for more than three fields.

  ```
  Plant/storage location      _____ / ______
  ```

- You can also split up the field name into subnames. Follow this approach only if the subname does not have more than one or two characters and the subkey is properly structured. Use the colons as in the example.

  ```
  Terms of delivery       1: _____  2: _____  3: _____  4: ______
  ```

Exception: The following fields have no field names:

- the field in which you can enter a start argument under a table
- selection columns

Scrollable Input/Output Fields

I/O fields can be made scrollable. Scrollable I/O fields use less space on the screen, because they display fewer characters than they are actually long. That is, their "visible" length is shorter than their "physical" length. Use scrollable I/O fields where space requirements are tight. There is, however, one disadvantage: Users see only one part of the information, they have to scroll to see the rest.

Appearance

You can recognize scrollable input fields by their dotted borders to the left and to the right. Scrollable output fields show only dots at the right border.

Figure 2: Scrollable output field

Scrollable descriptive texts are displayed flat on the screen. A right pointing arrow at the end of the text signifies that the text is scrollable.

Figure 3: Scrollable descriptive text

Scrollable Input Fields

Use	**Typical key fields** (e.g. business area, material number): Set the visible length long enough that under normal circumstances all characters are visible. Only under extreme conditions users should have to scroll.
	Texts and descriptions: Set the visible length long enough that excessive scrolling is avoided or that enough significant characters are displayed to enable the users to distinguish the meanings of the fields.
Do not use	**(Monetary) amount and quantities (numerical fields)**: In this case it is important that the user can view the complete numbers.

Scrollable Output Fields

Use	**Descriptive texts (check texts)**: These texts are the primary candidates for scrollable fields! Making these fields scrollable allows you to use explanatory texts in situation where space requirements otherwise would prevent their use. Set the visible length long enough that excessive scrolling is avoided.
Do not use	**Dates, quantity and monetary units**

top

Source: **SAP R/3 Style Guide**

Fields

Field Names and Short Descriptions

Field names

A field name is always assigned directly to an input or output field and briefly describes the contents of this field.

Appearance

Field names appear flat and in proportional typeface on the screen. The character font can set by the user and applies to all field names.

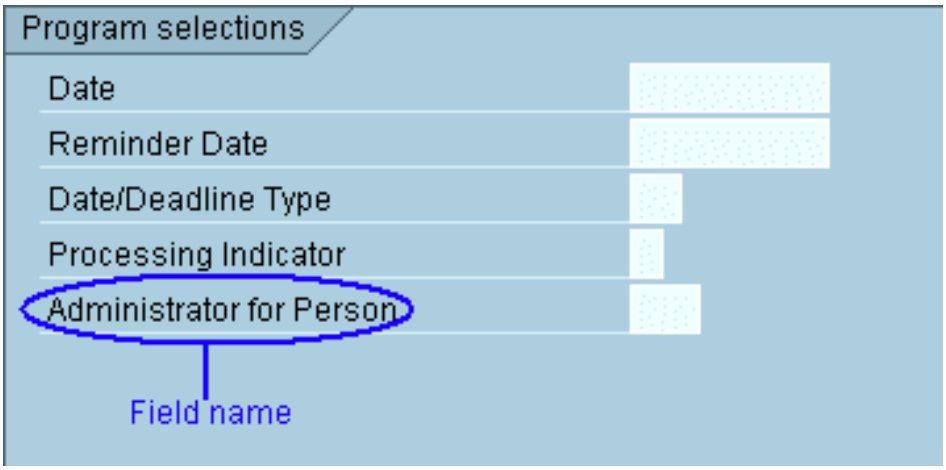


Figure 1: Example, showing the usage of field names

Design Guidelines

- **Familiar Terms**: Use words that are part of the user's working vocabulary. Do not use suffixes like "number" or "key" (**Exception**: personnel number).
- **Uppercase**: Always capitalize the first letter of a field name.

```
Initial value   12,6
Further value   13,7
```

- **Word Order**: If a field name consists of two or more words, the part containing the important information should appear at the beginning.
- **Length**: A field name should be no longer than 10 to 15 characters. If the field has been used very often, you can abbreviate the field name.
- **Number of Abbreviations**: Do not use more than two or three abbreviations in a field name.
- **Spacing Between Field Name and I/O Field**: Leave at least two blanks between the field name and the I/O field.

Icons as Field Names

- Icons can represent field names. They may replace common terms such as telephone, fax, date or time.
- Address icon plus text "Shipping address" to distinguish from "Factory address".
- Copy icon next to the field names *from* and *to.*

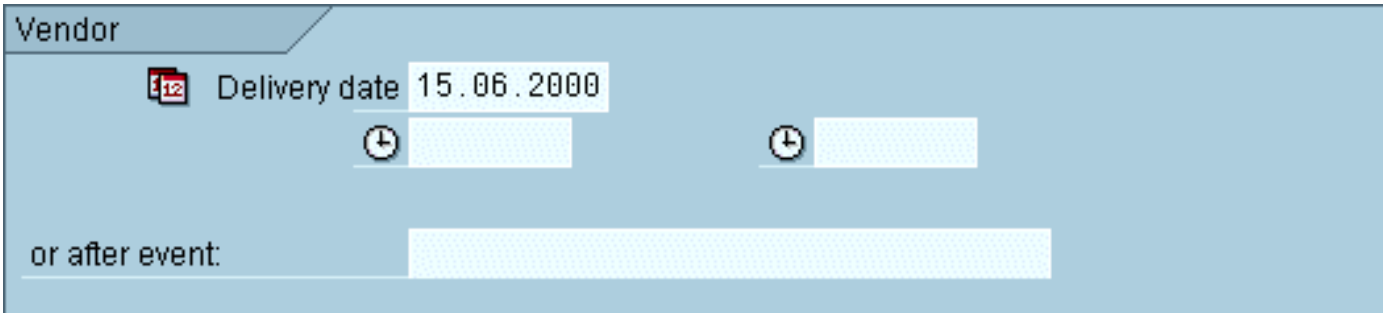


Figure 2: Field names implemented as text, icon, and icon plus text label

As an exception, icons can complete field names. In this case you should use either an icon plus a textual label or position the icon next to field names in order to indicate the relationship.

Short descriptions

Short descriptions (explanatory or check texts) are pure output fields. They follow an I/O field and explain what a numeric or alphanumeric key actually means.

Appearance

Check texts are usually displayed flat on the screen and in background color. If the check text has been entered by the user and not by the system, the check text appears on subsequent screens in 3D and background color.

Figure 3: Appearance of check text (right)

Check texts can be scrollable (see scrollable I/O fields). They are the primary candidates for scrollable, that is, shortened fields. A right pointing arrow at the end of the field indicates that the user can scroll the check text.

Figure 4: Appearance of scrollable check text

Design Guidelines

A check text follows an entry field, separated by three blanks. This makes sure that a possible entry-pushbutton does not hide the beginning of the check text.

Do not align the explanatory text with other fields on the screen if you have to move it so far away from the field to be explained that their spatial connection gets lost.

Exception: If the vertical alignment on a template is interrupted more than three or four times, you should redesign the template.

"23451-4" is the value of the output field, "Screws, nickel-plated" is the short description:

Plant	**2**	Requisitioner	**Piper**
Material	**23451-4**	Screws, nickel-plated	

If there is not enough room for the explanatory text field to the right of the value, place the text field in a separate line or make it scrollable. Define a separate field name for the text field in the first case.

top

Source: **SAP R/3 Style Guide**

Fields

Group Boxes

Group boxes help you to achieve a clearly structured screen design. They collate related information together in a box and identify it with a group heading.

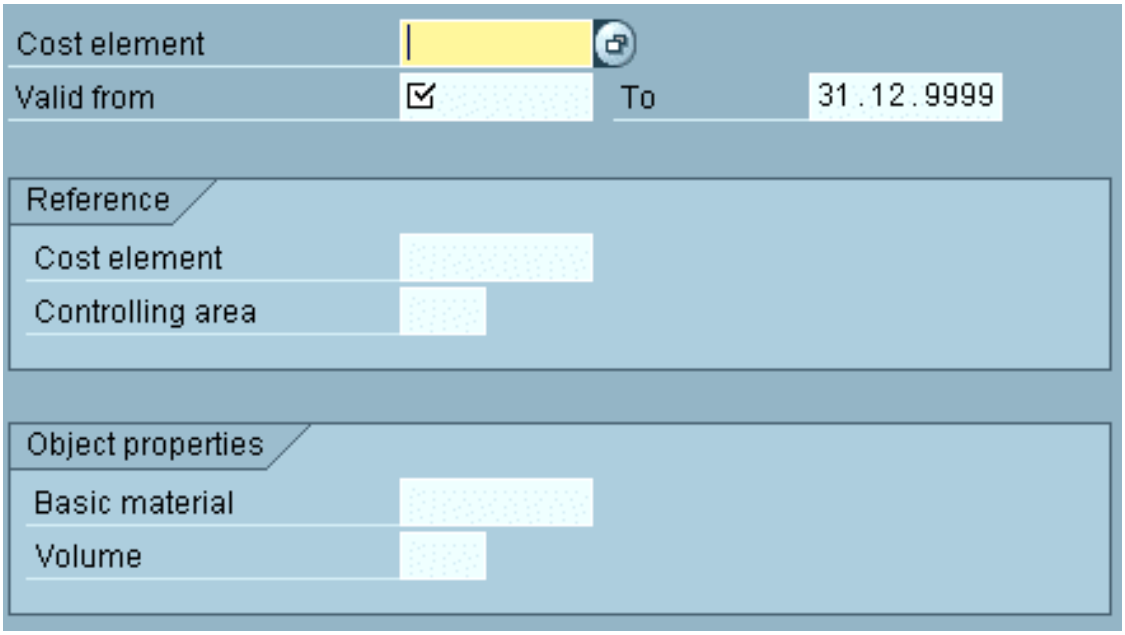


Figure 1: A typical R/3 screen with group boxes

Design Guidelines

Place larger groups of information units that you want to separate from other information in group boxes. Group boxes only contain particular groups, not the entire screen.

Exceptions

- Separate small information blocks that do not necessarily require separation from other information on the screen with a blank line only.
- Do not include header data in a group box. Use a blank line to separate them from other information on the screen.
- Dialogue boxes with only one information group do not contain a group box.

Headings

Every group box should have a group heading. Do not highlight the headings. You can also set headings at runtime.

Subscreens

When using subscreens, define the group box in the calling screen. You have to specify two extra lines for the box in the subscreen definition. The size of the area in the calling screen would have to be increased correspondingly.

Hiding

You can hide group boxes at runtime if all fields contained are also hidden.

Arrangement

Alignment of group boxes

Define group boxes according to the following guidelines, listed in terms of priority:

1. Often, you can divide the screen into two large columns. Place the right border of the left-column box at the center of the screen (as far as column 40). Place the right border of the right-column boxes that extend almost to the edge of the window (from about column 60, that is) at the right window border.
2. Align adjacent boxes, if you can achieve this by extending the border by a few blank columns or lines (orientation value: 2-3 blank lines or 5-10 blank columns).
3. If the first two guidelines are not applicable, place group boxes directly around the fields of the group.

With fields without a permanently displayed possible entries pushbutton, make sure to leave spacing of at least three columns behind the field so that the pushbutton does not obscure the group box.

Horizontal Arrangement

- If two group boxes appear next to each other, distribute them regularly on the screen: Place the border of the boxes at the screen center and/or the screen border. Separate the boxes by approx. 1-2 columns. Position larger boxes to the left.
- Text without group box (particularly header data) starts in column 1; this also applies to dialogue boxes.
- Texts and selection elements (checkboxes, radio buttons) inside the box start in the column directly following the box frame.

Vertical Arrangement

- Do not separate two group boxes positioned below each other by blank lines.
  Exception: On initial screens you can insert a blank line between the boxes to achieve a regular distribution.
- Separate the header data of the following group always by a blank line.

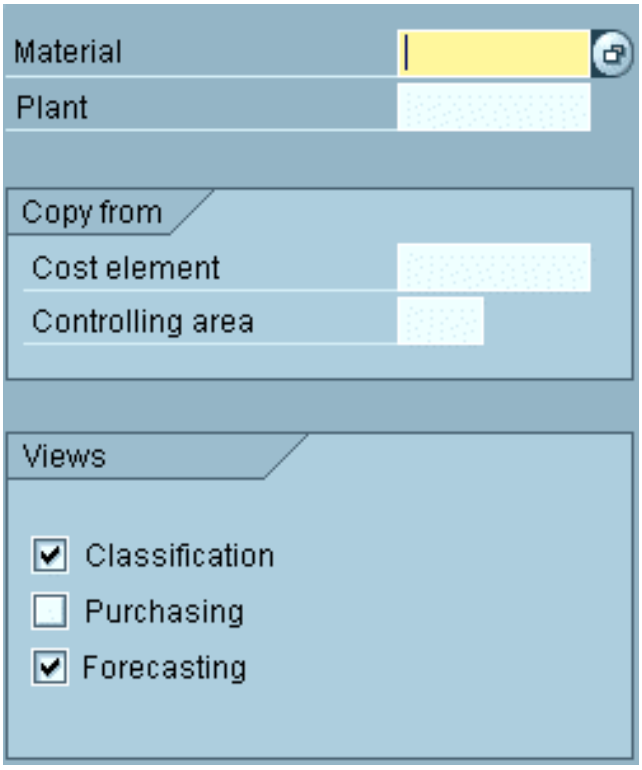


Figure 2: Illustration of guidelines for group boxes on entry screens

Tables

A group box may contain a table. The group heading serves as the table heading. Place pushbuttons for interacting with the table below the table inside the group box.

Last Check ...

When you have defined the group box following the above mentioned guidelines, check the screen if you should correct some of the elements slightly for a visually more pleasing result. A good overall impression has priority!

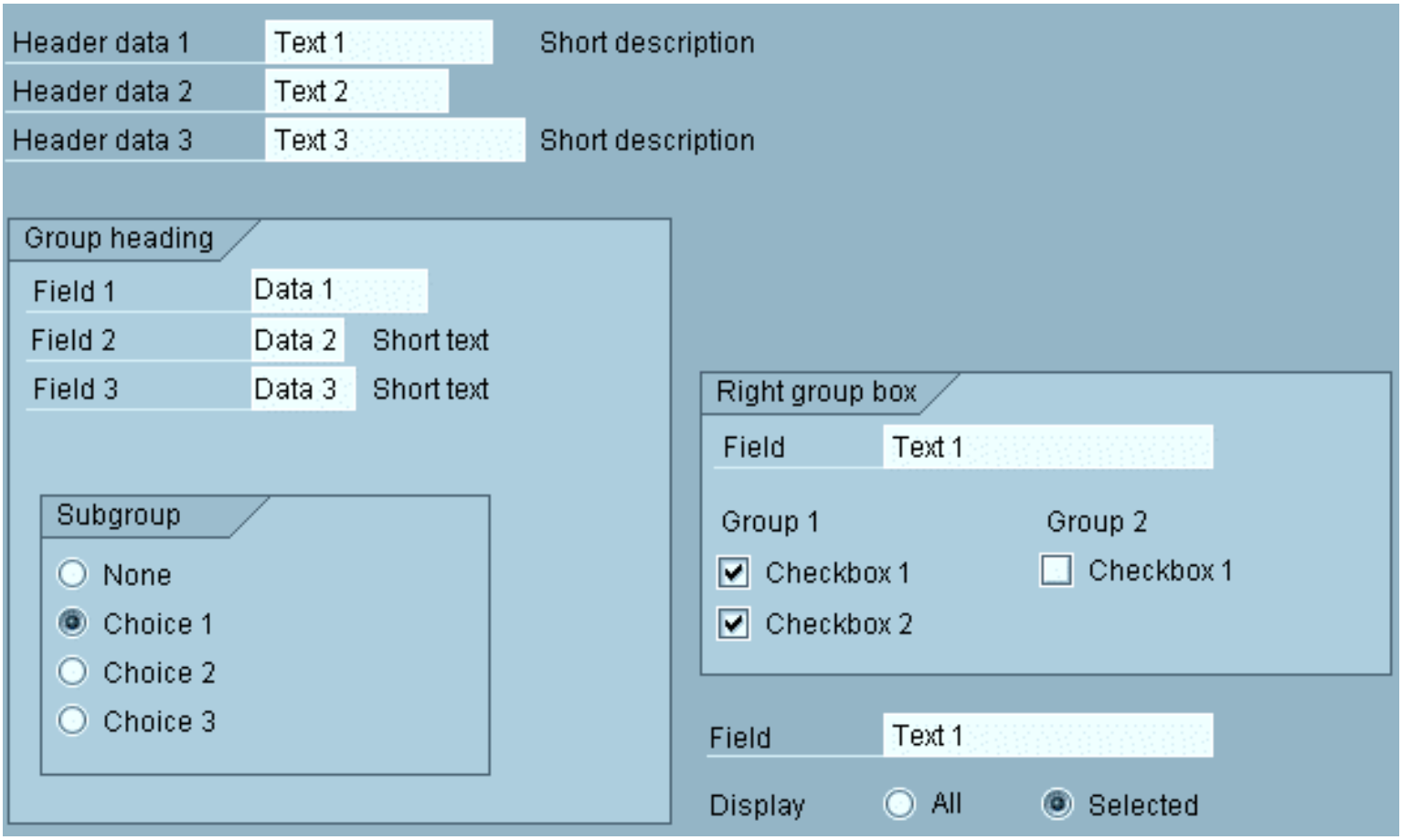


Figure 3: Illustration of guidelines for group boxes on data screens

top

Source: **SAP R/3 Style Guide**

Fields

Arranging Fields and Field Groups

A good screen design essentially depends on the arrangement of the fields within the work area. This document specifies criteria on how you should arrange the fields on the current or adjacent - that is, the previous or next - screen.

There are two general guidelines, you should keep in mind:

- **Keep templates consistent**: Make sure that all colleagues involved in designing the screens for a particular task all follow the same criteria.
- **Efficiency vs. good design**: An efficient dialogue procedure has priority over a pleasant visual design of the work area.

Arranging Fields With Regard to the Screen Sequence

The sequence of screens should correspond to the work process of the user. Place important fields on the first screens of a task.

Complement this by implementing an efficient navigation: Allow the user to address individual screens as variably as possible and to go back and forth in a screen sequence.

A screen in a screen sequence should correspond to an idea or a partial task of the user. In the work area, display only the data that are essential for completing a task. Additional information should only be available on request or on a separate screen.

Arranging Fields in the Work Area

To arrange fields, observe the following criteria:

- If there is a typical hardcopy form, the work area should correspond to this form.
- Position general fields before specific fields.
- Position important fields in the upper left corner of the work area, because the user focusses his or her attention there.
- Generally, fields should appear at the upper left of the work area. On data screens position the first field in the **first** line.
  Exception: On initial screens position the first field in the **third** line.

Arranging Fields in Blocks

Collate groups of field names and their respective I/O fields in blocks.

Both field names and I/O fields are left-aligned. The longest field name of a block group defines where the I/O fields begin.

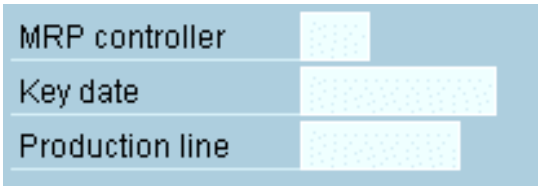


Figure 1: Field names and I/O fields arranged in left-aligned blocks

Arranging Values to be Compared

Position fields whose values the user has to compare with each other one below the other.

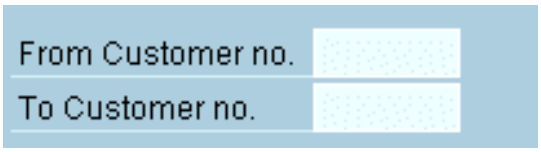


Figure 2: Fields whose values are to be compared are arranged below each other

Exception: If many "From-To" value ranges are grouped close together in the work area of a report selection screen, use the arrangement as shown in the example. Alternatively, the "From" values may be preceded by "From".

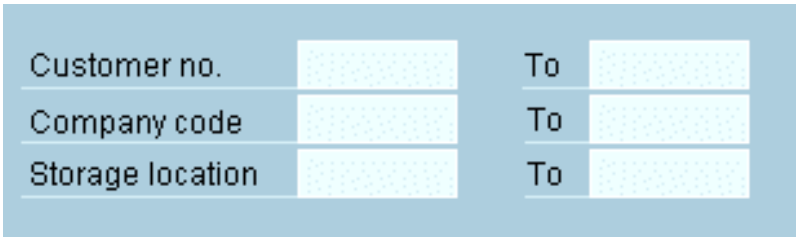


Figure 3: Arrangement of a group of to be compared values

Repeating Name Elements in a Block

If two field names are in part identical, you should still use the full name for both field names.

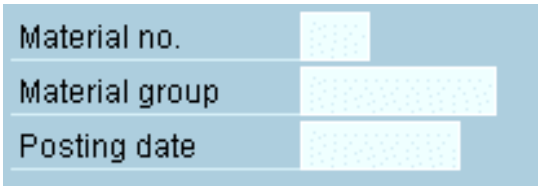


Figure 4: The name element Material is repeated in the block

Arranging Blocks to Form a Block Group

Number of Blocks, Lines and Fields per Block Group - Maximum Number of Block Groups

- In the simplest case, a block group contains only **one** block.
- Limit block groups to **two or three** blocks with six or seven lines at most and a total of ten to twelve input fields.
- If there is a large number of fields, try to group them in two block groups.
- You should have not more than **three or four** block groups on a screen.

Group Heading/Group Box of a Block Group

Provide a block group with a *group box* and identify it with a *group heading*.

You can also include dynamic fields in the heading if these are not too long (less than 15-20 characters).

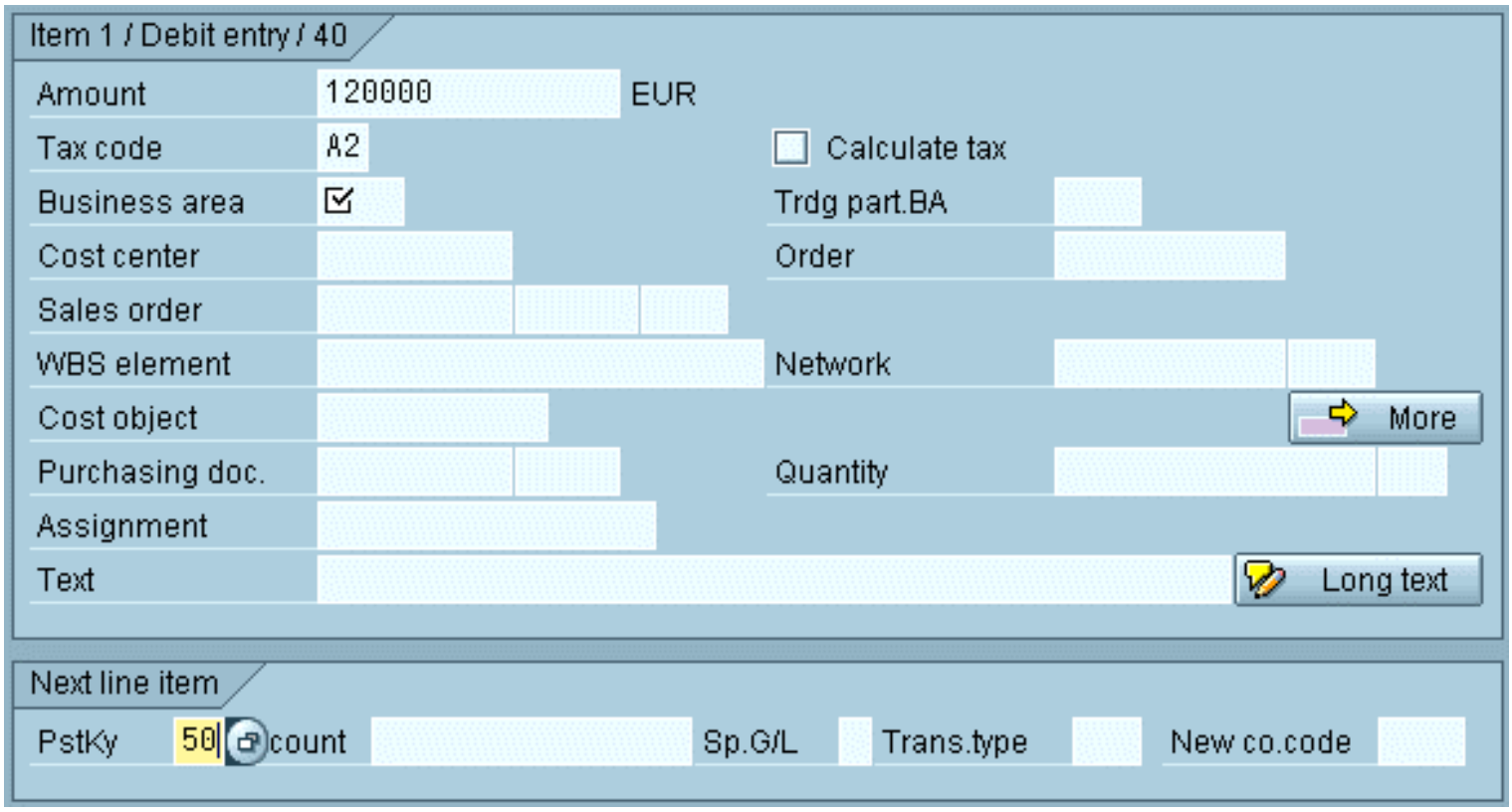


Figure 5: A block in a group box with identifying group heading (with dynamic field)

Exception: Header data is not provided with a group box. You start in the first line of the work area.

Separation of Elements in Blocks, Separation of Blocks

- **Separators in block groups**: Do not use separator lines, neither horizontal nor vertical, within a block group. Use blank lines to separate areas in block groups.
- **Minimum horizontal spacing between two blocks of a block group**: At least five characters between two blocks
  Exception: If the longest field of the left block will most likely not be filled, you can reduce the space between blocks to one character.
- **Maximum horizontal spacing between two blocks of a block group**: Not more than 10 to 15 characters

If you use *group boxes* these start at particularly marked locations, for example, at column 1 or 42. This also contributes to a consistent interface design.

Arranging Several Block Groups on a Screen

Blank Lines Between Block Groups

There is no blank line before a group heading (**Exceptions**: header data, group boxes on initial screens).

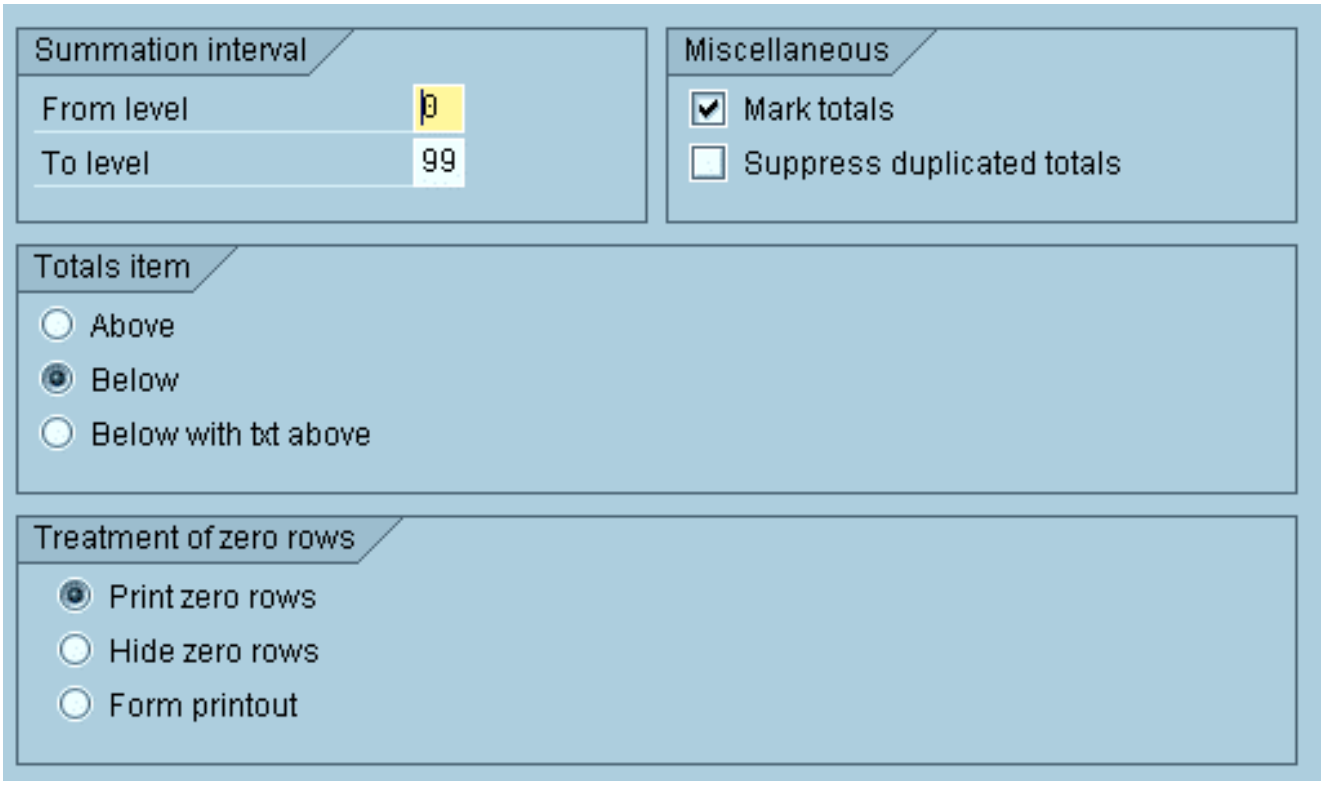


Figure 6: There are no blank lines before group headings

Aligning Blocks in Different Block Groups

Consider, you have block groups consisting of field names, input fields, another block of field names, etc. In this case you need not align the second or third blocks in block groups with the corresponding second or third block in an adjacent block group. You should arrange the blocks optimally within each block group, and then define the spacing for the blocks in the two adjacent block groups. If the distance is less than five to seven characters, you can align the leftmost block with the right-hand block. Otherwise, leave the left block as it is.

Example 1 - Optimizing by Aligning the Right Block Groups

In schematic terms, the fully filled work area originally looks as follows:

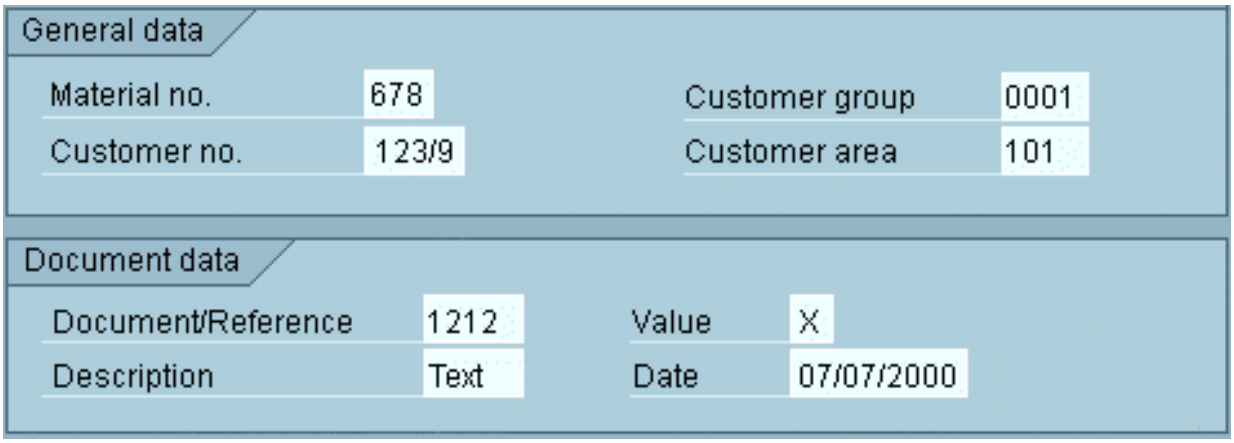


Figure 7a: Bad example for aligning right block groups

In this case, you should align the two right-hand blocks with each other to achieve a more harmonious arrangement:

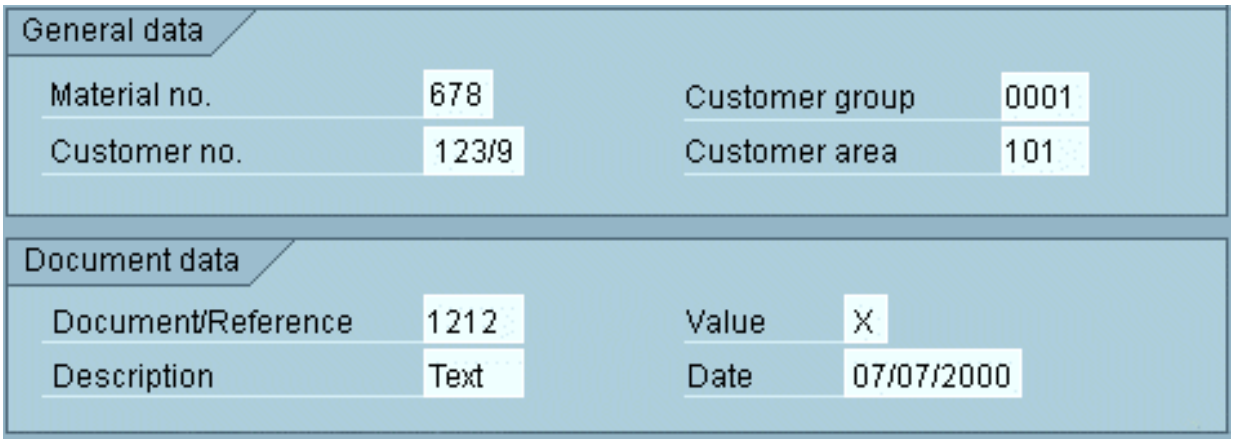


Figure 7b: Good example for aligning right block groups

Example 2 - Do not Align the Right Block Groups

If, schematically, the fully filled screen looks as below, you should not move the lower right block to the right to align it with the upper right block. Leave it close to the lower left group, instead.

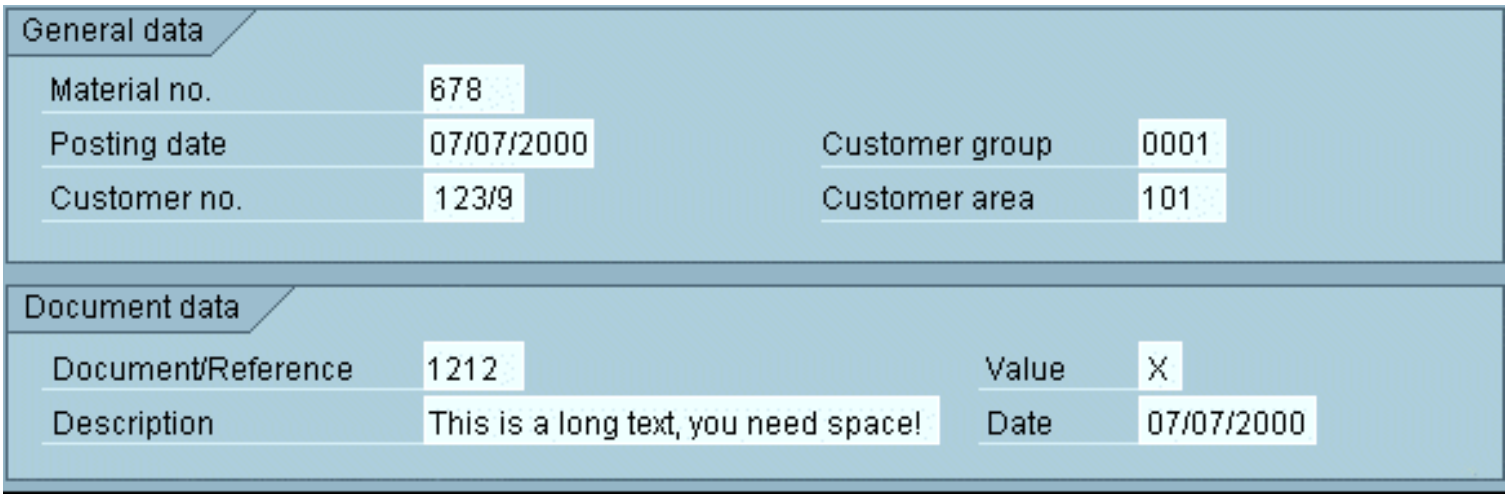


Figure 8: Do not align the right block groups

Priorities

If possible, try to align both field names and input fields of the blocks in different block groups with each other. However, the alignment of I/O fields has priority over the alignment of field names. This applies particularly to the leftmost blocks in the various block groups.

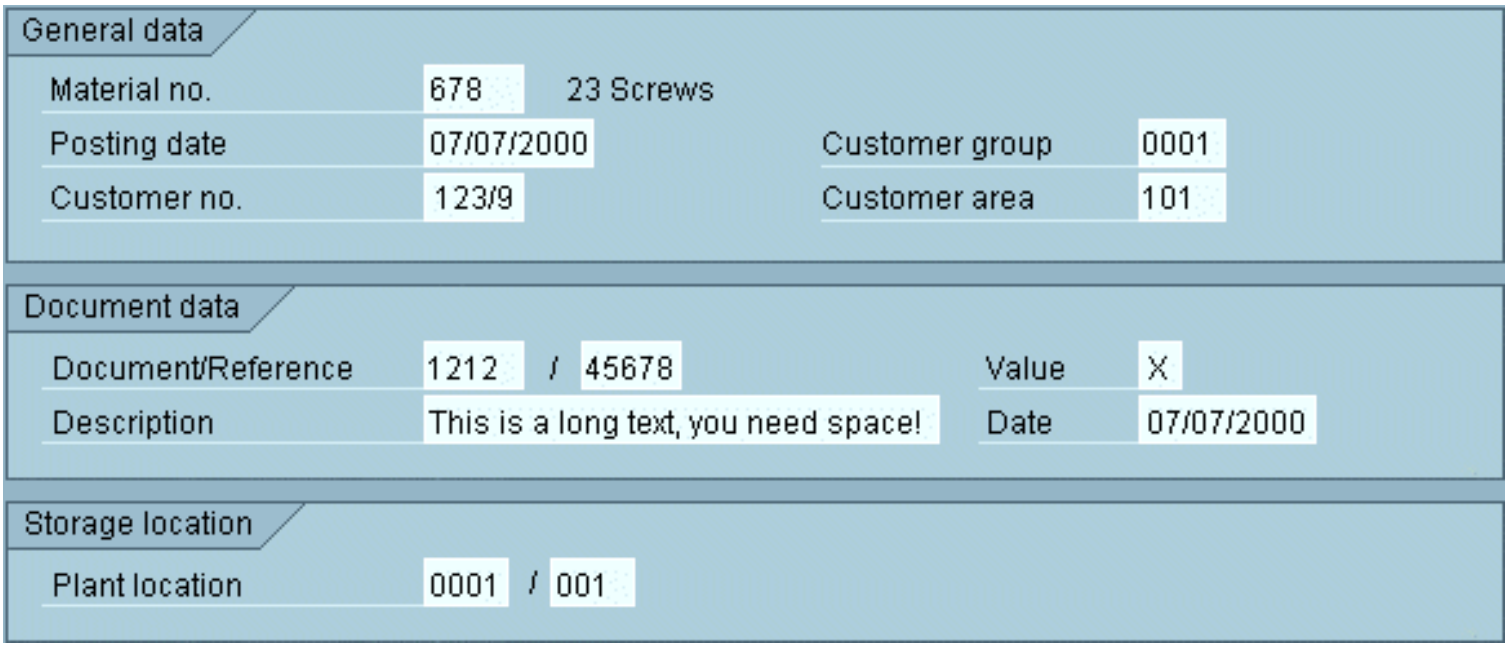


Figure 9: Align input fields instead of field names

In figure 9 the left block groups are left-aligned. The input fields of the right block groups are also left-aligned. The corresponding field names are not left- aligned because one does not want to shorten the field names in the upper block. In addition, the lower block group cannot be shifted further to the left because of the long text field.

top

Source: **SAP R/3 Style Guide**

 Fields

Check Texts

Advantage

Check table or explanatory texts are merely output fields. They usually appear at the end of an input field, and provide a fuller description of the entry; for example, what a numeric or alphanumeric key stands for.
The visible length of an input or output field can thus be smaller than the physical length actually defined by scrollable input and output fields. Users can move the contents of these fields with the mouse or by using the keyboard. This enhancement allows check table texts to be included where there is only a small amount of space available.

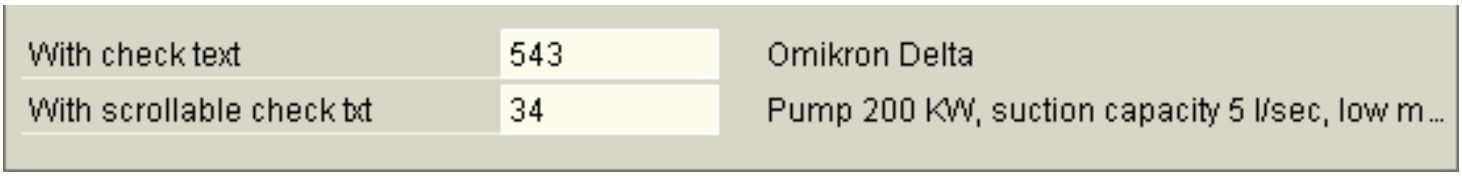
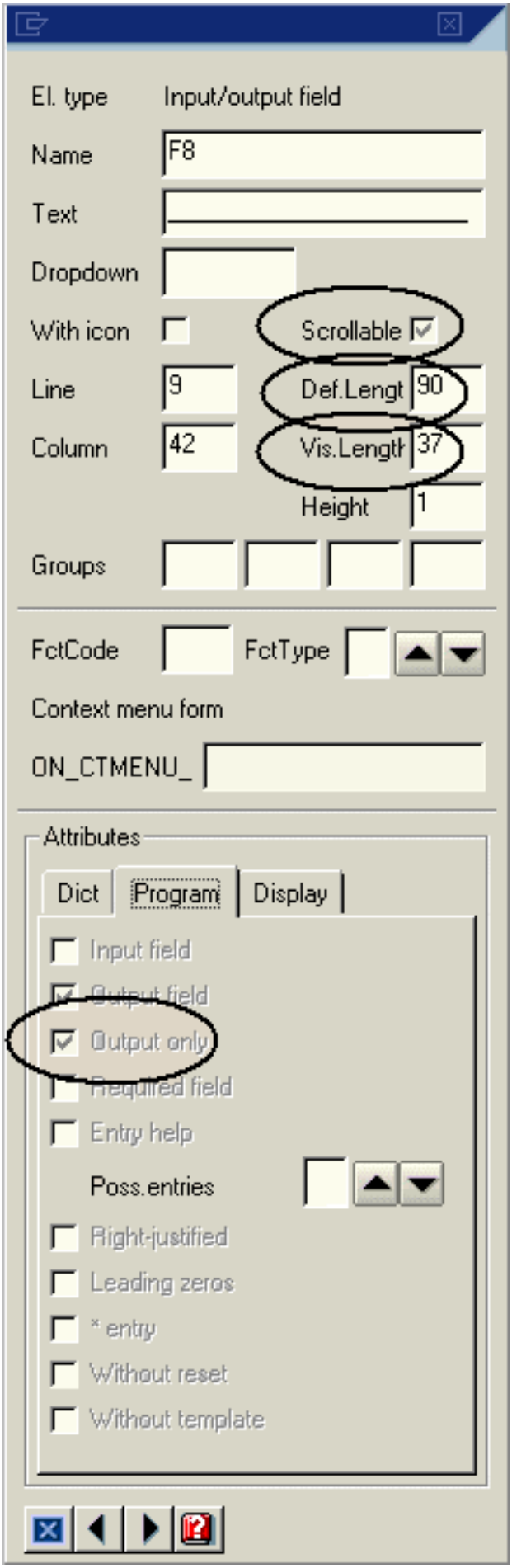


Figure 1: Scrollable check table text can be recognized in the Windows GUI by the gray line at the end of the visible field.

Area of Use

In principle, check table texts should only be available after input fields concerning the entry of an abbreviation or a key. Check

table texts make the picture easier to understand.



Rules for Use

- The visible length of an check table text should make the best possible use of the available space.
- If no Possible entries button ( ) is available, there should be **one** blank character between the check table text and the input field.
- Where there is a temporary possible entries button, insert **three blanks** between the input or output field and the check table text on the right, so that the text is not covered by the possible entries button. (Where there are permanent possible entries buttons, it is already determined in the Screen Painter that the first three characters after a field must be left blank.)
- Check table texts must not be aligned one under the other if this separates them from the input field.

In the following cases check table texts should not be scrollable, but should be shown in full:

- When there is sufficient space for the visible length
- Whenever **date fields**, **units of measure**, and **currency units** are involved

Realization

The Screen Painter (see the adjacent figure) provides the *scrollable* attribute, which can be set for each input and output field. If this attribute is set and the visible length is shorter then the defined length , the field will be scrollable at runtime. Since check table texts are dynamic texts, that is, they are not entered in an output field until runtime, the attribute *Output only* must be set in the Screen Painter for these fields, so that the output is in proportional font.

top

Source: **SAP R/3 Style Guide**

Checkboxes and Radio Buttons

Checkboxes and *radio buttons* are elements for making selections. Checkboxes allow the user to choose items from a fixed number of alternatives, while radio buttons allow the user to choose exactly one item from a list of several predefined alternatives.

Checkboxes

- Checkboxes are used to choose as many options as desired at a time from a limited number of options.
- You can select none, one, or as many options as desired in a group of checkboxes.
- There may also be only one checkbox.
- Choose a name that explicitly distinguishes two different states or contrasts.

Radio Buttons

- Radio buttons are used to choose one option at a time from a limited number of options.
- Arrange radio buttons in groups (you define graphical groups in the Screen Painter). Put at least two radio buttons in one group.
- One radio button in a group must always be selected.
- If the user should have the chance to choose no item from the offered options, create a separate radio button with a "No selection" label or similar text that turns off the other contents-related options.

Arranging Checkboxes and Radio Buttons on Screens

Whenever the following guidelines apply to checkboxes as well as to radio buttons, we speak of "elements". Otherwise we use the respective names.

The Element Refers to Adjacent Input Fields and Cannot be Separated From the Context

Align the elements with the input fields they refer to and place them below the input fields. In the case of radio buttons, you always have to use at least two of them.

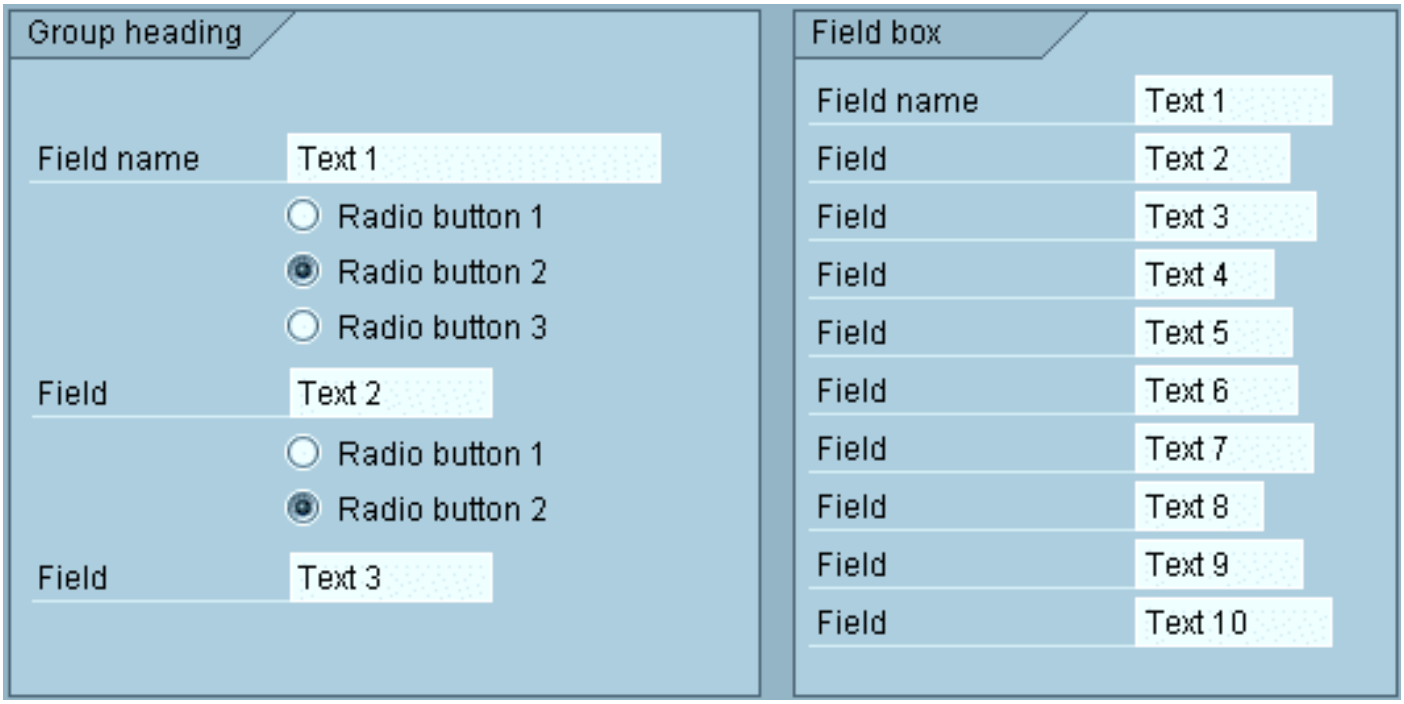


Figure 1: Radio buttons aligned with adjacent input fields

You can also place the elements to the right of the reference field(s) or to the right of one of the reference fields, separated by blanks. If several reference fields are available, place the element to the right of the bottom reference field. Arrange several elements of a group standing to the right in justified form. In the case of radio buttons, you always have to use at least two of them.

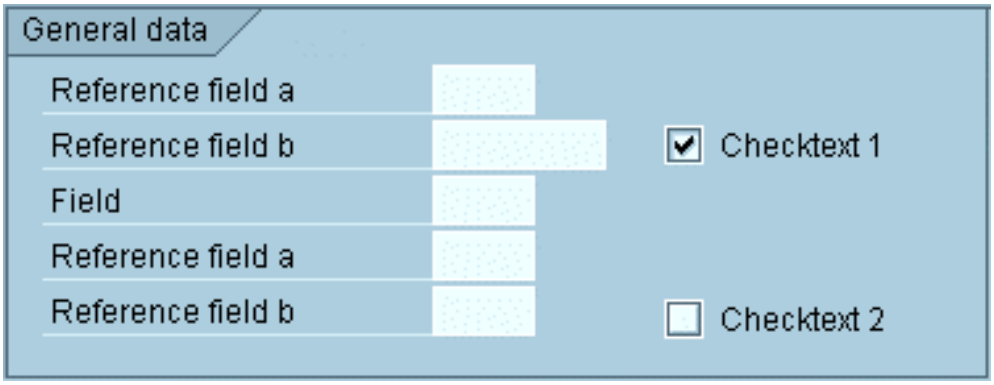


Figure 2: Checkboxes arranged to the right of the reference field

If there is not enough room to implement one of above mentioned solutions, proceed as follows: Place the graphical symbol to the right and left-align it with the input fields above. Left-align names with the field names above.

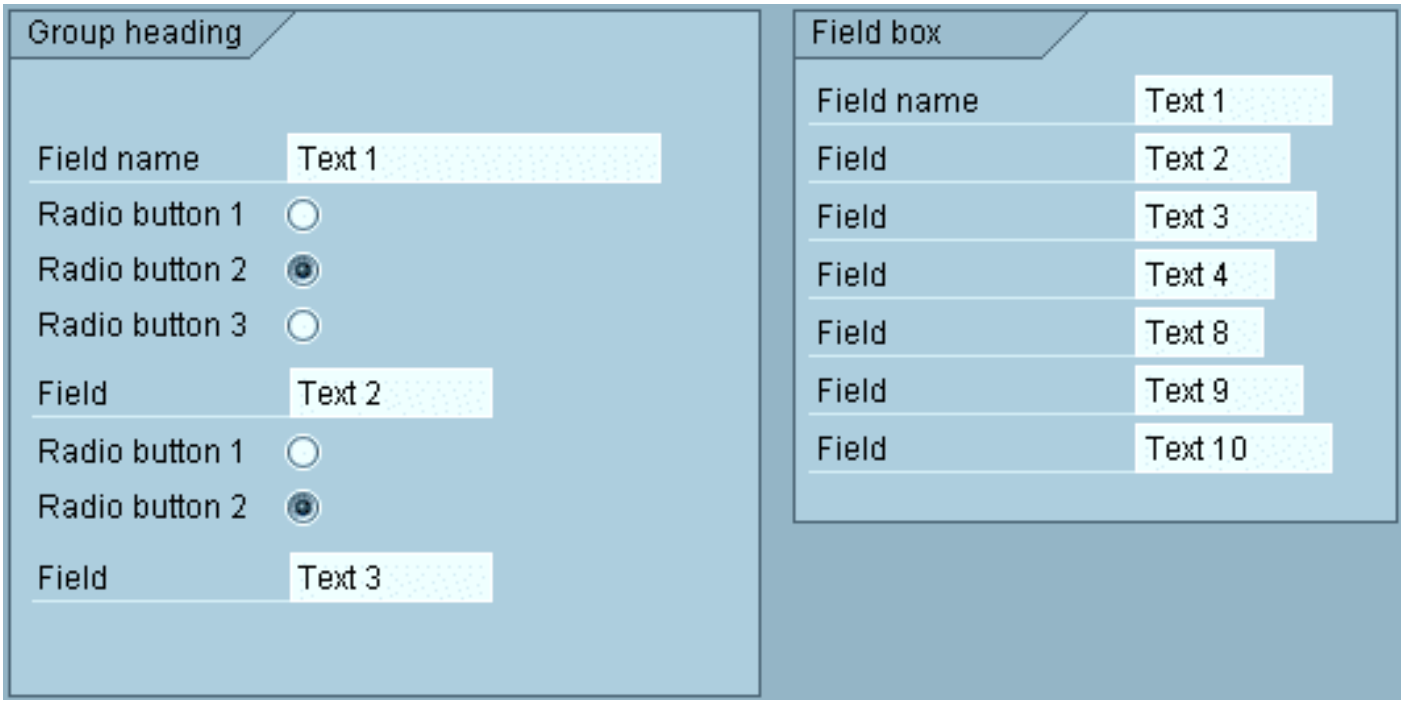


Figure 3: Radio buttons whose labels are left-aligned with the field names

The Element Refers to Several Fields

Elements that refer to several fields are to be left-aligned with the field names to the bottom end of the field group. You can insert a blank line between groups of information units (fields, radio buttons and checkboxes).

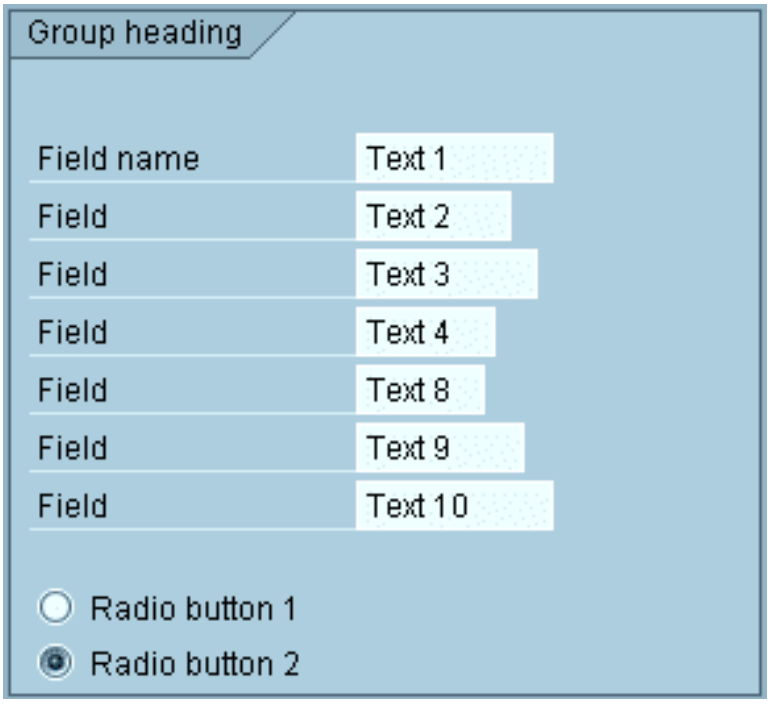


Figure 4: Radio buttons that refer to a group of fields

The Element Represents a Separate Independent Information Block

In this case, place checkboxes as a separate group with or without box next to or below other blocks.

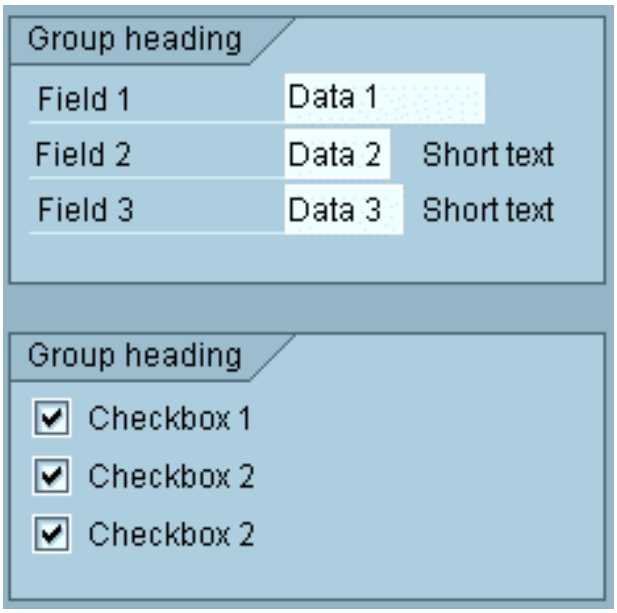


Figure 5: Checkboxes with group box that represent an independent information block

The same applies to radio buttons with one exception: As radio buttons are always related to each other, a group box (with group heading) is mandatory.

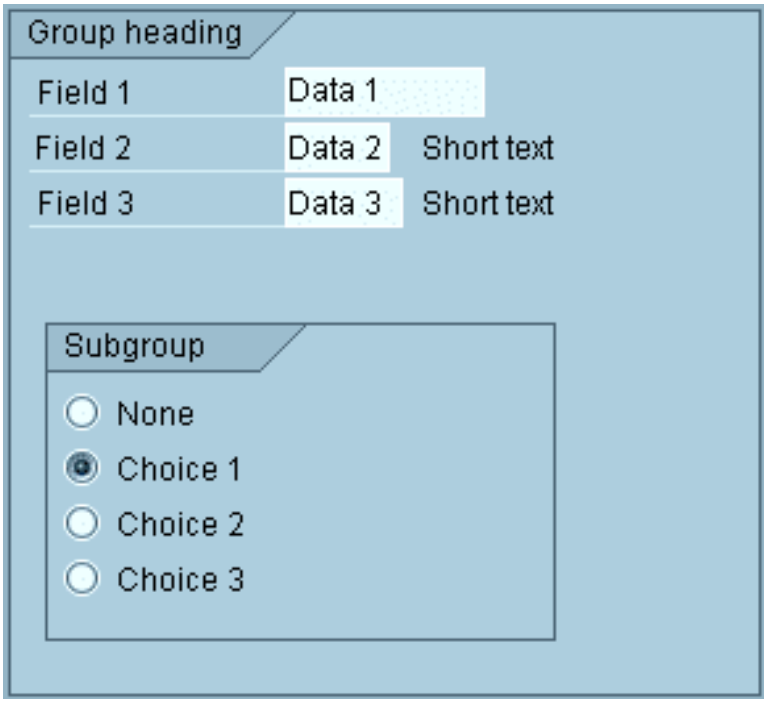


Figure 6: Radio buttons with group box that represent an independent information block

Sufficient Space Horizontally

If room is available horizontally but not vertically, you can position the elements next to each other in one line. In this case, leave at least two blanks between the elements.

You can write a name before the element, if required. Left-align the first element with the input fields.

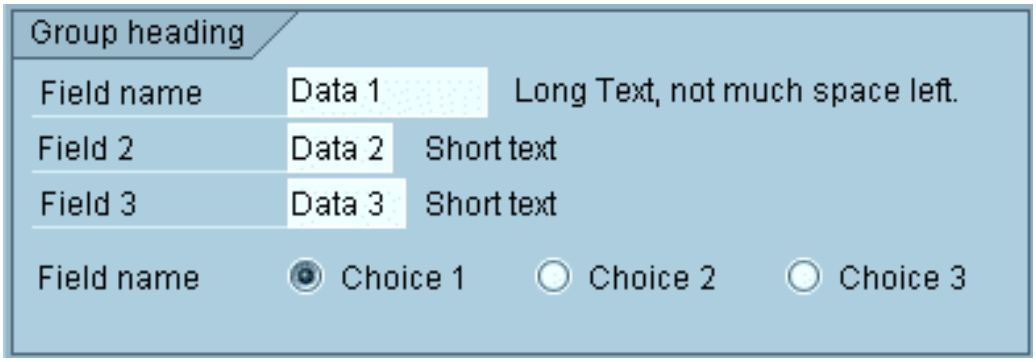


Figure 7: Horizontal arrangement of radio buttons with a descriptive name

If you do not use descriptive names, left-align the first element with the field names.

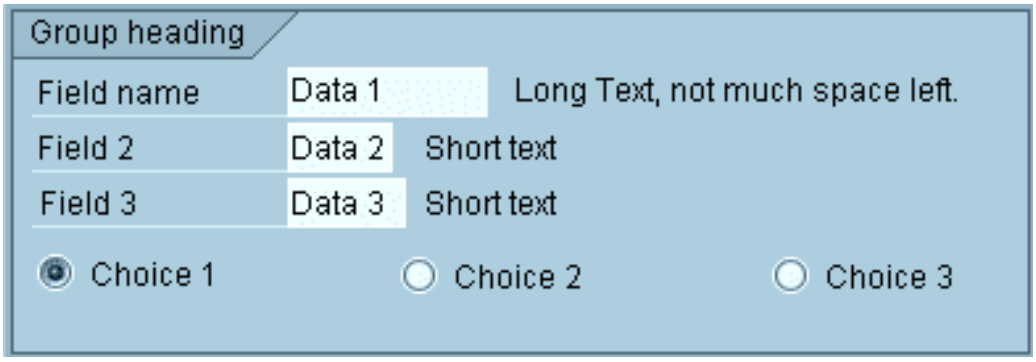


Figure 8: Horizontal arrangement of radio buttons without a descriptive name

top

Source: **SAP R/3 Style Guide**

Selection

Design of Tables or Lists for Making Selections

Multiple-Selection Tables or Lists

In multiple selection tables or lists the user may choose as many options as desired from any number of options.

- The table or list must contain at least two options.
- You can select as many lines of the table or list as you want or none at all.

Multiple-selection tables or lists can be a part of primary windows or appear in a separate dialogue box.

Selection

If several lines can be selected at a time in a table, display the selection fields as checkboxes, positioned to the left of the respective line (Exception: In multiple-line tables checkboxes appear only at the beginnings of logical lines). Put no space between them and the first field of the table entry. Checkboxes can also appear in other columns in the table (horizontally, that is to say) and are then to be used as in matrices.

A selected entry is to be highlighted by the task (Flag INTENSIFIED ON). To achieve immediate highlighting, support the *Select* (F9) functionality in addition to the mouse click. Groups of lines may be selected/deselected by the functions *Select all/Deselect all* and *Select block*.

Arrangement

The entries of the selection list are listed one below the other. Provide scrollbars, if the list exceeds the size of the window.

Column Headings

Provide column headings only used for multiple-column tables or for one-column tables, if the title bar does not explain the table contents. Checkboxes for selecting rows do not have a column heading.

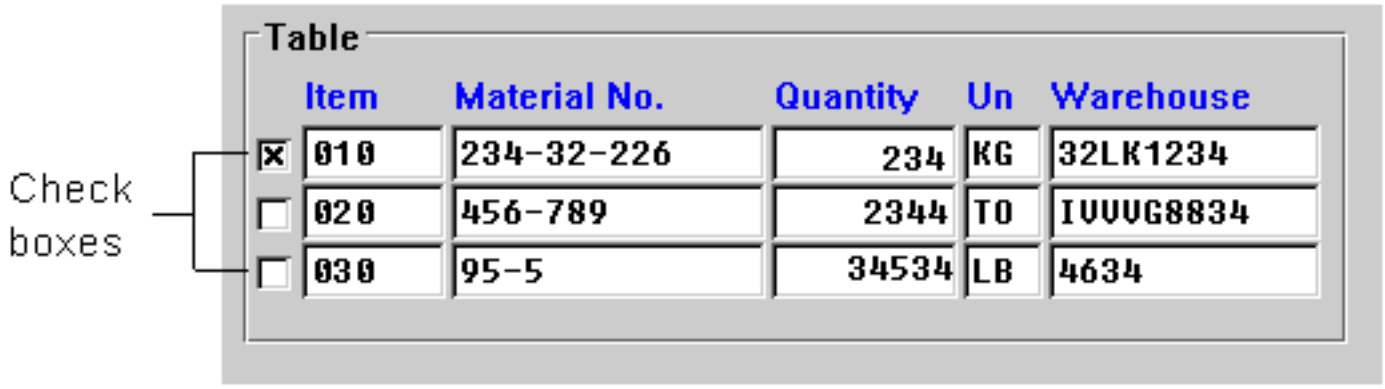


Figure 1: Example of a multiple-selection table with checkboxes; note that the checkboxes themselves do not have a column heading

Edit Flag

Sometimes it is useful to indicate that a table row has already been processed. You can indicate this by putting an asterisk "*" into a separate column, positioned directly before the checkbox. Display the asterisks flat on the screen.

When only one action is possible on the objects of the table, the asterisks can remain even though the user toggles several times between the table and, for instance, a detailed view. The asterisks may not remain, however, when several actions are possible.

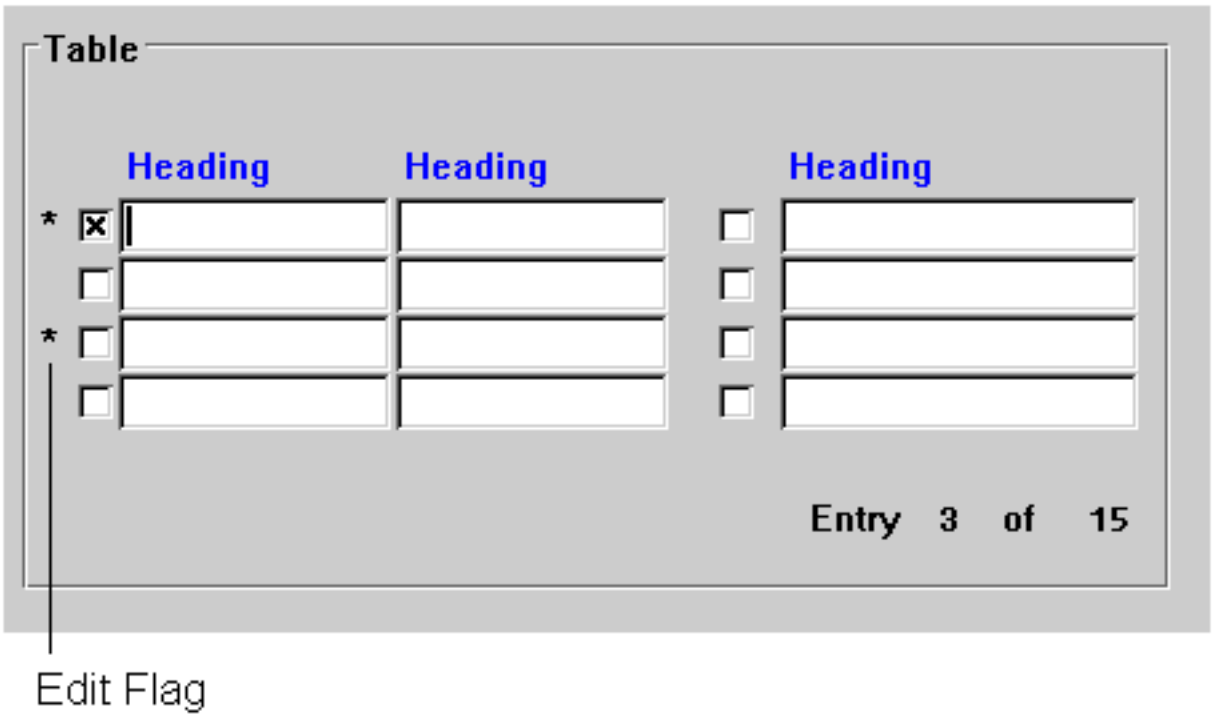


Figure 2: The edit flag is displayed as a column of asterisks

Single-Selection Tables or Lists

Use single-selection tables or lists to choose exactly one item (line) from any number of options.

- The table or list must contain at least two options.
- One option of the table or list has always to be chosen.

Single-selection tables or lists can be part of primary windows or appear in a separate dialogue box.

In addition, the guidelines specified for multiple-selection lists apply.

Selection

Single-selection tables or lists are not ready for input. Entire logical lines are selected.

For one-column or two-column tables, you can place a radio button before the tables to indicate single selection (explicit selection). For large multiple-column tables, radio buttons should currently not be used. Use automatic selection in this case or use the table control.

Matrices

If a list of objects exists and several independent options can be set to each object, list the objects one below the other. The user sets the options by activating adjacent checkboxes. In this case, the text of the option or the checkbox appears as a column heading above the checkboxes. If possible, the checkboxes should appear centered below the column headings.

This guideline also applies to radio buttons. The options of a line must be an affiliated group in the Screen Painter.

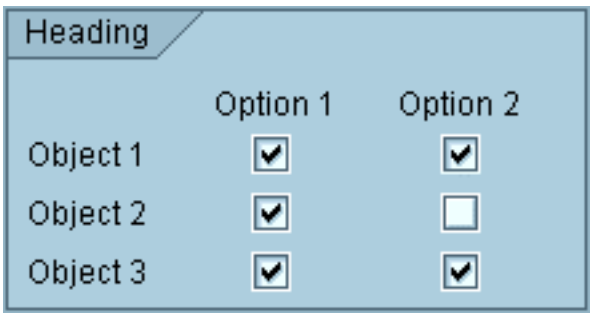


Figure 3: A matrix with checkboxes

SAP Matrix

"SAP matrix" in this context means a table ready for input containing fields, lines and columns as objects the user can select. The user can select as many lines, columns and fields as desired.

Selection

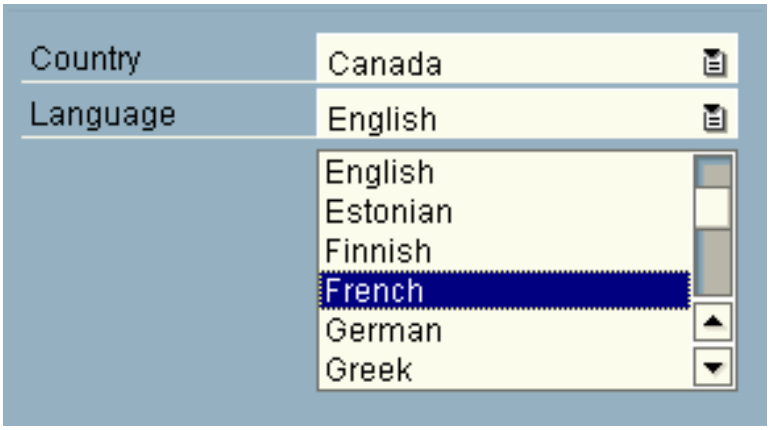
- **Fields**: Explicit selection and deselection. The selected field can then be processed (that is, edited) with any function.
- **Lines**: Same as with the multiple selection. The cursor, however, has to be positioned on the checkbox in front of the lines.
- **Columns**: An entire column is selected and deselected by explicitly selecting and deselecting the column header.

Checkboxes can select lines or fields, depending on the arrangement.

top

Source: SAP R/3 Style Guide

Drop-Down List Boxes



Drop-down list boxes allow the user to choose an entry from a list, which drops down immediately below the field, and shows which values can be chosen. The **drop-down list box** is read-only, while the **drop-down combo box** allows the reader to make an entry in the main field, which is then added to the list if it is not already there. This type of drop-down combo box is not currently supported und is therefore not described in this context.

Display of Text versus Display of Codes

There are two possible types of content in drop-down list boxes: displays of text or displays of codes.
Usually, meaningful texts are displayed in the drop-down list box (for example, "Germany", and not just "D"). If an explanatory text relating to a code is available, then this is always displayed. The codes are then suppressed in the display, but passed to the application once a text has been selected.
However, it also sometimes makes sense only to put the codes into the drop-down list box. In this case, only the code selected is passed to the application.

Operation by the User

Although users cannot make direct entries directly in a drop-down list box, they can select possible entries using either the keyboard or the mouse. Only the first letter of each keyboard entry is evaluated, for example, typing an "E" scrolls down to the first entry beginning with "E"; if you then type another letter, the system scrolls down to the first entry beginning with this letter. Either a value is already preselected in the drop-down list box, or the drop-down list box displays an "empty" entry.

Use of Drop-Down List Boxes

The general use of drop-down list boxes instead of possible entries dialogue boxes is not recommended. Instead, drop-down list boxes should be used selectively where it is possible, for example, to support the selection of a value from a limited quantity more effectively than via possible entries in F4 help. Drop-down list boxes should replace the normal possible entries dialogue box in the following cases:

There are only a certain number of possible entries available (less than 40).
Selection from the drop-down list box menu is hindered by a scroll bar (due to a large number of entries) since, if you position the mouse incorrectly, an entry can be selected inadvertently, or the list can simply click shut without any value being selected. Therefore, scroll bars should only be used in exceptional cases, or avoided completely.
Sorting and limiting values etc. is not supported in drop-down list box menus, so the user is offered no help in managing a large number of values.

Using the drop-down list box must improve usability.

It is inadvisable to convert a field such as "currency key" into a drop-down list box, and to display the text instead. "Currency key" ("DEM", "USD") is a generally recognized code, which can be remembered and entered simply and quickly by the user, whereas the selection or entry of "American Dollars" (instead of "USD") or "Austrian Shillings" (instead of ATS) would be unusual here, and would not be so easily remembered by the user.
This is not the case for SAP-specific codes (company code, posting key, industry sector, material type, ...), which are learned in an SAP-specific context. In other words, they are not generally recognized.

The relationship between code and text must be clear.

 top

Source: SAP R/3 Style Guide

Designing Lists

A Lists Classification Scheme | General Design Guidelines | Highlighting Important Information | Hotspots

Note before reading this topic:

The most of the ABAP lists and some of the table controls were replaced by one of the SAP List Viewer (ALV) components. The ALV components were developed under consideration and observance of all the guidelines concerning lists and tables. For a description and some examples about ALV components have look at the transaction LIBS and the reuse library (transaction SE83). If you use the ALV components you will be on the safe side concerning the guidelines of designing lists.

But there are some restrictions: Not all of the existing types of tables and lists can be replaced by one of the ALV components. Therefore we recommend the following:

- Have a look at this side and check whether your list type can be built by ALV.
- If yes, use the ALV.
- If no, have a look at the linked sides concerning designing lists before creating lists by your own.

General Considerations on Data Structures and Presentation Forms

When designing lists, it is important to present the information in a simple layout that is easy to understand. As an aid, we first discuss general data structures and ways in which they can be presented.

Creating lists means organizing data. The better you organize your lists, the easier the user will understand what the data mean. In the following, we discuss some issues that may help you in organizing your lists.

Data may occur and be organized in endless different ways. Ideally, each data structure should have a corresponding form in that it is always presented. However, since data is often complex and can be viewed in such a different number of ways, there must also be a number of different ways in which they can be presented. This is one of the strengths of the R/3 System, but it makes list design harder for you.

Linear Lists: Single-Line and Multiple-Line Lists

Let us begin with a simple data structure! Often data are just linear lists with all rows having the same column structure. This is the simplest and most important case. Normally, you display such data in a one-line table or list. There are cases, however, where the list is wider than the screen or window width. You have two options for this problem: You can force the user to scroll the list horizontally or you can transform the one-line list into a two-line or even multiple-line list. In the first case, it is important that the most frequently used data columns appear on the left so that the need for horizontal scrolling is minimized. In addition, you can fix data columns - usually you fix the key columns - so that users have horizontal guidance. There are, however cases, where scrolling is not possible or not appropriate. Here, you "fold" the "logical" lines into several "physical" lines, thus creating a multiple-line list. Such lists usually look very disordered. Often you cannot use vertical lines for separating the columns, and you can display fewer logical lines simultaneously on one screen. Therefore, it is very important that you design such lists carefully and try to introduce vertical flight lines. However, it is much better if you avoid three-line or multiple-line lists altogether.

Note: Single-line and multiple-line lists should be replaced by ALV.

Matrices

Sometimes data is not arranged sequentially but according to two or more aspects or variables. With two variables, you get a

rectangular matrix scheme, which is easy to handle (if it is not too wide). With more variables, you are in trouble - the more variables, the more trouble. In this case, you can do the following: You may use nested headings, and/or you may split the matrix into several matrices. In the first case, this may help for up to three variables per axis. It is very likely that your matrix will no longer fit on the screen horizontally. In many cases, it may be better to split the data into several matrices arranged one below the other or on separate screens (a tab control would come in handy in this case). Splitting data makes it more difficult to compare them. Decide on one of these solutions, dependent on what is important for the user!

Note: Matrices could not be replaced by ALV.

Hierarchical-Sequential Lists

This type of data structure has a horrible name, but is nevertheless important to the R/3 System. It is closely related to multi-dimensional matrices as well as to the hierarchical lists discussed below. All these data structures contain some sort of hierarchy that may be displayed either in a tabular or in a tree form. A tabular form is preferable, if the "leaves" of the tree are simple linear lists, as is the case with hierarchical-sequential lists. With matrices, the leaves usually consist of one element only, but the tree is "complete", that is, all combinations of variables have to be displayed. This again calls for a tabular design. Trees are more useful, if the hierarchy is irregularly shaped, is partially visible, and involves a lot of interaction.

As you will see below, hierarchical-sequential lists usually form the cross-section of a multidimensional data cube, where the dimensions create the hierarchical part of the data structure, and the data creates the sequential part of it.

Below you can find guidelines for implementing hierarchical-sequential lists. But keep in mind that there is an alternative solution to this presentation problem: You may also split the list into several simple lists, one for each group, which will follow one below the other or are distributed onto several screens (a tabstrip come in handy).

Note: Hierarchical-sequential lists could be replaced by ALV lists. But check first whether you can present your data by a linear list type.

Hierarchical Lists and Trees: Hierarchical Data Structures

Often, when data forms some sort of hierarchical structure it is best to present it in a tree-like fashion. You can present such data in a format that closely resembles a list (semi-graphics), or you can present it in a more graphical form where you need not observe the formal restrictions of lists. Choose the first case, where space requirements are tight, graphics cannot be used, the hierarchy is not too deeply nested, and if there is some sort of sequential structure imposed on the data (according to the user's point of view!). The Windows file manager is a good example for the latter case. A volume's file hierarchy has also an imposed sequential structure, because the files and directories are sorted to some criteria. For example, they may be sorted by alphabet or by date of the last revision. On the other hand, it has a "variable" hierarchical structure that makes it inappropriate for tabular presentation forms, as files and directories are permanently added and/or deleted. In addition, the file hierarchy is often not too deeply nested. While Windows uses graphics for displaying the file hierarchy, this hierarchy is equally well displayed in the semi-graphic list-style of the ABAP list processor.

However, with hierarchies which are more complicated and/or more deeply nested, you should consider using the R/3 Business Graphics. This tool also provides the graphics in a separate window which may be of advantage in many cases. However, such a presentation form also involves more complexity for the user. So, consider the advantages and disadvantages of either solution, before you use Business Graphics.

For detailed informations see Tree structures.

Nets

Nets are data structures with more complex interconnections than the hierarchical ones in trees. In their simplest form, they are just trees with loops or with only a few cross links between their branches. Here, you can try to split the nets into several trees. In more complicated cases, you have to use a graphics system like the R/3 Business Graphics that is capable of displaying nets.

Note: Nets could not be replaced by ALV.

Data with Little Structure: Object Lists

This case is the most irregular, and it may be the most difficult when you are designing lists. For instance, the data may have a basic sequential structure, but each block of data may have a different structure so that you cannot use a tabular design. In other cases, there may be no structure at all. We can only leave this case to your imagination. We urge you to try to impose some sort of structure onto the data in order to help the user find his or her way through this data chaos.

Note: Object lists could not be replaced by ALV.


top

Source: **SAP R/3 Style Guide**

Designing Lists

A List Classification Scheme

When you design a list, it is useful to look at the following classification scheme. If the list at hand fits into this scheme, your task is much easier, because there are design guidelines for it. Furthermore, lists in the R/3 System will have a uniform look. This helps users to recognize the list structure and to find the relevant information faster.

Header Information

Any list can contain the following header information to describe it in more detail:

- Title
- Object information
- Action performed on the list

Object Information

The term "object information" refers to header data (as displayed on the screens) and detail data. Object information identifies the object for which the list was created. It consists of field name, field value and an optional explanatory text.

Action Performed on Lists

The term "action performed on lists" refers to chosen actions and results. Examples: Sorting, scrolling, and totaling information.

One-Line Lists

One-line lists have lines of identical structure with column titles in a header line. Certain columns can have a key function and identify the lines. Examples of special lines are lines with totals or subtotals, or inserted information the user can display and/or hide. The column header can be structured hierarchically.

Multiple-line Lists

With multiple-line lists, as with multiple-line tables, the line information exceeds one physical line and thus is known as a **logical** line. The header line also extends over several physical lines.

Hierarchical-Sequential Lists

Hierarchical-sequential lists are extracts of a data cube whose dimensions are determined by the number of criteria used. Each combination of criteria is further specified by a number of key figures.

Fixed and/or Extracted Criteria

The extract from the data cube is determined by a number of criteria set to particular occurrences (fixed or extracted criteria). This results in a cube which is defined by the variable dimensions; this creates the list.

Fixed criteria with their occurrences are listed as object information because their occurrences are constant for the entire list.

Variable Criteria

Another group of criteria is varied systematically in its occurrences (variable criteria). These characteristics are the keys.

Internal and External Criteria

- External criteria vary slowly; they create groups of lines within which they are constant. They are listed with their occurrences in hierarchy information described below.
- Internal criteria vary within the groups created by the external criteria and have column headings as the key figures. They are located in the left data columns of the list body while the key figures are listed to the right of them.

Hierarchy Information, Groups

Key figures are often listed in addition to the external criteria. They describe the data further, but are constant within the group. The description of the occurrences of the external criteria with the constant key figures and their values is referred to here as "hierarchy information". It describes the "groups" in the list.

List Bodies, Key Figures

The lines of the lists within the groups consist of the internal criteria and the varied key figures. Their descriptions appear in the column headings, their occurrences and their values fill the list. The lines form a one-line or multiple-line sublist, depending on the quantity and arrangement of the internal criteria and the key figures.

The key figures are the actual data in the list.

Special Cases

Matrix Lists

Matrix lists are a special type of hierarchical-sequential lists; they have both column headers and line headers. They result, for example, from classification via several criteria.

Object Lists

If the internal criteria and the key figures of a list do not determine a matrix or a similarly ordered structure, we talk about an object list.

Hierarchies

Hierarchies which have a list-like form (tables of contents, for example), can also be displayed as lists.

Diagrams, Trees

Furthermore, lists can be displayed as simple diagrams, for example, bar charts or as a tree structure.

 top

Designing Lists

General Design Guidelines for Lists

Clean Up First, Then Use Colors!

You cannot improve poorly structured lists by merely using colors. Try to come up with a clear list structure in the black/white display before you add any colors.

Arrangement

You can enhance legibility by:

- using alignments
- using extensive information units on the screen
- separating information units with blank lines
- inserting blank columns in lines to arrange the information clearly

One-Line vs. Multiple-Line Lists

Avoid multiple-line lists and use one-line lists instead. If you use multiple lines, try to align the subordinate physical lines to achieve a good display design.

Colors

Pre-Considerations

The color palette was developed based on the following considerations:

- Desaturated colors that harmonize with each other were chosen because they are less tiring on the eye.
- Each basic color has two shades (saturated and desaturated). This way, you can structure information much better.

Using Colors

When using colors, keep the following in mind:

- To avoid a mosaic effect, use only a few colors and apply them across large areas. Fill any gaps between colored areas. Extend colored areas beyond the length of the field to the borders of the list.
- Restrict gaps between columns and between the list body and the group box to one character, that is, to the separator line itself.

The List Color Scheme

Logical vs. Physical Colors

The following logical colors are reserved for particular list components. The color setting defines how logical colors look on a certain platform, that is, by assigning specific physical colors to the ABAP list colors.

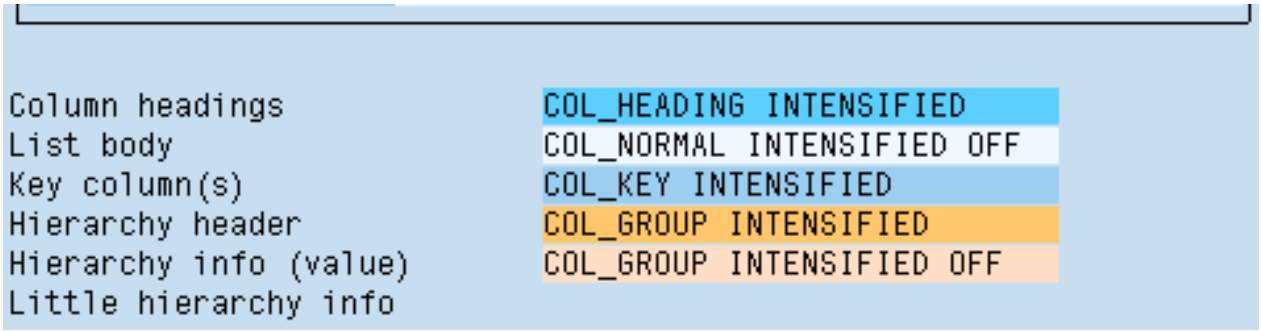


Figure 5: Hierarchical-sequential list with few criteria and key figures

The hierarchy information is placed in front of the actual data. This occurs in the same arrangement as on screens. Use the following colors:

- **Field name of hierarchy information**: COL_GROUP INTENSIFIED
- **Field values and explanatory texts** (if available): COL_GROUP INTENSIFIED OFF stored

Case 2: Hierarchy Information With Many Criteria and Key Figures

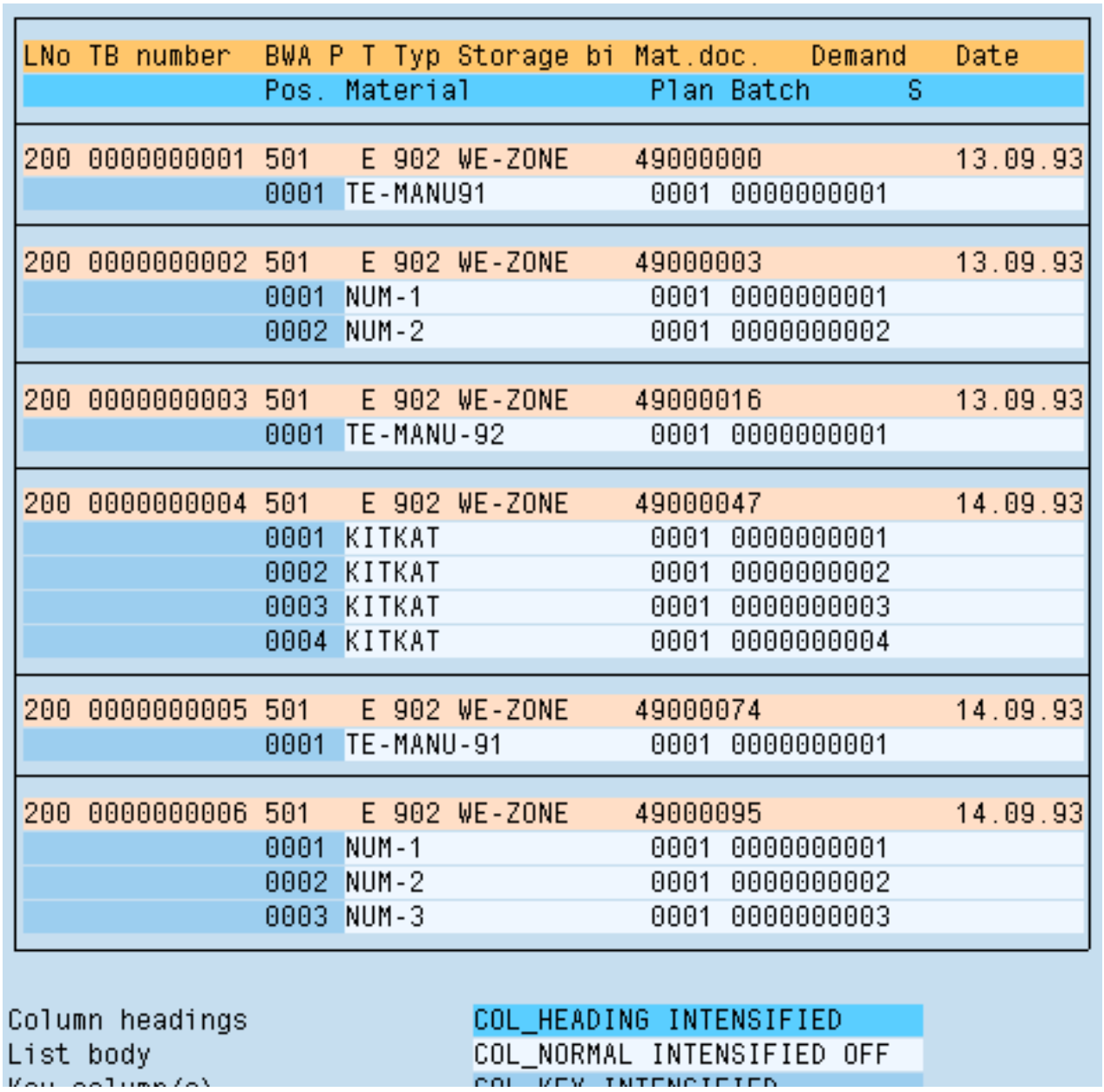


Table 1 also gives the names of the physical colors as delivered with the standard setting. When coloring lists, you should refer to this setting. However, the users can change each of the listed colors to suit individual preferences. Leave the colors COL_POSITIVE and COL_NEGATIVE green or red, anyway, because they represent threshold values.

#	Symbolic Color Name	Intensified	Intensified Off
0	COL_BACKGROUND	Background	Background
		GUI-dependent	*GUI-dependent*
1	COL_HEADING	Headers	Secondary headers
		grayish blue	*bright grayish blue*
2	COL_NORMAL	List body (1. stripe)	List body (standard coloring or every 2nd stripe)
		bright gray	*almost white*
3	COL_TOTAL	Totals, subtotals	Subtotals of higher degree
		yellow	*bright yellow*
4	COL_KEY	Key columns	Highlighting lines/columns
		bluish green	*bright bluish green*
5	COL_POSITIVE	Positive threshold value	Inserted lines
		green	*bright green*
6	COL_NEGATIVE	Negative threshold value	Unused
		red	*bright red*
7	COL_GROUP	Hierarchy header	Hierarchy information
		violet	*bright violet*

Table 1: Color settings

By default, INTENSIFIED is switched on.

The intensive colors (INTENSIFIED) are more saturated that the normal colors (INTENSIFIED OFF). When defining the physical list colors, a gray or white background (COL_BACKGROUND INTENSIFIED OFF) was assumed.

COL_BACKGROUND INTENSIFIED provides blue typeface on the background color to be compatible with the former display in highlighted form. COL_BACKGROUND INTENSIFIED OFF provides black typeface on the background.

Using Colors and Lines in Lists

We start with header information and one-line lists. For the other list categories we mention only the differences.

Note: You find example lists in the transaction LIBS. Information on how to program lines and colors is included in the ABAP online help on FORMAT and WRITE.

Header Information

The guidelines for the header area apply to all list categories.

Title

A title can optionally be issued in the work area (for example, if a report generates several lists).

The title is

- centered
- in COL_BACKGROUND INTENSIFIED
- and followed by a blank line

Object Information

Object information consists of field name, field value and, optionally, an explanatory text.

Object information is displayed

- as a one-column, two-column or multiple-column display as the display on normal screens, depending on whether there is enough space or not
- with no leading characters between field name and field value
- Field name: COL_BACKGROUND INTENSIFIED
- Field value and explanatory text: COL_BACKGROUND INTENSIFIED OFF
- and followed by a blank line

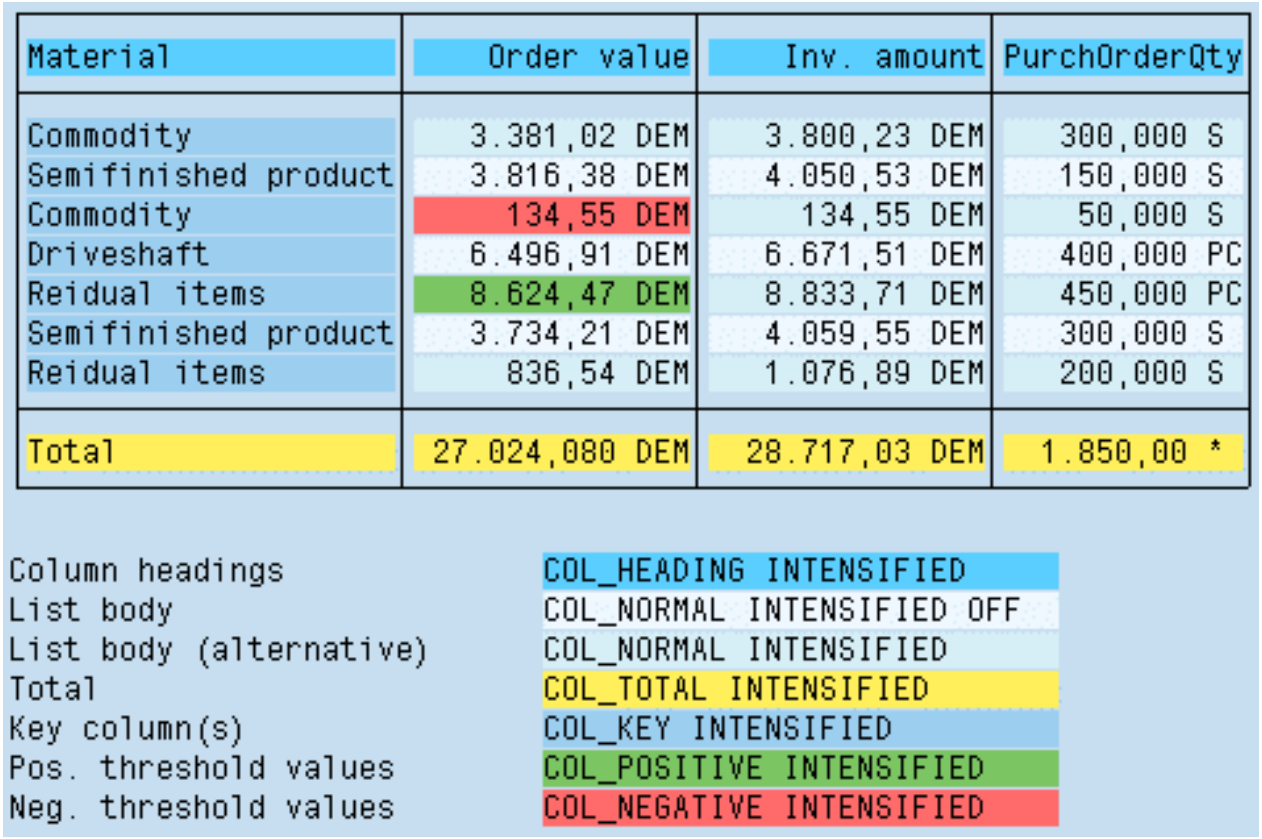
Action Performed on Lists

Action information is displayed as

- action and parameter are next to each other, separated by a colon and a blank character:
  Action: Parameter
  Example: Ascending sorting: Amount
- if there is enough room, in two columns or multiple columns
- in COL_BACKGROUND INTENSIFIED OFF
- and directly followed by the list without blank line

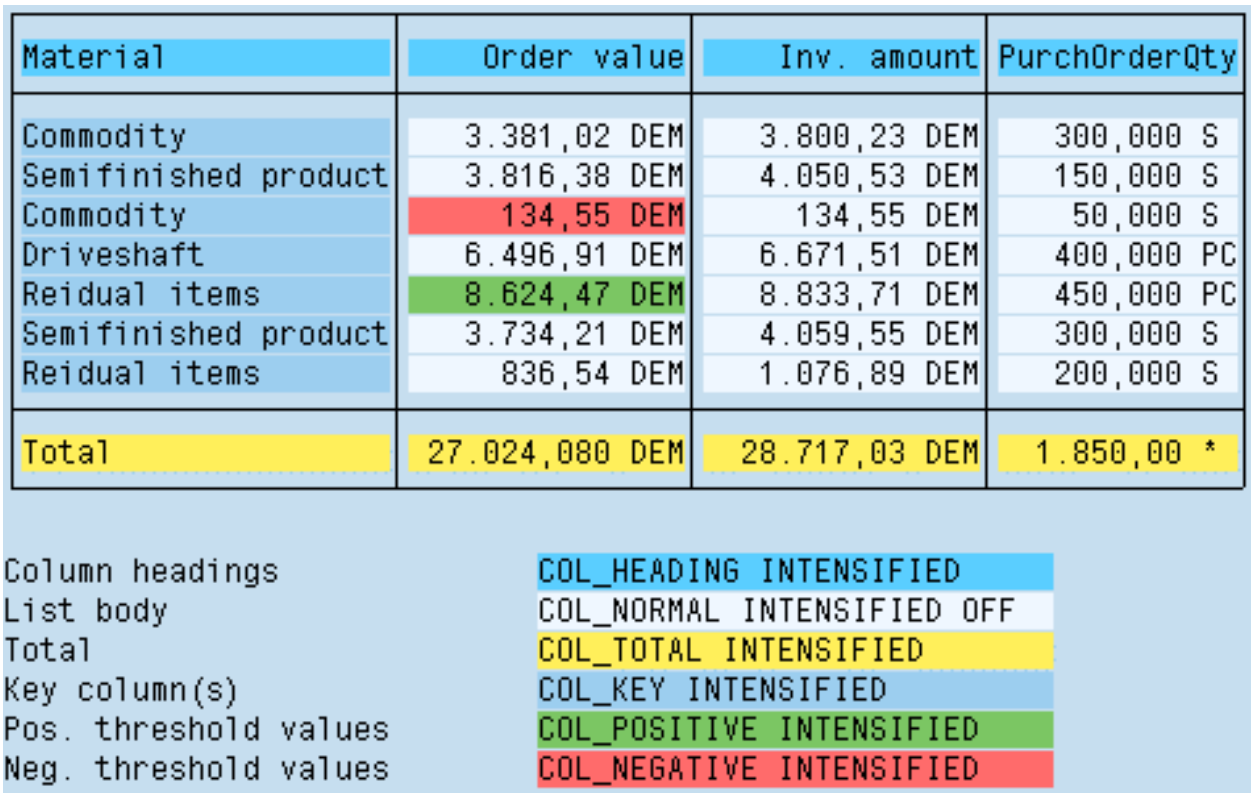
One-Line Lists

The two following illustrations show typical one-line lists.



Material	Order value	Inv. amount	PurchOrderQty
Commodity	3.381,02 DEM	3.800,23 DEM	300,000 S
Semifinished product	3.816,38 DEM	4.050,53 DEM	150,000 S
Commodity	134,55 DEM	134,55 DEM	50,000 S
Driveshaft	6.496,91 DEM	6.671,51 DEM	400,000 PC
Reidual items	8.624,47 DEM	8.833,71 DEM	450,000 PC
Semifinished product	3.734,21 DEM	4.059,55 DEM	300,000 S
Reidual items	836,54 DEM	1.076,89 DEM	200,000 S
Total	27.024,080 DEM	28.717,03 DEM	1.850,00 *

Figure 1: One-line list with striped list body



Material	Order value	Inv. amount	PurchOrderQty
Commodity	3.381,02 DEM	3.800,23 DEM	300,000 S
Semifinished product	3.816,38 DEM	4.050,53 DEM	150,000 S
Commodity	134,55 DEM	134,55 DEM	50,000 S
Driveshaft	6.496,91 DEM	6.671,51 DEM	400,000 PC
Reidual items	8.624,47 DEM	8.833,71 DEM	450,000 PC
Semifinished product	3.734,21 DEM	4.059,55 DEM	300,000 S
Reidual items	836,54 DEM	1.076,89 DEM	200,000 S
Total	27.024,080 DEM	28.717,03 DEM	1.850,00 *

Figure 2: One-line list with uniform list body

Lines

Display one-line lists with a frame border. Separate columns and column titles by vertical lines. Separate column titles from the list body by a horizontal line. If there are hierarchical titles, separate these by horizontal lines as well. In addition, you may separate semantic groups of list rows by horizontal lines, if this increases readability of the list.

Column Headings

Align column headings like the respective data columns, for example: text information is left-aligned, formatted number formats are right-aligned, and column headings of higher levels are centered (compare figure 3).

Display column headings in color COL_HEADING INTENSIFIED.

If there are hierarchical column titles, display subtitles in color COL_HEADING INTENSIFIED OFF.

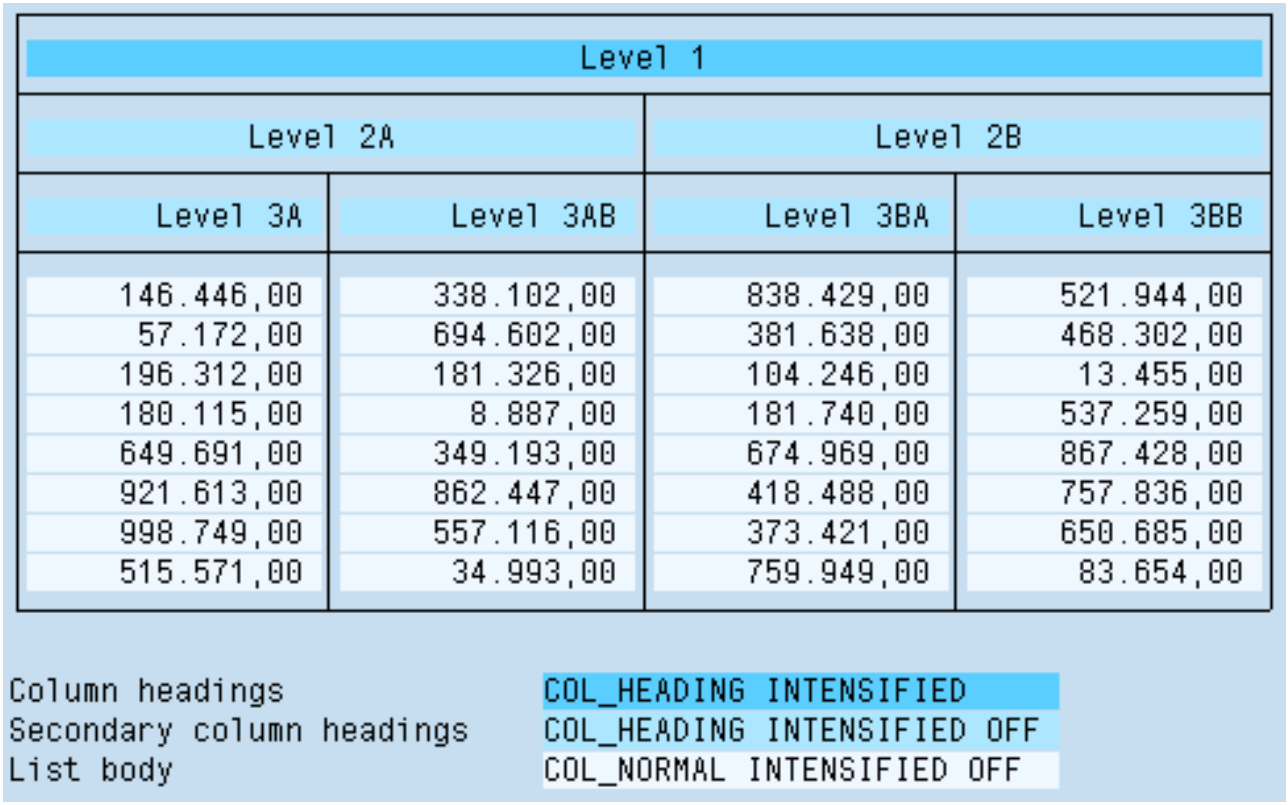


Figure 3: One-line list with hierarchical column headings

List Body

Display the list body in color

- **Standard**: COL_NORMAL INTENSIFIED OFF (uniform)
- **Exception**: If it is important to easily follow the lines or if the list is very wide: alternating COL_NORMAL INTENSIFIED and COL_NORMAL INTENSIFIED OFF (starting with COL_NORMAL INTENSIFIED)

Display key columns in color

- COL_KEY INTENSIFIED (uniform);
- **Exception**: If the list contains too many colors, use the color of the list body (this also applies to all other list categories).

Multiple-Line Lists

For multiple-line lists, the guidelines for one-line lists apply except for the exceptions or modifications described below.

Mat.doc.	Pos.	Material		Plan	Batch	S
TB qty	TR qty	QU	Status%	Endg.		
LNr	200		TB-Nummer		0000000001	
Typ	902		Lagerplatz		WE-ZONE	
49000000	0001	TE-MANU-91		0001	0000000001	
1.464,000	1.464,000	ST	100,00	X		
LNr	200		TB-Nummer		0000000002	
Typ	902		Lagerplatz		WE-ZONE	
49000003	0001	NUM-1		0001	0000000001	
3.381,000	3.381,000	ST	100,00	X		
49000011	0002	NUM-2		0001	0000000002	
8.383,000	8.383,000	ST	100,00	X		
LNr	200		TB-Nummer		0000000003	
Typ	902		Lagerplatz		WE-ZONE	
49000016	0001	TE-MANU-92		0001	0000000001	
5.219,000	5.219,000	ST	100,00	X		
LNr	200		TB-Nummer		0000000004	
Typ	902		Lagerplatz		WE-ZONE	
49000047	0001	KITKAT		0001	0000000001	
6.945,000	6.945,000	ST	100,00	X		
49000053	0002	KITKAT		0001	0000000002	
3.816,000	3.816,000	ST	100,00	X		
49000060	0003	KITKAT		0001	0000000003	
4.683,000	4.683,000	ST	100,00	X		
49000071	0004	KITKAT		0001	0000000004	
1.963,000	1.963,000	ST	100,00	X		
LNr	200		TB-Nummer		0000000005	
Typ	902		Lagerplatz		WE-ZONE	
49000074	0001	TE-MANU-91		0001	0000000001	
1.813,000	1.813,000	ST	100,00	X		
LNr	200		TB-Nummer		0000000006	
Typ	902		Lagerplatz		WE-ZONE	
49000095	0001	NUM-1		0001	0000000001	
1.042,000	1.042,000	ST	100,00	X		
49000104	0002	NUM-2		0001	0000000002	
135,000	135,000	ST	100,00	X		
49000107	0003	NUM-3		0001	0000000003	
1.801,000	1.801,000	ST	100,00	X		

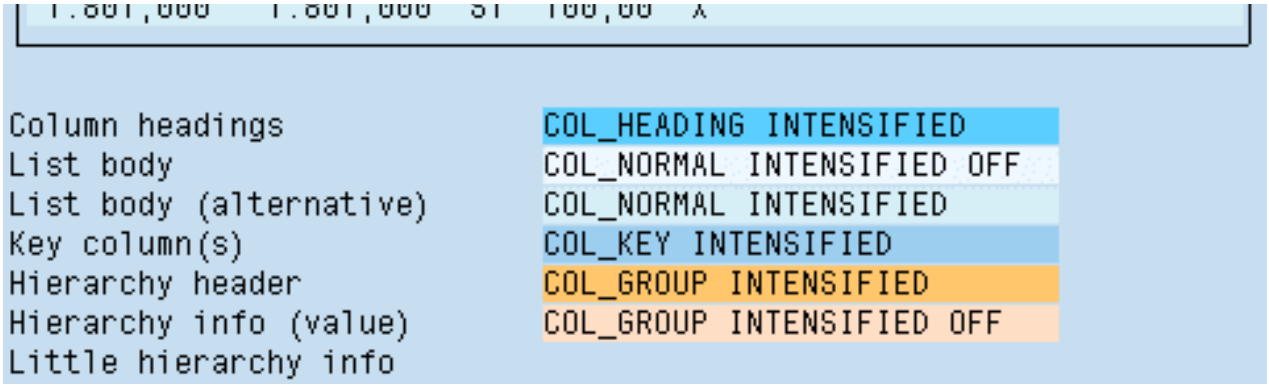


Figure 4: Multiple-line list

Lines

- **Header rows**: Separated from the list body as a whole by a horizontal line below the last physical header row; if it is possible to align the physical table columns, you can define vertical lines between the column headings.
- **Columns**: Usually separated from each other by at least two blanks; if it is possible to align the columns, you can also define vertical lines between the columns.
- **List body**: Do not separate individual logical rows from each other; you can, however, separate semantic groups of logical rows from each other by lines as in one-line lists to increase readability.

List Body

Display the list body with a stripe pattern from logical rows using the colors COL_NORMAL INTENSIFIED alternating with COL_NORMAL INTENSIFIED OFF (starting with COL_NORMAL INTENSIFIED).

In multiple-line lists, no columns are highlighted except for key columns.

You can highlight columns, however, if the columns are aligned with each other and if a continuous highlighting is useful with regard to the contents (see figure 4).

Hierarchical-Sequential Lists

Hierarchical-sequential lists are built from "groups" of lists which are introduced by their respective hierarchy information and put into a single frame border. Within this frame, separate groups from each other by horizontal lines.

Two cases are distinguished:

- The hierarchy information only consists of few criteria and key figures (orientation value: 4 criteria; decisive are transparency and storage requirement):
  Display it just like object information, that is, field name, field value and explanatory text, if necessary.
- The hierarchy information consists of a number of criteria and key figures:
  The descriptions of the criteria and the key figures set up a separate header line on top of the normal header line. Display the corresponding values as normal line information above the list body for the group.

The actual data of a hierarchical-sequential list form a one-line or multiple-line list. With few exceptions, the guidelines for these list types apply to the list body.

Hint: If you already know beforehand that particular criteria do not vary, you should extract these criteria from the list and place them into the header information or into the hierarchy information. The list becomes more compact this way and the display is clearer.

Lines

- **Column headings** are not separated by vertical lines.
- **Columns** in a one-line list body are not separated by vertical lines.
- **Groups**: Each group of lines which is introduced by a hierarchy information is separated from the following group by a horizontal line.

List Body

- **One-line design**: COL_NORMAL INTENSIFIED OFF
- **Multiple-line design**: as multiple-line lists (stripe pattern from logical lines)

Hierarchy Information

As the hierarchy information creates the vertical global structure of the list - it indicates the beginning of a new group - it is highlighted in a special color.

Case 1: Hierarchy Information With Few Criteria and Key Figures

Mat.doc.	Pos.	Material	Plan	Batch	S
LNo	200		TB number		0000000001
Type	902		Storage bin		WE-ZONE
49000000	0001	TE-MANU91	0001	0000000001	
LNo	200		TB number		0000000002
Type	902		Storage bin		WE-ZONE
49000003	0001	NUM-1	0001	0000000001	
49000011	0002	NUM-2	0001	0000000002	
LNo	200		TB number		0000000003
Type	902		Storage bin		WE-ZONE
49000016	0001	TE-MANU-92	0001	0000000001	
LNo	200		TB number		0000000004
Type	902		Storage bin		WE-ZONE
49000047	0001	KITKAT	0001	0000000001	
49000053	0002	KITKAT	0001	0000000002	
49000060	0003	KITKAT	0001	0000000003	
49000071	0004	KITKAT	0001	0000000004	
LNo	200		TB number		0000000005
Type	902		Storage bin		WE-ZONE
49000074	0001	TE-MANU-91	0001	0000000001	
LNo	200		TB number		0000000006
Type	902		Storage bin		WE-ZONE
49000095	0001	NUM-1	0001	0000000001	
49000104	0002	NUM-2	0001	0000000002	
49000107	0003	NUM-3	0001	0000000003	

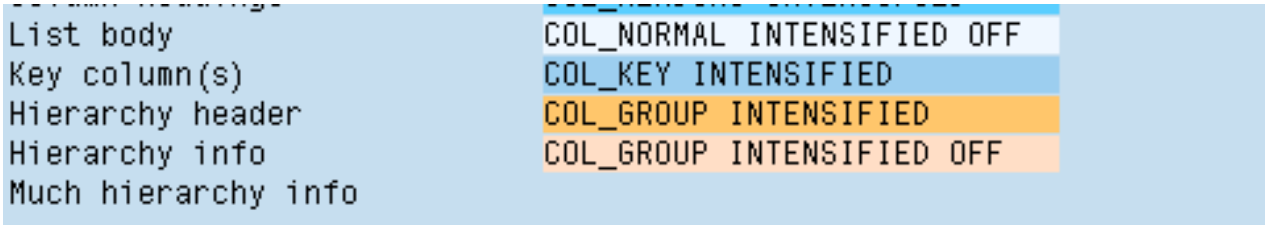


Figure 6: Hierarchical-sequential list with many criteria and key figures

Headers of the Hierarchy Information: Placed before the header line and structured like the column headings, however, with a separate column structure; there is no separator between them.

Color for Headers: COL_GROUP INTENSIFIED.

Hierarchy Information: Placed before the actual data; there is no separator between them.

Color for data: COL_GROUP INTENSIFIED OFF.

Matrix Lists

Column Headings and Row Headings

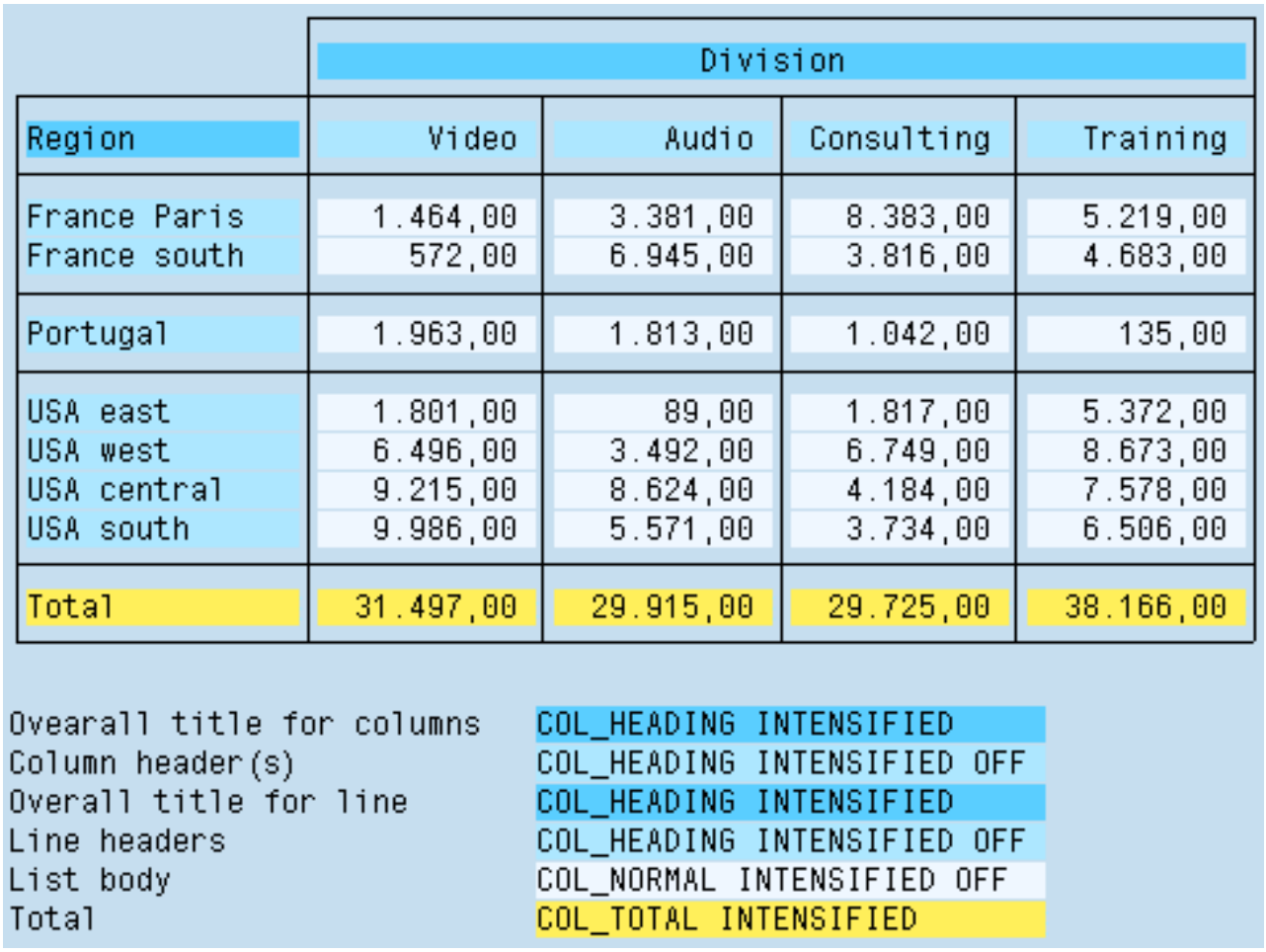
Display column headings and row headings in color

- **Headers**: COL_HEADING INTENSIFIED
- **Subordinate headers**: COL_HEADING INTENSIFIED OFF

Exception: A matrix list may result, for example, from grouping together data. In this case, data which used to be a key are now line headings. Here you can use COL_KEY INTENSIFIED for headers and COL_KEY INTENSIFIED OFF for subordinate headers to avoid changing colors.

List Body

The list body is uniformly displayed in color COL_NORMAL INTENSIFIED OFF.



	Division			
Region	Video	Audio	Consulting	Training
France Paris	1.464,00	3.381,00	8.383,00	5.219,00
France south	572,00	6.945,00	3.816,00	4.683,00
Portugal	1.963,00	1.813,00	1.042,00	135,00
USA east	1.801,00	89,00	1.817,00	5.372,00
USA west	6.496,00	3.492,00	6.749,00	8.673,00
USA central	9.215,00	8.624,00	4.184,00	7.578,00
USA south	9.986,00	5.571,00	3.734,00	6.506,00
Total	31.497,00	29.915,00	29.725,00	38.166,00

Figure 7: Matrix list

top

Source: **SAP R/3 Style Guide**

Designing Lists

Highlighting Important Information

Highlighting Rows or Columns

- **Rows/columns**: COL_KEY INTENSIFIED OFF
- **If required**: COL_KEY to COL_GROUP INTENSIFIED OFF, if they are not used in the list.
- **If no totals are used**: then you may also use COL_TOTAL INTENSIFIED OFF.

Highlighting Totals and Subtotals

Open	Due			
CC BA	Total	up to 8 days	p to 30 days	than 30 days
0001 0000	4.368,00	4.192,00	169,00	7,00
0001 0001	3.528,00	3.473,00	29,00	26,00
0001 0002	1.235,00	982,00	234,00	19,00
0001 0003	128,00	67,00	52,00	9,00
0001 ****	9.259,00	8.714,00	484,00	61,00
0002 0000	922,00	909,00	4,00	9,00
0002 0001	2.098,00	1.746,00	325,00	27,00
0002 0002	5.076,00	4.608,00	434,00	34,00
0002 ****	8.096,00	7.263,00	763,00	70,00
0003 0000	4.041,00	3.789,00	209,00	43,00
0003 0001	2.196,00	1.867,00	279,00	50,00
0003 0002	466,00	175,00	258,00	33,00
0003 0003	2.484,00	2.404,00	42,00	38,00
0003 ****	9.187,00	8.235,00	788,00	164,00
**** ****	26.542,00	24.212,00	2.035,00	295,00

Column headings	COL_HEADING INTENSIFIED
Secondary column headings	COL_HEADING INTENSIFIED OFF
List body	COL_NORMAL INTENSIFIED OFF
Total	COL_TOTAL INTENSIFIED
Subtotal(s)	COL_TOTAL INTENSIFIED OFF
Key column(s)	COL_KEY INTENSIFIED

Figure 1: A one-line list with subtotals and totals

Colors

- **Totals**: COL_TOTAL INTENSIFIED
- **Subtotals**: COL_TOTAL INTENSIFIED OFF
- If subtotal and total are wide apart, you can also use COL_TOTAL INTENSIFIED for subtotals.

Horizontal Lines

- To highlight individual totals and subtotals, separate them from the remaining information by horizontal lines.
- If several similar subtotals are one below another, do not separate them from each other by a horizontal line.
- Separate different categories of subtotals, on the other hand, from each other by horizontal lines.

Vertical Lines

- If you total via several columns, such as totals over the months of a year, separate the totals columns from each other as the data columns by vertical lines.
- If only one total is created, do not use vertical lines in grand totals lines and subtotal lines.

Marking Totals of Different Level

- Totals are additionally marked by "*" in an extra column.
- If you have totals of different levels, then use the corresponding amount of asterisks: "***" means the total of the 3rd level.
- If more than 5 levels are used, it is difficult to distinguish the different numbers of asterisks. Therefore opt for the following display:
  *(n)with n = level, for example, *(7) for a total of the 7th level

Highlighting Inserted Information

In some lists, it is possible to choose a row and to show its information in detail by inserting a number of rows into the list. This insertion can also be carried out on several levels. For indication inserted rows use the color COL_POSITIVE INTENSIFIED OFF.

Note: You should refrain from using a multiple-level colorization here because you do not know the number of levels that have to be included.

Highlighting Cells

Use the following colors for highlighting individual cells:

- COL_POSITIVE INTENSIFIED (green, for example, positive threshold value)
- COL_NEGATIVE INTENSIFIED (red, for example, negative threshold value)
- Other colors, if available: COL_TOTAL, COL_KEY or COL_GROUP (in each case INTENSIFIED OFF)

Caution: COL_POSITIVE and COL_NEGATIVE are assigned in the default setting to shades of green and red and thus have a predefined meaning.

Priority of Colors

If lines as well as columns of a table are marked in color, conflicts between colors occur on the screen because color attributes cannot be combined and only one color can be displayed.

Do not interrupt the highlighting of key columns and totals or subtotals (totals and subtotals have even priority over keys). In

addition, colors for highlighting cells have priority over other colorings.

 top

Source: SAP R/3 Style Guide

Designing Lists

Hotspots in Lists

Hotspots are areas on lists the user can click on to trigger an action. They serve to simplify interaction with lists.

Appearance

Hotspots are flat on the template. When the user positions the cursor on a hotspot, the cursor takes on the shape of a hand. Since this is the only way for the user to recognize a hotspot on the screen, we recommend to indicate hotspot areas additionally with a specific color.

Area of Use

- **Hypertext**: Hotspots are primarily used in hypertext links. Hotspots are to be highlighted in the text (in future underlining of links should be preferred).
- **Hierarchies**: Use hotspots in hierarchical lists to expand or collapse information. One example is the hierarchical graphic, another inserted information.

  If objects that can be expanded and collapsed should remain in a selected state for other actions, precede the object with an appropriate symbol, for example, a folder symbol, or use the Expand/Collapse icons.

- **Detail**: You can use hotspots similar to the present F2 functionality.

Restrictions of Use

Do not use hotspots to replace pushbuttons with no fixed position. You cannot implement hotspots as pushbuttons, because their appearance does not meet the corresponding criteria. Hotspots implemented as pushbuttons would rather confuse than support the user.

Because on some platforms hotspots are triggered by the mouse-down event, use hotspots only for short-running actions which the user can undo easily and which do not change the database!

More Information

For information on programming hotspots, please refer to the online documentation in the ABAP Editor under help hotspot.

top

Source: SAP R/3 Style Guide

Designing Tables

Design of Tables or Lists for Making Selections

In general, it is possible to create one- and multiple-line tables. But: We recommend to avoid multiple-line tables

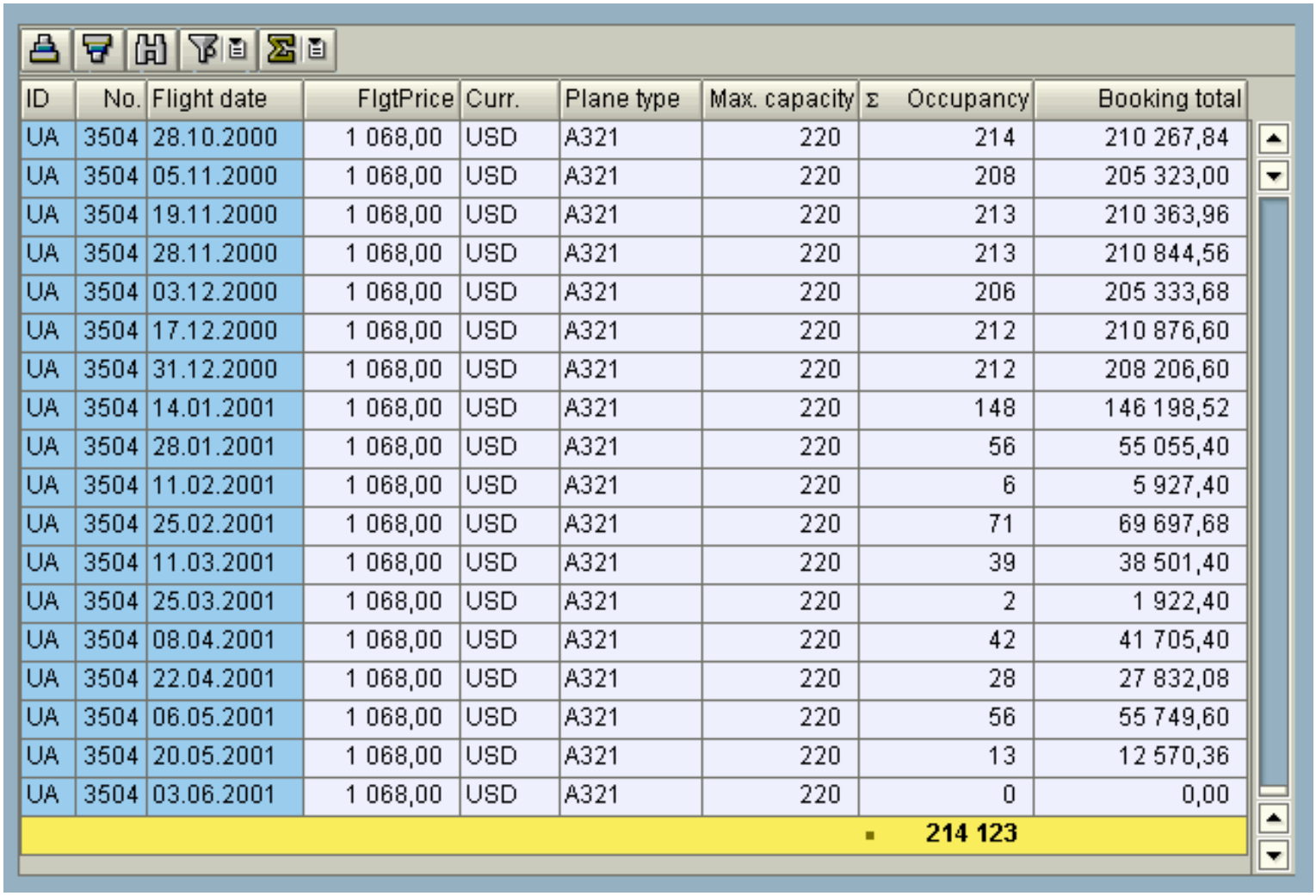
- by transferring these to several one-line tables or
- by providing only one data row, with further data being displayed on a "normal" template by choosing of a table entry.

In the following discussion we are only talking about one-line tables.

For displaying table-like structure there are two techniques which can be used by developers

- the table control
- the SAP List Viewer (ALV) components preferable the ALV grid control which replaces the table control more and more

The ALV Grid Control



ID	No.	Flight date	FlgtPrice	Curr.	Plane type	Max. capacity	Σ Occupancy	Booking total
UA	3504	28.10.2000	1 068,00	USD	A321	220	214	210 267,84
UA	3504	05.11.2000	1 068,00	USD	A321	220	208	205 323,00
UA	3504	19.11.2000	1 068,00	USD	A321	220	213	210 363,96
UA	3504	28.11.2000	1 068,00	USD	A321	220	213	210 844,56
UA	3504	03.12.2000	1 068,00	USD	A321	220	206	205 333,68
UA	3504	17.12.2000	1 068,00	USD	A321	220	212	210 876,60
UA	3504	31.12.2000	1 068,00	USD	A321	220	212	208 206,60
UA	3504	14.01.2001	1 068,00	USD	A321	220	148	146 198,52
UA	3504	28.01.2001	1 068,00	USD	A321	220	56	55 055,40
UA	3504	11.02.2001	1 068,00	USD	A321	220	6	5 927,40
UA	3504	25.02.2001	1 068,00	USD	A321	220	71	69 697,68
UA	3504	11.03.2001	1 068,00	USD	A321	220	39	38 501,40
UA	3504	25.03.2001	1 068,00	USD	A321	220	2	1 922,40
UA	3504	08.04.2001	1 068,00	USD	A321	220	42	41 705,40
UA	3504	22.04.2001	1 068,00	USD	A321	220	28	27 832,08
UA	3504	06.05.2001	1 068,00	USD	A321	220	56	55 749,60
UA	3504	20.05.2001	1 068,00	USD	A321	220	13	12 570,36
UA	3504	03.06.2001	1 068,00	USD	A321	220	0	0,00
							214 123	

Figure 1: The ALV grid control with it's adaptable toolbar

The ALV components were developed under consideration and observance of all the guidelines concerning lists and tables. For a description and some examples about the ALV components see the transaction LIBS and the reuse library (transaction SE83). If you use the ALV components

you will be on the safe side concerning the guidelines of designing tables.

Selecting Objects in Tables

Terms

- **Objects**: Objects of a table are the screen elements that the user can select and edit. Typical objects of a table are individual cells, entire lines or an entire column. "A line" here means an entire logical line that is both physical lines for double-spaced tables.
- **Select/Deselect**: To "select" an object means to "activate" an object for further processing. The selected object will be affected by any action the user initiates subsequently. For example, if the user chooses the function *Delete* the object will be deleted. The term for cancelling a selection is "deselect". Usually the selected object is highlighted, but see for exceptions below!
- **Single vs. Multiple Selection**: If only one item is involved, we talk about "single selection". If several items are involved, we talk about "multiple selection".

General Guidelines for Making Selections

Explicit Selection and Deselection

- **With the Keyboard**: The user positions the active control indicator on the desired object and presses the *Select* key.
- **With the Mouse**: The user positions the mouse pointer on an object and presses the mouse button.

The selection state of the entire object changes: If the object is not selected, it is selected now and vice versa.

Automatic Selection

Sometimes it is more efficient to select an object by positioning the cursor on it. With character-specific user interfaces, the selection is indicated merely by the cursor position. On graphical user interfaces, the mouse-based automatic selection also has to be linked with the display of the selection state.

- **With the Keyboard**: The user positions the cursor on the required object. The object is thereby selected (currently merely by the position of the cursor).
- **With the Mouse**: Automatic mouse selection is identical to explicit mouse selection.

The user can now directly execute an action on an object or select further objects, if multiple selection is possible.

Priority of the Principles

Explicit selection is the basic selection method and has priority over automatic selection. It must always be supported for both the keyboard and the mouse. You can replace explicit selection by automatic selection, if appropriate.

Cursor-based automatic selection in a table is only possible, if the user has not explicitly selected an object. If the user positions the mouse pointer on an explicitly selected object and presses the mouse button, the object in question is deselected and not automatically selected.

Indicating Selection

Selected objects have to be visually distinguishable from non-selected objects. In the R/3 System, selected objects currently must have the attribute "highlighted". However, currently selection with a single mouse click is not supported locally; therefore selection may only be indicated by cursor position.

Positioning the Cursor

If only the table is displayed on a screen, or if it is its main element, you should position the cursor automatically on the first object when the screen is first displayed.

If the user has branched from a table to the details of a line and then returns to the table, position the cursor on the object of which the user has last viewed the details.

Designing Tables

Design of Tables or Lists for Making Selections

Multiple-Selection Tables or Lists

In multiple selection tables or lists the user may choose as many options as desired from any number of options.

- The table or list must contain at least two options.
- You can select as many lines of the table or list as you want or none at all.

Multiple-selection tables or lists can be a part of primary windows or appear in a separate dialogue box.

Selection

If several lines can be selected at a time in a table, display the selection fields as checkboxes, positioned to the left of the respective line (Exception: In multiple-line tables checkboxes appear only at the beginnings of logical lines). Put no space between them and the first field of the table entry. Checkboxes can also appear in other columns in the table (horizontally, that is to say) and are then to be used as in matrices.

A selected entry is to be highlighted by the task (Flag INTENSIFIED ON). To achieve immediate highlighting, support the *Select* (F9) functionality in addition to the mouse click. Groups of lines may be selected/deselected by the functions *Select all/Deselect all* and *Select block*.

Arrangement

The entries of the selection list are listed one below the other. Provide scrollbars, if the list exceeds the size of the window.

Column Headings

Provide column headings only used for multiple-column tables or for one-column tables, if the title bar does not explain the table contents. Checkboxes for selecting rows do not have a column heading.

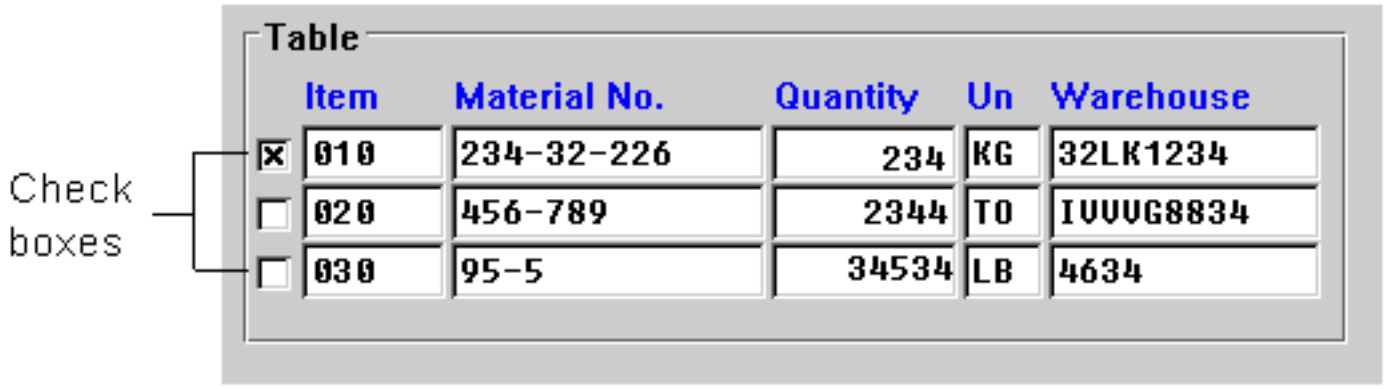


Figure 1: Example of a multiple-selection table with checkboxes; note that the checkboxes themselves do not have a column heading

Edit Flag

Sometimes it is useful to indicate that a table row has already been processed. You can indicate this by putting an asterisk "*" into a separate column, positioned directly before the checkbox. Display the asterisks flat on the screen.

When only one action is possible on the objects of the table, the asterisks can remain even though the user toggles several times between the table and, for instance, a detailed view. The asterisks may not remain, however, when several actions are possible.

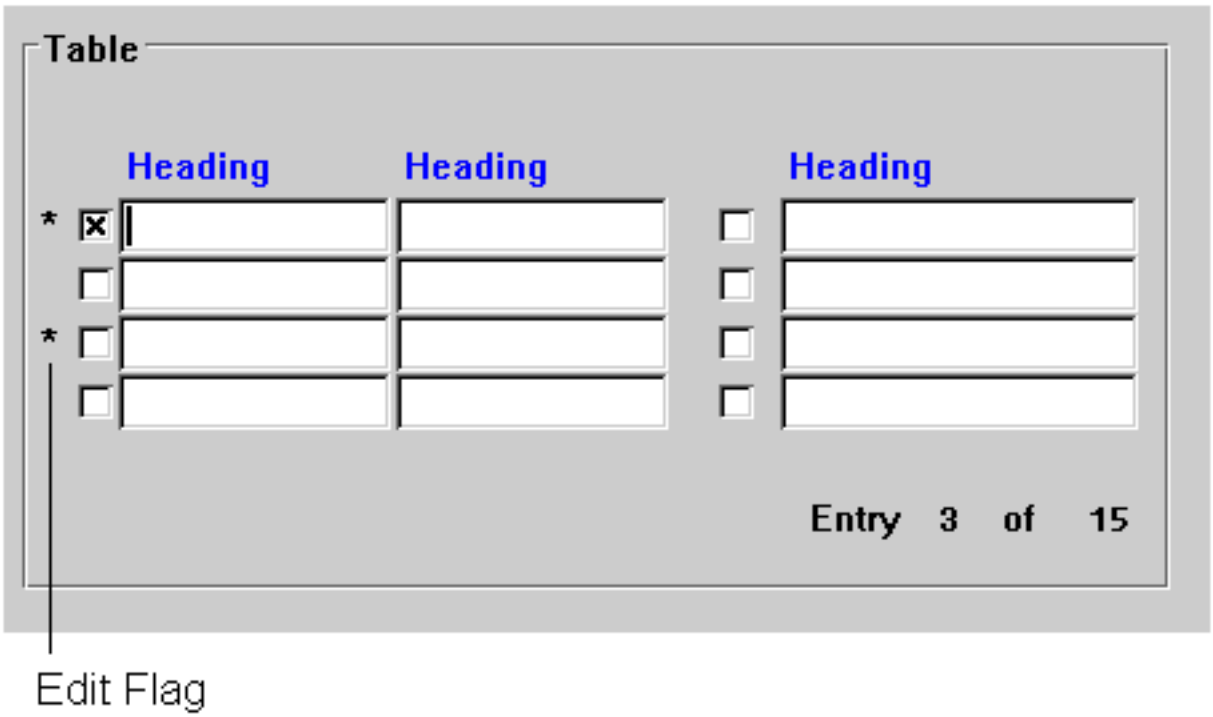


Figure 2: The edit flag is displayed as a column of asterisks

Single-Selection Tables or Lists

Use single-selection tables or lists to choose exactly one item (line) from any number of options.

- The table or list must contain at least two options.
- One option of the table or list has always to be chosen.

Single-selection tables or lists can be part of primary windows or appear in a separate dialogue box.

In addition, the guidelines specified for multiple-selection lists apply.

Selection

Single-selection tables or lists are not ready for input. Entire logical lines are selected.

For one-column or two-column tables, you can place a radio button before the tables to indicate single selection (explicit selection). For large multiple-column tables, radio buttons should currently not be used. Use automatic selection in this case or use the table control.

Matrices

If a list of objects exists and several independent options can be set to each object, list the objects one below the other. The user sets the options by activating adjacent checkboxes. In this case, the text of the option or the checkbox appears as a column heading above the checkboxes. If possible, the checkboxes should appear centered below the column headings.

This guideline also applies to radio buttons. The options of a line must be an affiliated group in the Screen Painter.

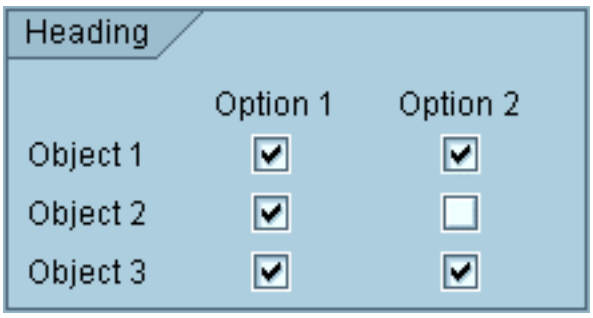


Figure 3: A matrix with checkboxes

SAP Matrix

"SAP matrix" in this context means a table ready for input containing fields, lines and columns as objects the user can select. The user can select as many lines, columns and fields as desired.

Selection

- **Fields**: Explicit selection and deselection. The selected field can then be processed (that is, edited) with any function.
- **Lines**: Same as with the multiple selection. The cursor, however, has to be positioned on the checkbox in front of the lines.
- **Columns**: An entire column is selected and deselected by explicitly selecting and deselecting the column header.

Checkboxes can select lines or fields, depending on the arrangement.

top

Source: SAP R/3 Style Guide

 Designing Tables

Table Control

The table control provides a look similar to spreadsheet programs, with column and row headers that may select entire columns or rows, a new selection logic, an inbuilt scroll mechanism, and much more. This control makes interaction with tables much easier for the users, but also for the developer.

In cases of one-line tables the table control should replace the step loop when presenting structures in table form. It offers improved usability and clarity compared with the loop due to the range of functions installed and the more precise visualization. You still have to implement multi-line tables in steploop technique.

Areas of Application

The table control should replace loops in the system. In general terms using the table control has the following advantages:

- Scrollbar directly on the table control, not at the edge of the window. This is particularly important where multiple table controls appear on a screen
- Easier to select lines (single and multiple)

When converting from table control to loop, functions specific to table control (for example, Sort, Line insert, Line delete) should be removed from the application toolbar and placed near the table control.
The way it is presented makes the table control **unsuitable** for editor simulation.

Rules of Application

Configuration

Users can change the width of columns and rearrange their order.

Configuration should be switched off in the following cases:

- If there is no meaningful way to configure the table control
- If the table control is dynamically modified by the program. Problems may arise if configuration is not switched off

When configuration is active, user settings take priority over all other settings.

Dynamic Modification

The program can dynamically modify the table control (for example, columns can be hidden). When a table control is dynamically modified configuration must be switched off to avoid unforeseen problems. **Either the program or the user can control the table control, but not both.**

Dynamic modification means that a LOOP AT SCREEN, for example, can influence the appearance of the table control. **Under no circumstances** should a pseudo-modification be executed by manipulating table TCVIEW in which settings configured by the user are stored. This could lead to unforeseen problems since the table is defined as a customer table and is supplied empty.

Where users are to be given a choice of views, these should be defined by the application and offered as pushbuttons.

Examples of the different design possibilities can be found in the example transaction BIBS.

Look of the Table Control

Depending on the setting of its attributes the table control can have a very different look. We show two typical applications of the table control, namely entry and display tables.

Base planning obj R-1110 CPU-66

Costing items - basic view

M	Item	T	Resource	Plant/...	Pur...	Quantity	U...	L	Value - total	Description
	1	E	4275	1420		0,033	H		2,56	Machine Hours
	2	E	4275	1420		0,033	H		2,56	Machine Hours
	3	M	R-1210	3200		1	PC		175,00	PROCESSOR M-375
	4	M	R-1220	3200		1	PC		100,00	MEMORY, 8 MB
	5	B	R-1240			1	PC		37,25	PCA
	6	E	4275	1423		4	H		107,11	Burn-in Hours
	7	E	4275	1420		0,017	H		1,32	Machine Hours
	8	E	4275	1421		0,017	H		0,86	Labor Hours
	9	M	R-1230	3200		1	PC		6,00	BIOS
	10	G							42,15	OHS - raw material
	11	S							474,81	

Figure 1: An entry table with single-selection for lines implemented with the table control

Base planning obj R-1110 CPU-66

Costing items - basic view

M	Item	T	Resource	Plant/...	Pur...	Quantity	U...	L	Value - total	Description
	1	E	4275	1420		0,033	H		2,56	Machine Hours
	2	E	4275	1420		0,033	H		2,56	Machine Hours
	3	M	R-1210	3200		1	PC		175,00	PROCESSOR M-375
	4	M	R-1220	3200		1	PC		100,00	MEMORY, 8 MB
	5	B	R-1240			1	PC		37,25	PCA
	6	E	4275	1423		4	H		107,11	Burn-in Hours
	7	E	4275	1420		0,017	H		1,32	Machine Hours
	8	E	4275	1421		0,017	H		0,86	Labor Hours
	9	M	R-1230	3200		1	PC		6,00	BIOS
	10	G				0,000			42,15	OHS - raw material
	11	S				0,000			474,81	

Figure 2: An display table implemented with the table control

 Table Control

Functions of the Table Control

Included Functions

Selection

- Insertion of selection column and display of selection state (highlighting)
- Selection functions:
  - Select/Deselect cell
  - Select/Deselect row/column
  - Select/Deselect All

Scrolling

- Scroll functions (horizontal and vertical) accessible via the table scrollbars and via scroll buttons
- Hiding of the vertical scrollbar at runtime, when the visible data area is smaller than the visible table area
- Resizing (vertical)
- Static and/or dynamic resizing of table columns

Operations on Data Columns

- Column width can be manipulated directly (the column width is stored independently of the field width)
- Fixation and display of an arbitrary number of key columns
- Moving columns via drag and drop
- Dynamic hiding of columns by the task

User Preferences

- Storing and retrieving of user preferences for the column layout
- Disabling (at design time) and thus deactivating (at runtime) the customize pushbutton
- Display
- Displaying column headings as pushbuttons (feature also settable at runtime)
- Optional display of horizontal and vertical separator lines in the table body
- Displayable Field Types
- Assigning attributes to any cell at runtime
- In the table body you can use the following controls: scrollable I/O fields, vertical and horizontal radio buttons, checkboxes

Not Available or Delayed Functions

- Using pushbuttons in the table body
- Muli-line display, especially for text fields
- Pushbuttons for multi-line headings
- Joint columns (i.e. jointly movable columns)
- Ruler (for rows)
- Selecting any number of cells (e.g., not contiguous selections)
- Local display of entry number and current row number
- Pushbuttons for selections that can be filled with text and icons by the task
- Horizontal resizing of the table control

Functions to be Implemented by the Developer

- Inserting blank lines between already filled rows (at the end of the data rows blank lines are inserted automatically)
- Deleting and processing of selected objects
- Calling detail information for selected objects
- Other task-specific functions

Pushbuttons for Self-Defined Functions

Arrange the above-mentioned functions as pushbuttons with no fixed position. Place them horizontally and left-aligned below the table control in the order:

- Detail
- Insert line(s)
- Delete line(s)
- other functions

Use the appropriate icons for these functions.

Alternatively, you may arrange these functions to the right of the table control, starting at the lower border of the table.

Do not position the general clipboard functions *Cut*, *Copy*, and *Paste* as pushbuttons at the table control. They are not limited to the table control, but effect all selected screen objects. Therefore, position them in the standard toolbar.

 top

Source: SAP R/3 Style Guide

Designing Tables

Placement of Table-Related Functions

The goals of the following proposed changes are:

- To specify that table-related functions should appear as pushbuttons near the table control, and not in the pull-down menus or in the application toolbar.
- To specify the order in which the table-related pushbuttons should appear.

Background

Due to the ever-increasing number of functions in most R/3 applications, the pull-down menus are becoming longer and longer, making it very difficult for users to find the functions they want. Application toolbars are also becoming more crowded, making particular pushbuttons difficult to find and making the various icons more difficult to differentiate.

As more complex contexts appear in the R/3 system it will become even more difficult to find the controls that pertain to a particular table or other screen element. In the specific case of table controls, placing functions in the pull-down menus and application toolbar forces users to change contexts, leaving the table area to go search the menus or toolbars for the table-function that they need.

Placement and Spacing

Functions that are related to a table should appear as pushbuttons in the window work area, very close to the table itself. It is not necessary for these table-related functions to be in the pull-down menus or in the system or application toolbars.

All table-function pushbuttons should appear in a single row, immediately below the table. In most cases, this placement matches the top-to-bottom workflow of the user, and positions the pushbuttons clearly in context of the table and away from other controls.

This placement and spacing of buttons applies whether the table appears in a main window, in a tab area, or in a pop-up dialogue. In a pop-up dialogue, the table-function pushbuttons must be in the same row as the dialogue control pushbuttons "OK" and "Cancel".

Exception: In some cases, it is allowable to place the button row above the table, instead. One example would be if you expect the enduser to be using a very large screen and you want the buttons to be closer to the top of the table to make them more visible and to minimize mouse movement to get to the pushbuttons.

Horizontal Spacing

The leftmost button should be left-aligned with the bottom left edge of the table. Pushbuttons for related functions must be placed right next to one another, with one horizontal space (one column in the Screen Painter) between them. Groups of functions must be separated by a two spaces (two columns in the Screen Painter).

See the suggested ordering in figure 2 for a recommendation of which buttons to group together.

Vertical Spacing

There should be a very small vertical space between the bottom of the table and the tops of the pushbuttons. This space is about

half the height of a push-button. The necessary space will be automatically generated by the GUI. Do **not** insert an empty row between the table and the buttons - this will cause too much vertical space to appear.

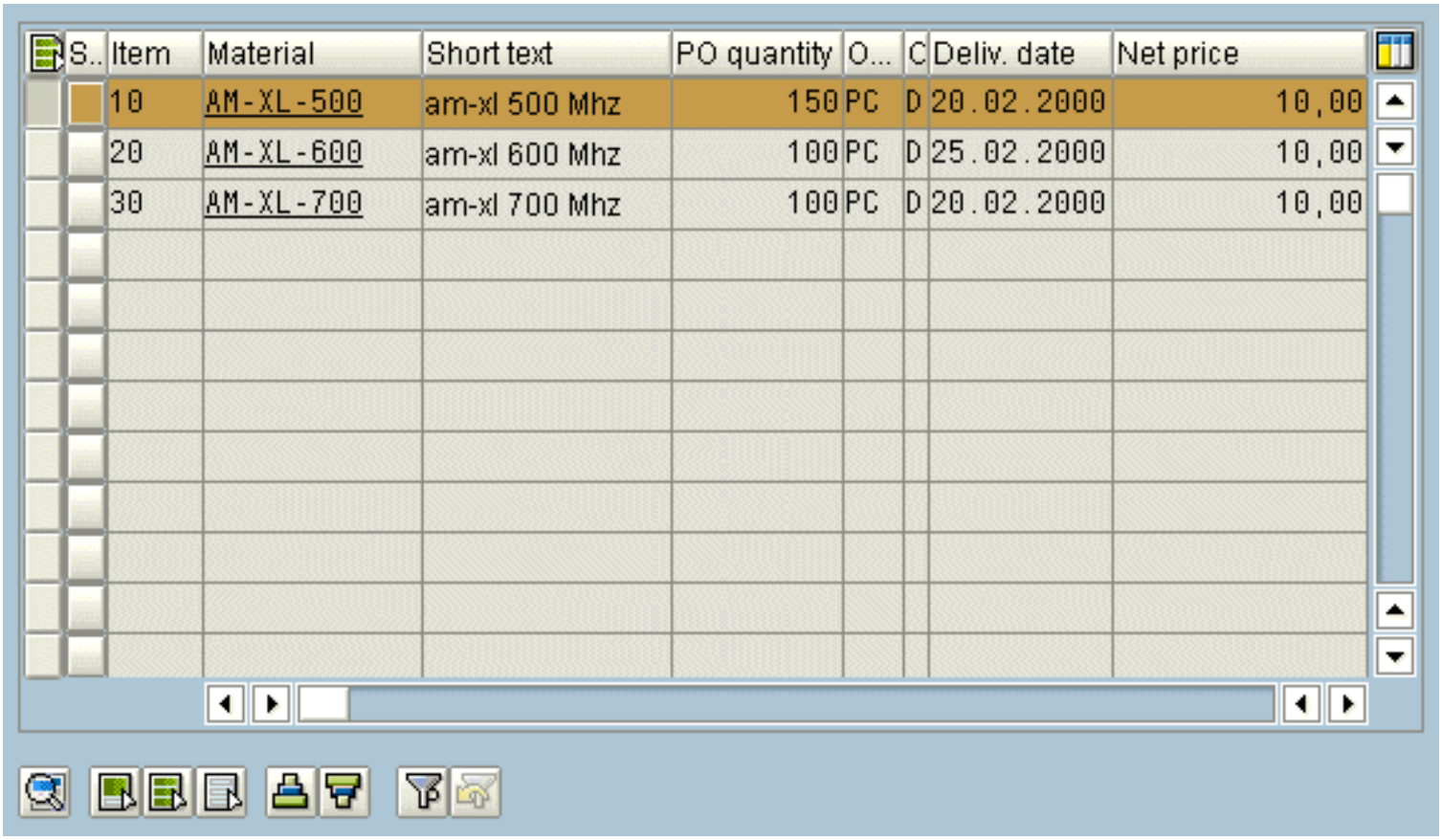


Figure 1: Placement of table pushbuttons

Order from Left to Right

Figure 2 shows a list of the most commonly used table-related functions, as well as a nearly complete list of all table functions. The functions that you choose to use depend on the context of the table and the needs of the users' tasks, but usually you need only the most common ones, plus any functions specifically related to your application.

Once you determine which functions are appropriate for your application, place the pushbuttons in the order shown below, from left to right. Include separator spaces between groups of buttons, as indicated.

Any functions that you do not use should not be included. These unused functions should not appear in the button row at all - do **not** include them and then gray them out.

This ordering of buttons applies whether the table appears in a main window, in a tab area, or in a pop-up dialogue. In a pop-up dialogue, the "OK" button should appear at the far left and the "Cancel" button should appear at the far right, after all of the-related pushbuttons.

Most Commonly Used Table Functions: Order from Left to Right

Detail
Select Block
Select All
Deselect All
Insert row
Delete Row
Sort Up*
Sort Down*
Position
Search
Application-specific function 1
Application-specific function 2

.
.
.

= separator spaces

* Use either *Sort* alone, or *Sort Up* and *Sort Down* together.

(Nearly) All Table Functions: Order From Left to Right

Detail
Select Block
Select All
Deselect All
Insert row
Delete Row
Move
Copy as
Replace
Undo Changes
Refresh
Sort*
Sort Up*
Sort Down*
Position
Search
Search Next
Search Previous
Filter
Highlighting
Print
Print selected
Export
Expand
Collapse
Sum
Intermediate sums

Figure 2: Order of most commonly used functions

top

Source: **SAP R/3 Style Guide**

Tree Structures

Selecting the Right Display for the Right Purpose | Design and Layout | Functions and Control Elements | Reducing Hierarchy Levels

Tree structures (short: trees) and lists are elements that can be used to model a large amount of data in easily readable form. They are particularly well-suited to representing hierarchical dependencies and give the user a clear overview of a large quantity of information.

Do Use

Lists and tree structures can therefore be extremely effective when they are used for the following areas, for example:

- Trees allow an excellent **overview** of different objects that are related to one another. This means many objects are displayed, but each individual object only has a small amount of information.
- Trees allow the modeling of graphical **relationships** between objects. Illustrations of this type can help promote orientation and comprehension of the relationships for users charged with performing a task.
- Trees represent a central access point for **navigation** between different objects. The user can avoid frequent screen changes and having to remember where specific information is located.
- Trees allow **fast access** with direct manipulation (via drag and drop) of objects within the structure. As a result, trees are particularly well-suited to activities that require users to re-sort or move objects.

Do not Use

To enable the above features, the trees and activity structure must "fit" one another. In some situations, the use of trees can cause more problems than it solves. For example, you should **not** use trees to:

- Model existing hierarchical menu and navigation structures in the system without prior revision. Simply copying nested area menus into a tree structure does not help the user.
- Display complex information in trees, as well as interface elements like pushbuttons or checkboxes. This is not recommended due to the complexity of the resulting tree and the limited space available.

Problems

In addition, the use of trees results in other typical problems:

- **Space requirements**: The tree occupies space on the screen, restricting the size of the work area.
- **Visual complexity**: Because the overall screen is divided into different areas, it appears to be more complex.
- **Complex interaction** between index area and work area: Changes in views and highlighting for object selections have to be carefully coordinated between the two areas.
- **Too much information**: Displaying too much information for individual objects in a tree can quickly make the entire structure confusing.
- **Misuse of icons and colors**: It is easy to use unsuitable, too many, or confusing icons. The same thing applies to colors.
- **Assigning functions:** It is often unclear which pushbuttons affect the tree structure and which ones affect the data.
- **Tree functions in menus:** Functions that only apply to a specific tree structure can be "hidden" in various places in the menus.

 Tree Structures

Selecting The Right Display for the Right Purpose

Data can be displayed using many different methods other than the classic "tree structure". Options range from simple list-like displays to complex presentation in ALV grids or the use of so-called shufflers. By answering the following questions, you can help decide which method is the best for displaying the required data:

- What does the presentation of the data have to achieve?
- What is supposed to happen to the data itself?
- Which technique is best suited to this purpose?

To answer these questions and decide on a possible design, we recommend using a decision table that contains the following criteria:

Usage

The user's activities and objectives in a certain concept determine the basic interaction and display forms.

Examples: Display or selection of objects, entering data for complex interactions such as reassignment, re-sorting items

(Data) Structure

In many cases, the data is already present in a logical structure. This structure and the resulting dependencies affect the type of display and how the data is used.

Examples: Lists, tables, outlines, hierarchical tables, hierarchical/sequential lists, tree structures

Hierarchy Depth and Delimitation

Categories or levels are created depending on the type and division of the data. These categories can be indigenous to the data itself or applied externally to clearly sort a large number of elements. The more levels are introduced, the more difficult the manipulation of the overall structure becomes. As a result, your design objective should be to get by with as few levels as possible.

Examples: No hierarchy, one level, two to three levels, many levels, fixed or variable number of levels.

Number of Columns

The user's information requirements determine the amount of data that is displayed. Columns are generally used to display additional information for a specific object. The number of required columns determines the type of suitable presentation and interaction.

Examples: One, two to three, three - n

Overview

The above criteria cannot be considered independently of one another. The following table supplies a rough overview of the possible combinations. However, you should also consider the exact description and analysis of the scenario, as well as the technical and other framework conditions for using the various display forms, for your individual case at hand.

Structure	Hierarchy Levels	Number of Columns	Usage (Scenario)		Recommended Display
List	1	1	Browsing through information		List Table ALV grid
Table	1	2-n			ALV grid Table control
Outline	2	1			
Hierarchical/sequential list	2	2-n			ALV grid
Regular hier. table	2-10	2-n			ALV grid
Irregular hier. table	2-3	2-n			ALV grid
Tree	3 or more	1	**->** Navigation and search		
			-> Re-sorting and copying		
			-> Browsing through information		
		2 or more	**->** Navigation and search		
			-> Re-sorting and copying		Simple tree with one column
			->Browsing through information		ALV tree as full-screen display

Table 1: Basic interaction and diplay forms

Always remember the general rule: Hierarchical structures are difficult to comprehend and use. Therefore, they should be as simple as possible. You should use every possible option for reducing hierarchy levels.

top

Source: SAP R/3 Style Guide

Tree Structures

Design and Layout

Trees are well-suited for navigating through hierarchically structured data and for manipulating these structures. They are less well suited - especially considering the limited space available (tree as docking control to the left of the main window) - for displaying large amounts of detailed information.

In this context, when you use a tree, you should make sure that it only contains the information that the user really needs to uniquely identify the contents.

Tree structures are used whenever you want to display more than two levels of a hierarchy. Due to the increasing complexity of the display, however, you should **never model more than five** levels in a tree.

Note: If you determine that you will need more than five levels, you should consider whether the amount of information at hand is too extensive for the screen. In this case, you should consider other options for displaying the additional information, such as navigation to a further main screen, calling a dialogue box, or structuring the additional information in a tabstrip.

Tree Variants

The tree control has three different variants: The Simple tree as the simplest form of the hierarchy, the List tree and the Column tree , which can be used primarily for trees with heterogeneous line structures.

Nodes, Scrolling, and Objects

The original concept of a tree consisted of a hierarchy of folders (nodes) and pages (documents, or "printed pages"). In many cases, this metaphor is not suitable for the objects that are involved in an application. Therefore, whenever you use a tree, you have to consider exactly which functions and views you intend to model with the different elements of the tree.

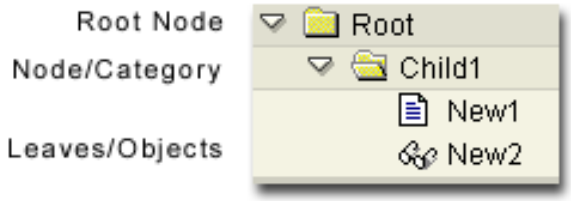


Figure 1: Elements of a Simple Tree

You may encounter problems when allocating the different "object types" to the corresponding elements of the tree. The following table lists an overview of possible combinations and the resulting problems.

Object Type	Display
Objects (in the real world; applications)	Node / page
Functions (of an application)	Should not be contained in a tree
Attributes (of an object)	Columns in a column tree
Views (logical views; different lists / detailed views)	Sheets
Tasks (of a workflow)	Sheets (nodes if lower-level hierarchy structures exist)
Concepts	---

Table 1: Allocation of Objects to Tree Elements

- When *objects* are displayed as *nodes* (category), it is not always clear what to display in the work area when a category is selected. If an object/sub-object relation exists (object as node, sub-object as page), the object data is displayed in the work area when the node is selected, and the sub-object data is displayed in the work area when a page is selected.
- Functions (such as *Create* or *Change)* should generally not be displayed as elements of a tree. These functions are better placed in the toolbar of the tree itself (ALV tree) or in the application toolbar. The functions can also be contained in the context menu of the tree.
- Whenever possible, you should not use a root node if it merely represents the "header" of a tree. Instead, you should use the header area for displaying headers (in simple trees). By eliminating the root node, you save valuable space in the left screen area and reduce the visual complexity.
- Branches that only contain one additional node or one additional page are pointless, and should instead be summarized at the next-higher hierarchy level.

Header Area

The header area of a tree structure is generally used to identify the type of information in a list. In a simple tree, for example, this could be the name of the root node. In column trees, the header is used to describe the individual columns and change the respective column widths.

This results in the following recommendations for headers:

- In simple trees, you should use the header to describe the root node. As a result, you no longer have to model this node in the tree structure itself, and you can save a hierarchy level, making the tree clearer and easier to read.
- In a column tree, you should make it possible to display and hide the individual columns in the header. This enables the user to determine which degree of information is displayed in the tree. This is particularly useful for toggling between the objectives "simple, fast navigation" and "targeted browsing of detailed information".

Colors, Fonts, and Icons

Because the design options within the list technology were too restricted for structuring the information, list colors were created as an additional structuring option. This aid was also used for structuring within the list tree, but is no longer necessary thanks to a specialized user interface element.

As the table below shows, using colors in the SAPTree OCX is only sensible in a few places from an ergonomic and design standpoint.

Attributes	List tree	SAPTree OCX
Hierarchy levels	Color	No color (except for the default background colors), since the hierarchy is already sufficiently visualized through indentation, icons, and/or lines. A tree control is an interface element that is optimized for displaying hierarchies.
Status	Color, symbols, and icons	Individual status with foreground color critical status and background color icon for object attribute
Object classes/type, attribute	Color, symbols, and icons	No color, but icons
Line/columns structure	Color	No other color is necessary, since lines, columns, and location make additional differentiation through color superfluous

Table 2: Using Colors and Icons in a Tree

Notes

- The "folder" icon  should be used with great caution, since it symbolizes a container function! However, R/3 only has a few objects that perform a "pure" container function. This icon can be used sensibly to represent a work list or user-defined views, for example. In this case, the "folder" contains the elements and the folder characteristics can be filters or selection criteria (example: see below)
- If the lowest level of a tree represents different types of objects for which the system does not have any defined icons, you can use the so-called "unspecified" icons to differentiate between these types. These are the icons "Icon_unspecified_1" through "Icon_unspecified_5". These icons differ in both color and form.

Characteristics

Differentiating the status refers to the frequency with which a status appears in a tree. To indicate critical values, for example, it makes sense to choose an obvious background color, whereas obvious icons should be used to indicate object attributes that occur in nearly every line of the tree. The use of the foreground color to indicate the status is only sensible when you want to visualize a two-value element, since the selection of legible foreground colors is limited to black, blue, and red.

As a result, the following options for color-coding nodes are possible in the SAPTree OCX:

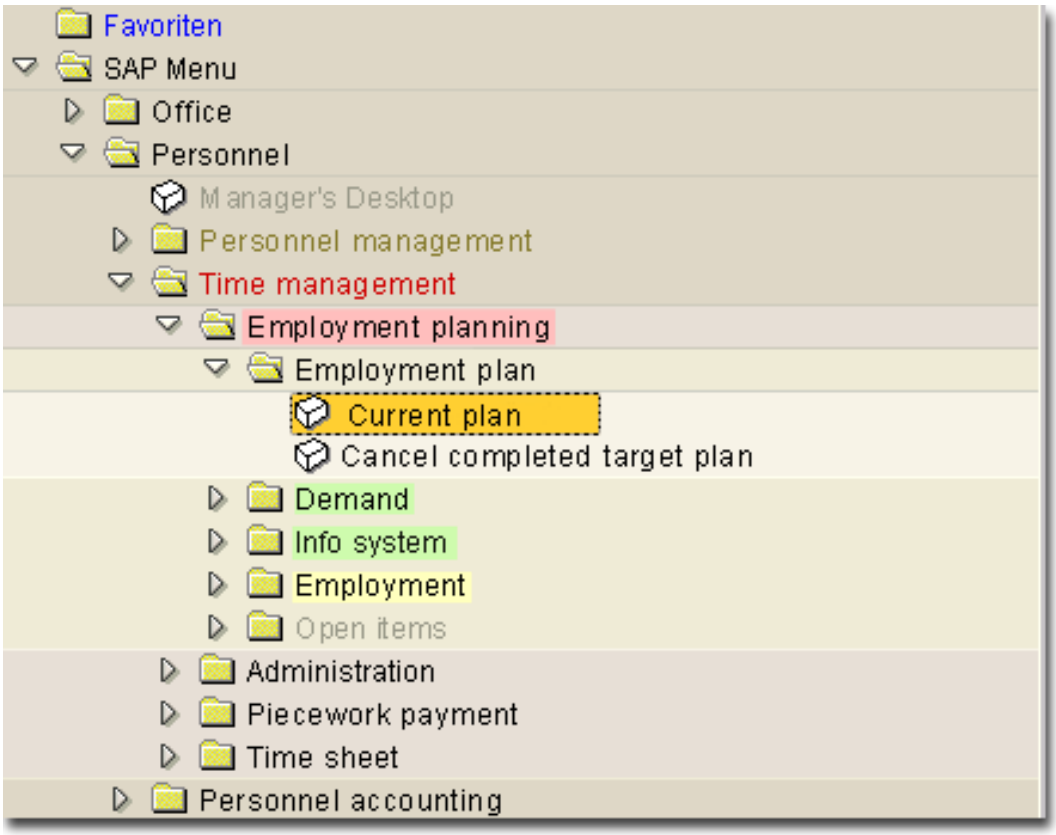


Figure 2: Color-Coding in Trees

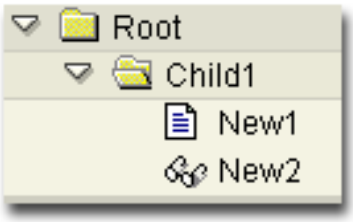
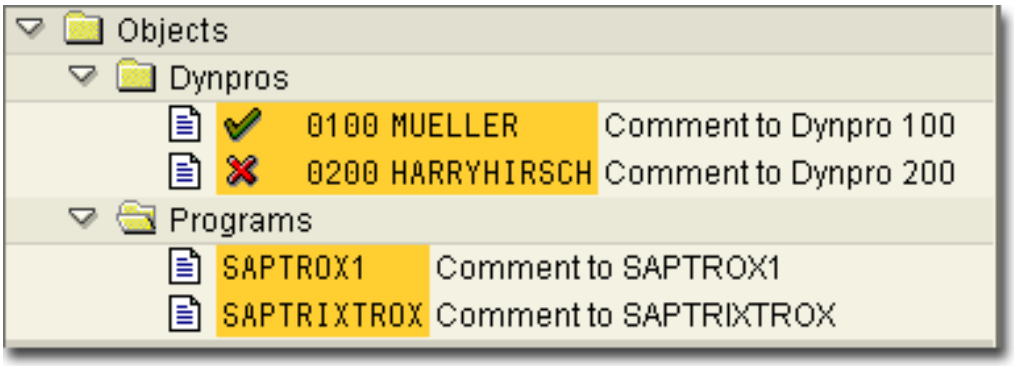
Color are assigned in the SAPTree OCX using **styles**, which are defined through style constants. Styles cannot be combined with one another.

Meaning	Color	Style Constant
Default	Black FG	TREEV_STYLE_DEFAULT
Positive importance	Blue FG	TREEV_STYLE_INTENSIFIED
Inactive status	Gray FG	TREEV_STYLE_INACTIVE
Negative importance	Red FG	TREEV_STYLE_INTENSIFD_CRITICAL
Negative threshold	Red BG	TREEV_STYLE_EMPHASIZED_NEGATIV
Positive threshold	Green BG	TREEV_STYLE_EMPHASIZED_POSITIV
General highlighting	Yellow BG	TREEV_STYLE_EMPHASIZED
Selection	Windows Standard	---

Table 3: Meaning of the Color Codes and Style Constants

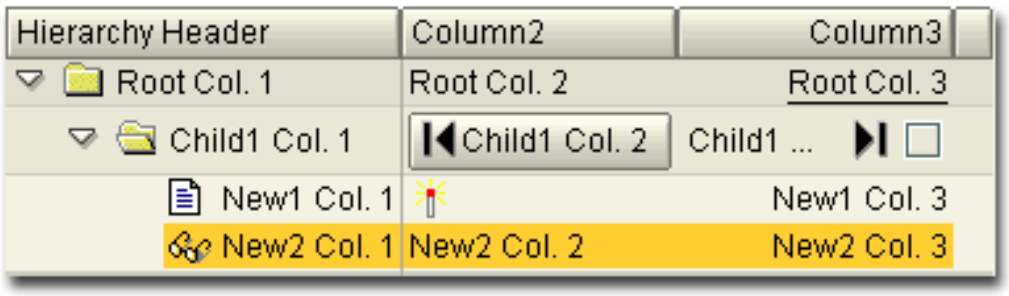

Design and Layout

Overview of Tree Variants

Name	Display	Description
Simple tree		The simple tree represents the "traditional" variant, in which only a small amount of information, such as the name, is displayed for an object. As a result, this type of tree is extremely well-suited for navigating between objects in an "attached" window area.
List tree		The list tree display largely corresponds to the existing SEUT trees. In addition to naming the additional tree elements, the structures contain a variety of other useful information, such as status and additional texts. Due to the amount of information displayed, such trees require significantly more space on the screen (or requires horizontal scrolling) and are therefore suitable for applications that use the tree as the primary screen area.

Column tree

a) With header area

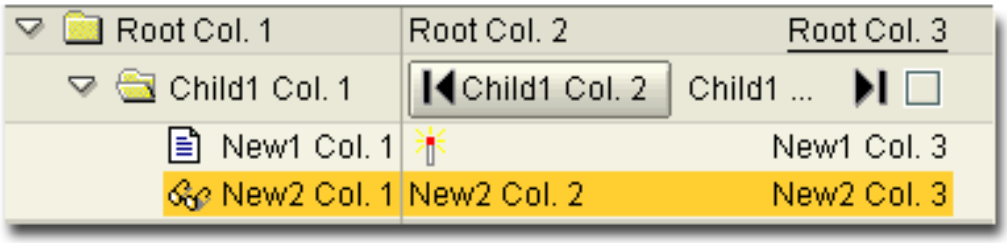


The column tree is a mixture of a tree (hierarchical display) and a table control (variable columns with headers). In addition, this type of tree can use pushbuttons and checkboxes in the columns, in addition to text and icons.

The problem with this type of tree is that longer entries are cut off at the point where the next column begins, and only the first (left-most) tree area supports horizontal scrolling. If the text alignment (right/left-aligned) differs as well, the display can quickly become confusing.

Column tree

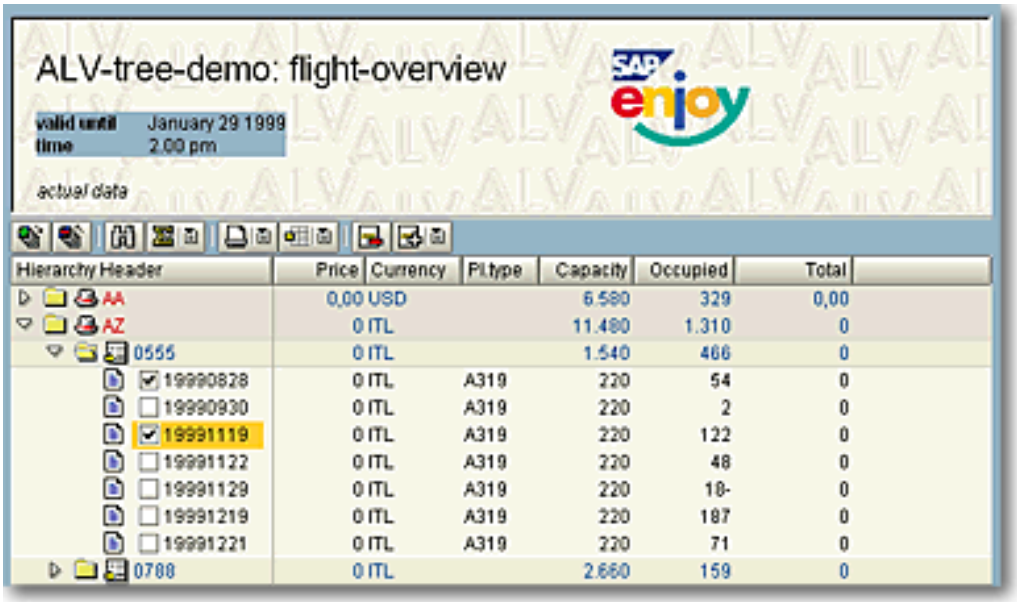
b) Without header area



The column tree can also be used without the header area. As a result, however, the individual columns do not have a fixed width. Each entry in the tree fills the columns variably. Especially when combined with indentation in the tree structure, this design can be very confusing and difficult to read.

Column tree

c) ALV tree



The ALV tree is a variant of the column tree that also contains its own toolbar and a title area.

Table 1: Overview and Function of Tree Variants

 Design and Layout

 top

Source: SAP R/3 Style Guide

 Tree Structures

Functions and Control Elements

Pushbuttons

The function for displaying and hiding the entire tree area appear in the far left of the application toolbar. Pushbuttons that contain functions for a tree (such as expand all) usually appear on the left in the application toolbar. All the other functions appear adjacently to the right. An exception to this rule is the ALV tree, since it has its own toolbar for the functions within the tree.

The pushbutton for displaying/hiding the entire tree remains on the left in the application toolbar, since this function would otherwise disappear when the area is hidden. This button should not contain any icons, but instead the labels "<Show area>" and "<Hide area>" (short: "<Area> on" and "<Area> off"). The text must then be changed dynamically so that it always describes the current function of the pushbutton.

If the tree cannot be re-sized with a docking control or splitter control, you can add additional pushbuttons to successively increase or decrease the size of the area. In this case, the area containing the tree is increased or decreased in size by a fixed amount each time one of the buttons is pressed. These buttons always appear in the application toolbar, right next to the functions for displaying and hiding the area.

Navigation with the Mouse

The mouse is the primary navigation tool within a tree. In general, all navigation (expanding/collapsing trees) and object selection should be performed with the mouse. The following table supplies an overview of the recommended mouse actions.

Tree object	Mouse click	Action
+ or 	Single left	Expand node
- or 	Single left	Collapse node
Node	Single left	Highlight node and select object
Node	Single right	Highlight and context menu
Node	Double left	Default action, such as expand node; possibly combined with actions for displaying the node contents
Node	Shift + Single left	Select several adjacent objects
Node	Ctrl + Single left	Select several non-adjacent objects
Page	Single left	Highlight object (and possibly display it)
Page	Double left	Select object and perform the default action (such as display)
Page	Single right	Highlight and context menu

Table 1: Mouse Navigation in Trees

 top

Source: SAP R/3 Style Guide


Tree Structures

Reducing Hierarchy Levels

Various options are available for "slimming down" hierarchies, and they differ in their usage and simplicity of implementation. The common starting point of all these techniques is moving the upper-most hierarchy levels to a type of "filter" or "view" mechanism. These variants are called tabstrip, filter and shuffler.

TabStrip

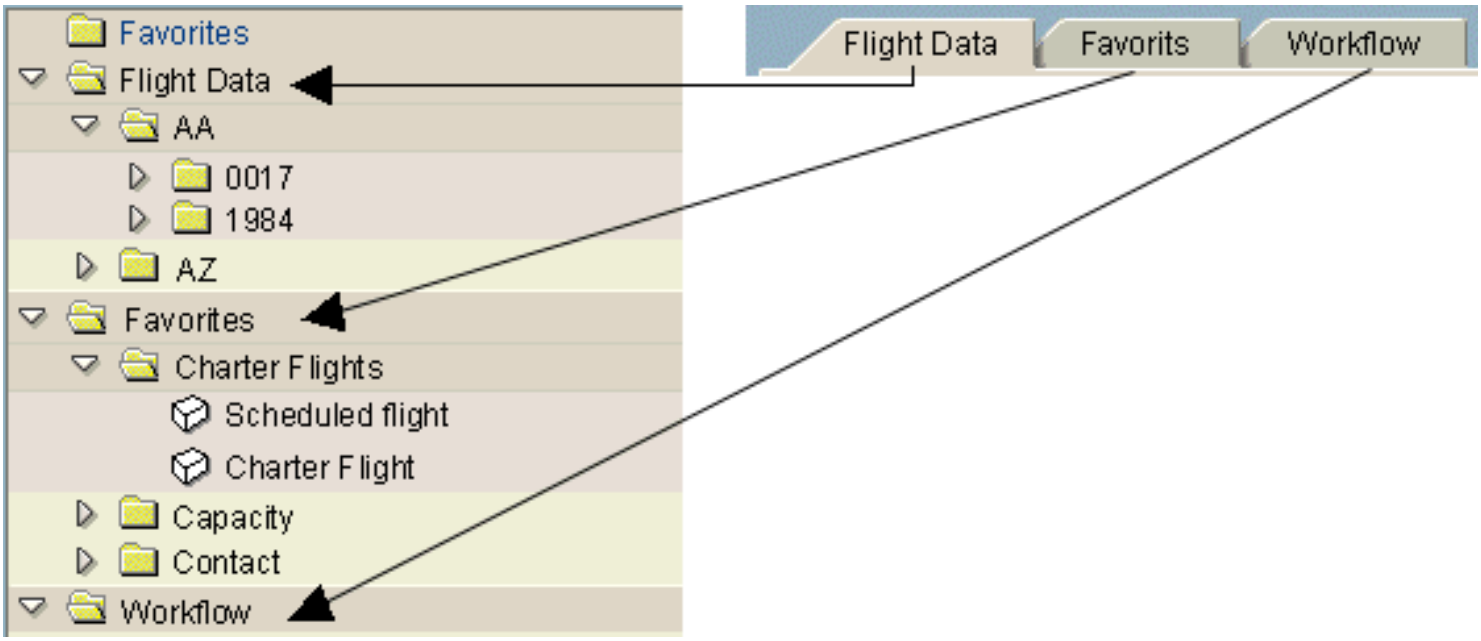


Figure 1: Converting nodes to tabstrips

A tabstrip allows the user to switch between different *views* of a dataset. Due to the relatively large space requirements, only a small number of nodes (3-5) can be modeled.

Summary: Tab strips

- Display different **views**
- Display a **small number** of categories

Filter

A filter reduces the amount of displayed data by setting a criterion. Different design options are available for entering the criterion. The example in the following figure shows the conversion of hierarchy level "Flight number" to a drop-down list box.

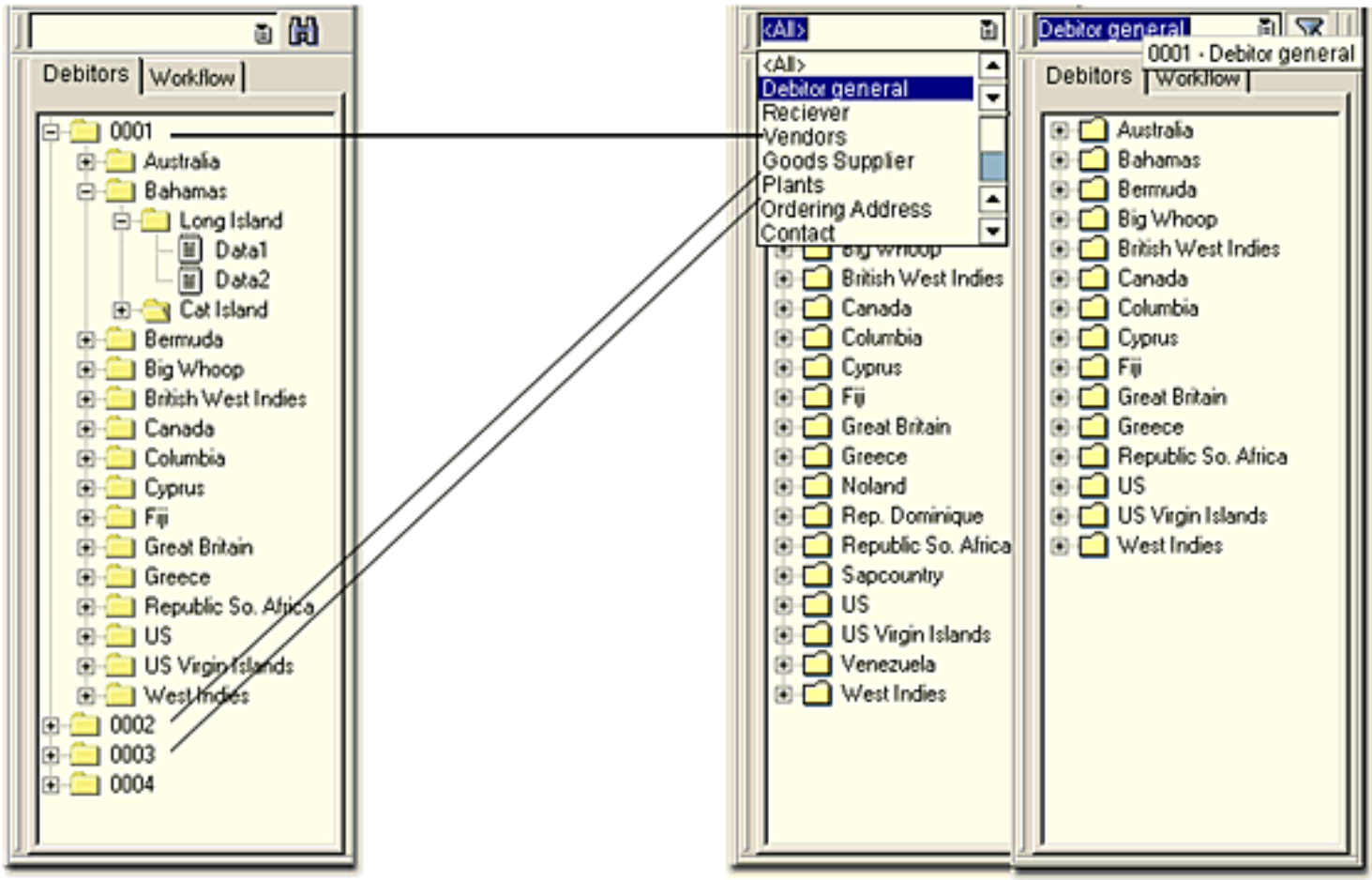


Figure 2: Filter for reducing the number of hierarchy levels

The advantages of the filter solution are its relatively low space requirements and its ability to model a relatively large number of nodes. However, not all the "views" are visible at the same time, and the user has to perform an additional interaction to change the view.

Summary: Filter

- **Suitable for modeling nodes**
- Reduces a **medium-sized number** of categories
- Data is **filtered** and displayed as **views**

Shuffler

The shuffler concept extends the filtering of data described above with two additional aspects. First of all, a shuffler usually consists of a collection of queries, for example, a combination of order numbers created during a specific time interval. Secondly, the variable components of the query are embedded in a semantic relationship - that is, a single sentence. For example, this sentence could say "Show me all the *order numbers* that were created *in the last year*". Theoretically, each variable component enables you to roll out one hierarchy level in a structure.

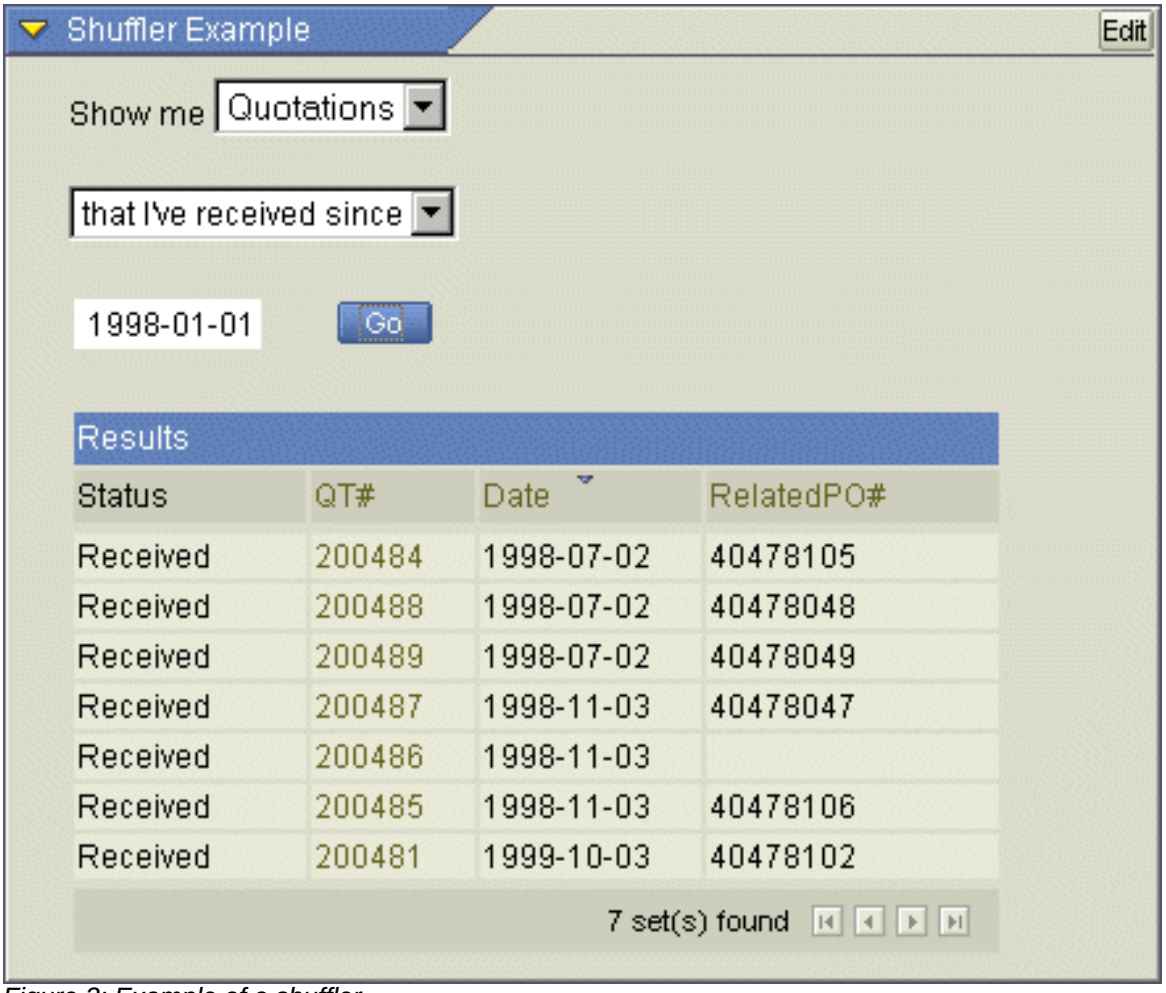


Figure 3: Example of a shuffler

The figure shows an example of how a shuffler can be used to reduce a three-level hierarchy (object type, processing type, and processing interval) to a simple list.

Summary: Shuffler

- Suitable for modeling **complex dependencies**
- Formulates the filter criteria in "natural" language
- May be complex to program

top

Source: SAP R/3 Style Guide

Tabstrips

Use and Design of Tabstrips | Examples of Use

Tabstrip controls (short: tabstrip) allow you to easily and comprehensively define different object components of an application on one screen and to navigate between them. In contrast to the classical screen change, the user at one glance sees all destinations he or she must call to accomplish the given task. Using this technique, the user can comprehend the structure of an application more intuitively as with conventional techniques such as the *Goto menu*. This reduces the learning expense and facilitates the usage.

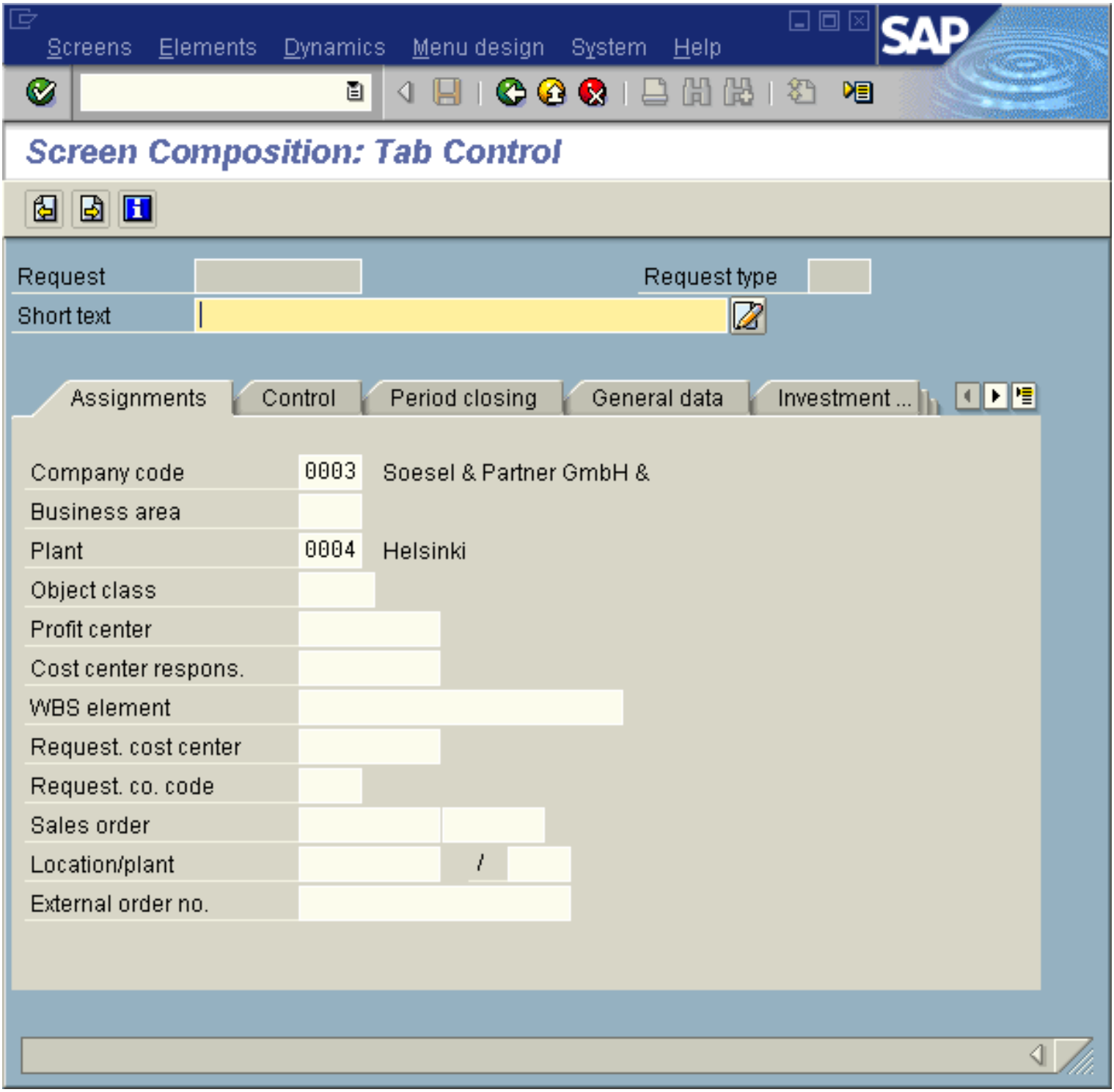


Figure 1: Tabstrips

Assessment of the Use of Tabstrips in an Application

Tabstrips should be used where they help users in their day-to-day work, especially:

- By breaking up screens that are overloaded with fields into units for different tasks
- By improving and simplifying navigation by reducing the need to move between screens (all screens available at once)

However, it is not always helpful to break up screens or group boxes, and sometimes moving objects in screens one-to-one to tabstrips is not helpful if it is not task-oriented. You should always ensure before implementing any tabstrip that the individual page elements do support the task flow and that the tabstrip does improve **clarity** and **simplify navigation**.

Don't Use Tabstrips if

- Each component can be regarded as its own application, or if the tabstrip environment (menus, pushbuttons, header data) cannot be kept constant.
- The components have to be processed in a particular order. Tabstrips should allow users to switch freely between components.
- The components are processed dynamically, that is, if an entry on one tab page causes other, previously invisible tab pages, to become visible.

top

Source: **SAP R/3 Style Guide**

Tabstrips

Use and Design of Tabstrips

Tab Titles

The labels on tab titles must be as short and meaningful as possible. You should arrange your tab titles from left to right in the sequence of the work process flow.

Since the advantage of the tabstrip control is that it displays all possible navigation options, you should try not to define more tab titles than it is possible to display on the screen. Certain sections of tasks in the SAP R/3 system are extremely complex and therefore it is not always possible to follow this guideline in practice.

If you define more tab titles than can be displayed, some will be concealed. Navigation buttons appear automatically at the end of the row of tab indexes to help users find the concealed titles. Users can also request a popup menu containing a list of all the tab titles.

If tab titles correspond to entries in Goto or Extras menus, remove them. However, the screen containing the tabstrip control itself must be accessible using the Goto menu.

While a user is working with the tabstrip control, the number of pages and their contents must remain constant.

Visibility

Tab indexes (title elements) must always be clearly visible and legible. This means there should only be as many tab indexes as there is space for them on the standard size sceen. If more tab pages are needed than can be shown on the screen, choose a different design solution, for example a list of the pages, a sequential processing "Roadmap", or even a wizard.

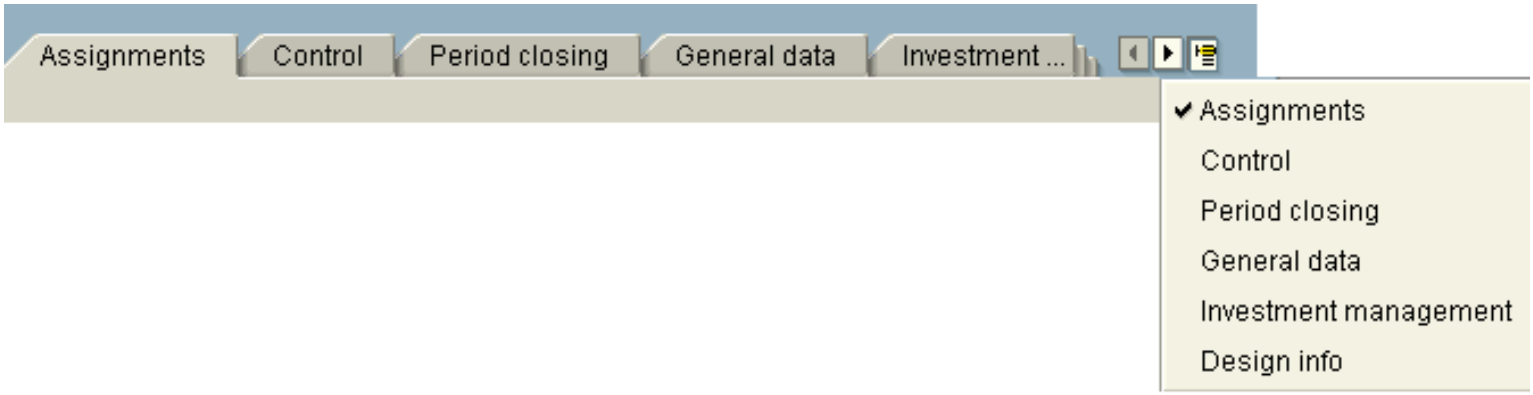


Figure 1: Avoid scrolling of tab titles!

Tab Page

The first line of the tab page is locked by the Screen Painter, since it is required by the tab titles. Normal screen design rules apply.

Including the tab title line, frame below, and locked first line, a tabstrip control requires three additional dynpro lines.

Tabstrip control and the tab page titles (labels on the tab titles) are the equivalent of a group frame, which means that frames to surround all fields in a tabstrip control page are unnecessary.

It is possible to arrange several group frames on a tabstrip control page. Take care to ensure that the tab page titles (labels on the tab titles) provide an appropriate description of the field groups on the tabstrip control page.

Screen Design

Screens should never be scrollable.
If there are more fields than there is room for on the screen, check whether tabstrips or other design alternatives would be appropriate. Always decide which alternative to use in the light of the task and the context:

- Tabstrip
- Closer packing of fields on a screen
- Dialogue box
- Separate screens
- Mechanisms for zooming in and out of particular screen areas
- Long screen with vertical scrollbar

Tab Environment

The environment of the tabstrip control must remain constant. This means that the menu bar and application toolbar must not change when the user switches between tab pages. Consequently, specific functions that are only active on one tab page must be implemented as pushbuttons on the page itself. These functions do not appear in the menu.

Leave one line between the header data and tabstrip control, as normal.

Interaction between Page Elements

Avoid complex interaction between tab pages. Users should not have to worry about why, say, a field that is not ready for input on page element no. 4 should become active when a selection is made on page element no. 2.

Navigation

Screens that contain tabstrip controls are regarded by users as a single screen. Therefore, the individual tab pages are part of that one screen, as though they were group boxes or field groups. The "F3", "F12", and "F15" functions act not on the individual pages, but on the whole of the 'container' screen.

If the user navigates from a tab page to another screen, you should ensure that the tab page from which they navigated is the active page when they return to the screen containing the tabstrip control.

Arrangement of Information

Do not distribute information the user wants to see together across different tabs. If in doubt, do not use a tabstrip. If, in a typical user scenario, the user would have to visit several tabs to work on a small number of fields, choose a different design that reduces navigation and increases clarity.

Required Entry Fields

The first page on a tabstrip should contain the most important and frequently used information. Any required entry fields should be on the top page element. If, for any reason, required entry fields are needed on other pages elements than the first, you must use the "backend variant" (field check on changing tab page).
Fields that are not technically required entry fields but which nevertheless must be maintained for completeness (for example, the short texts for data elements) should not be "hidden" on other page elementss than the first. If necessary, pages that the user must visit should be marked with a special icon on the tab.

Field Check

When you use required entry fields on a page element other than the top (first) page element there may be problems with field checking. Depending whether you use the frontend or backend method, you may create a situation where fields are not checked or where users see various error messages.

Branching away from the Tabstrip Area

Pushbuttons that trigger a jump to another screen should not be used in the tab area. Tabstrips should not include options to branch away during the task process flow because of the risk of disorientation.
Exceptionally, objects in a tabstrip may be choosable by double-click as long users are subsequently returned to the same page element they came from.

Function Triggered

When users choose a tab index (title element), it must be a tab page (page element) that appears, not a dialogue box or similar.

Element Arrangement

The arrangement and design of elements on tab pages (page elements) is subject to the same standards as on screens. If there is more than enough space available, do not space the elements overgenerously. Arrange the elements as compactly as usual, even if this means leaving a part of the tab page empty.
The vertical and horizontal points of origin of control elements on page elements should be consistent, to avoid any flicker effect when moving from page to page.

Non-Hierarchical Design

Do not use nested tabstrips, because they increase visual complexity and cause navigation problems. Instead, use other design solutions (for example, a dropdown listbox for choosing tabstrip units) .

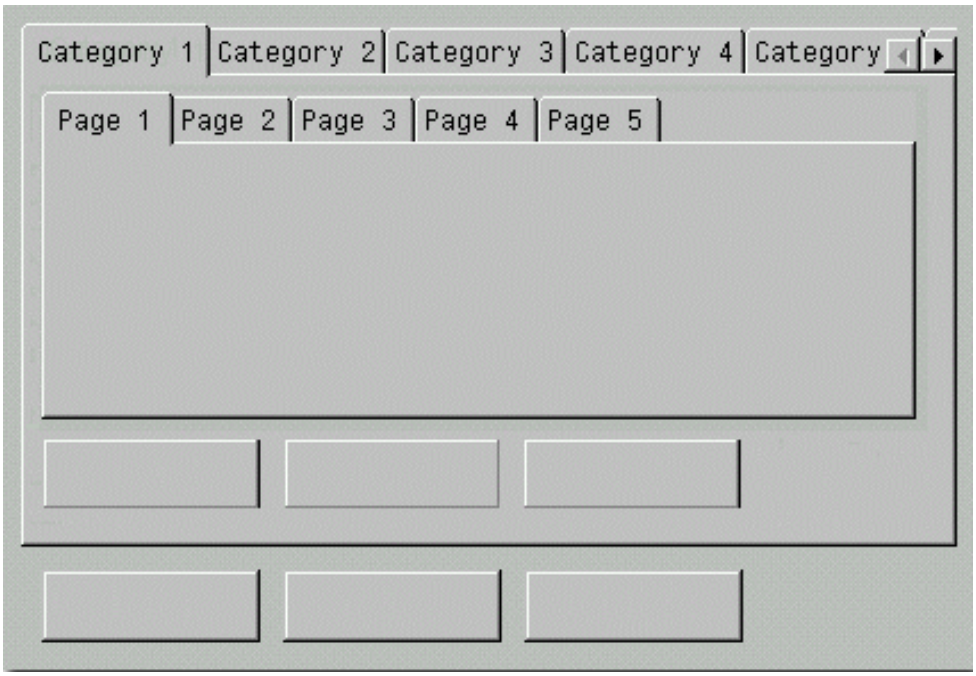


Figure 2: Don' t use tabstrips in tabstrips !

Arrangement of Pushbuttons

Buttons that are valid for all tab pages (page elements) or for a whole screen should be placed outside the borders of the tabstrip or in the application toolbar. Buttons valid for a particular tab page should be on that page.

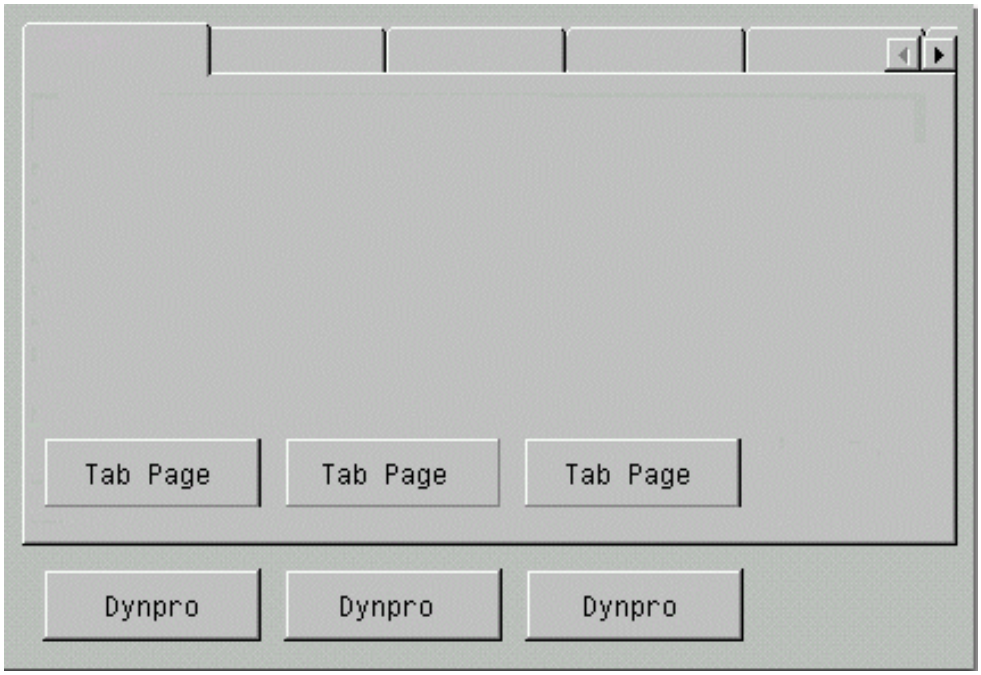


Figure 3: Distinguish between function belonging to the tab page and functions belonging to the dynpro !

Access to Tab Pages

Direct access to tab pages (page elements) by pushbutton or menu option is not appropriate from the page the tab is on.

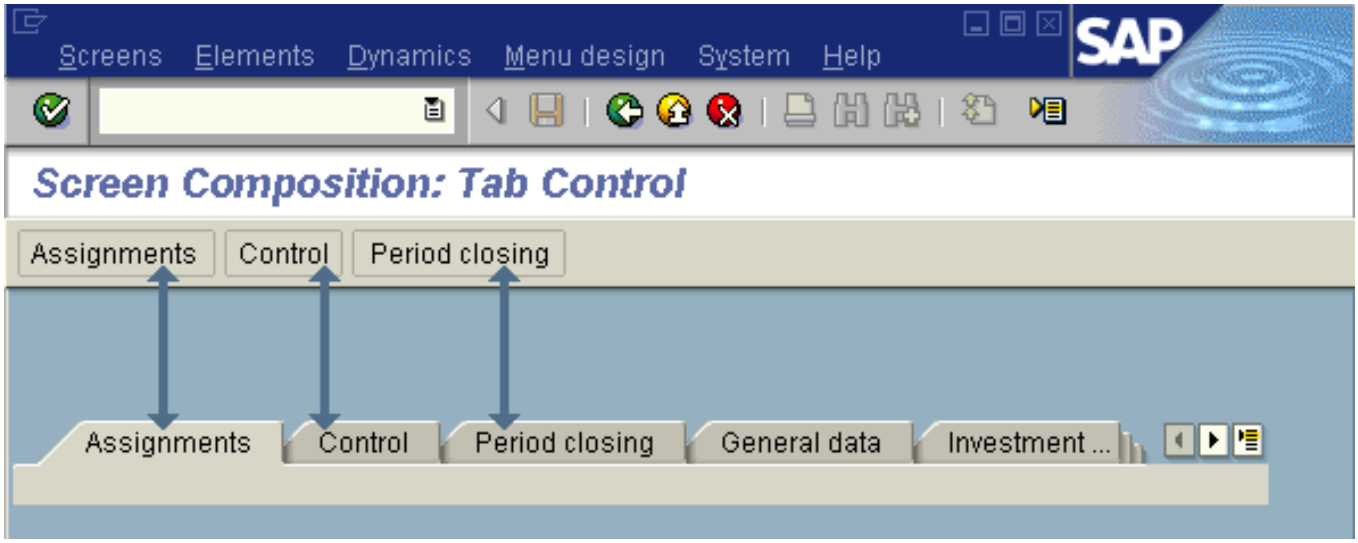


Figure 4: Incorrect pushbutton navigation from the page where there is also a tabstrip access

It may be appropriate to provide fast navigation.

Using Icons

In general, you should not use icons in tab indexes (title elements). If you must, the icons you use must help the user to recognize the individual tab indexes quickly: Using the same icon in more than one tab index (title element) tends to confuse users and does not provide any additional useful information.

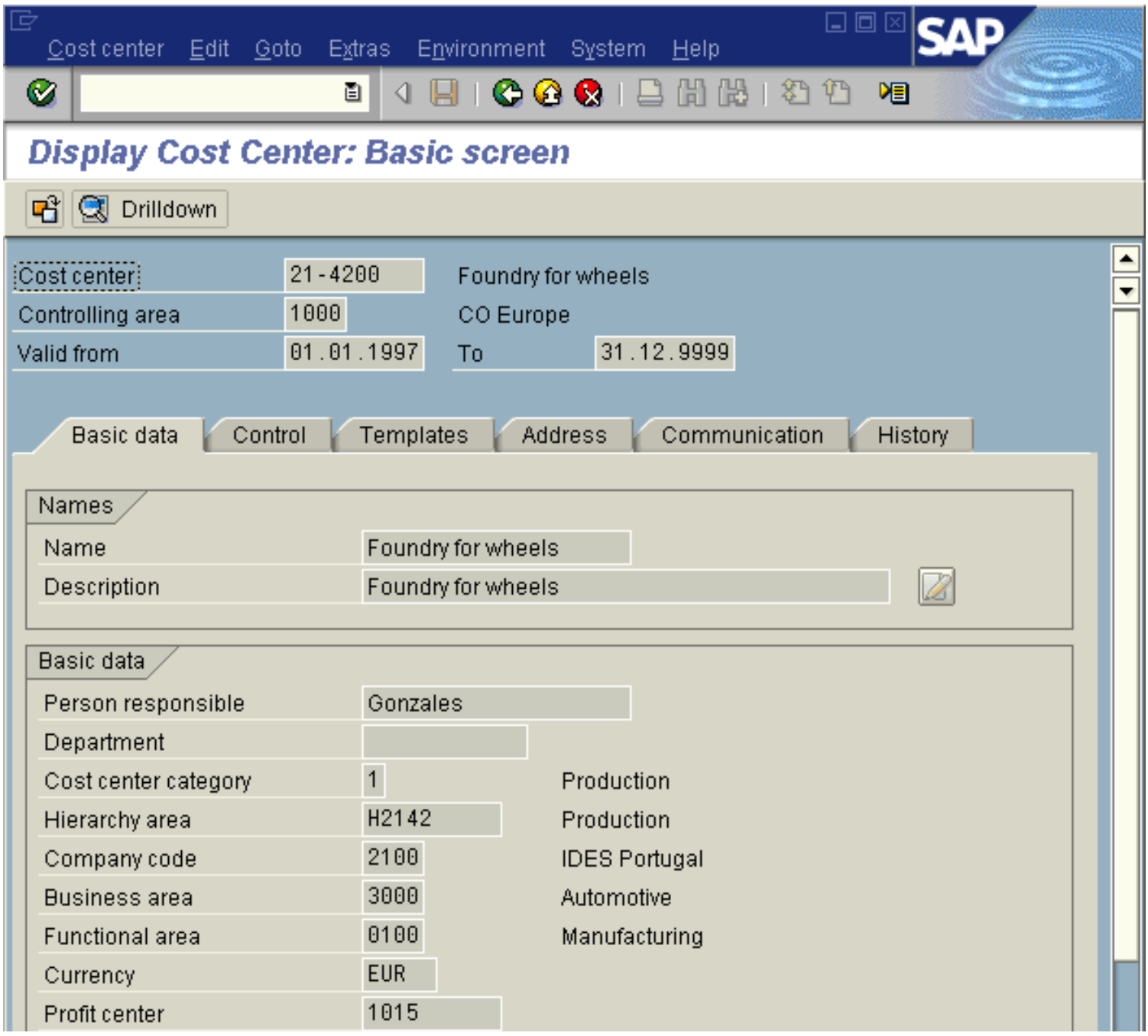
top

Tabstrips

Examples of Use

Display and Maintenance of Subobjects

Often, business objects in the R/3 System are maintained as subobjects, such as address data and bank details. Tabstrips are a good way of supporting users in their work. Subobjects, which typically are presented as boxed field groups or tables, can be put on page elements, so that the tabstrip combines all the maintenance options for the business object on one screen.
It can also be advantageous to spread options for very complex subobjects across several page elements, since the overall context of the application is not lost.
By creating hierarchical subscreens representing the content of page elements and modifying the tab indexes at runtime, you can create very dynamic subobject structures that can change with the task context (Customizing, initial screen, others).
Caution: Do not use tabstrips if users should do their work on subobjects in a mandatory sequence or if a particular required sequence is determined by the system depending on input.
An example of working on subobjects is cost center maintenance. This is also an example of the use of icons as labels for tab indexes.



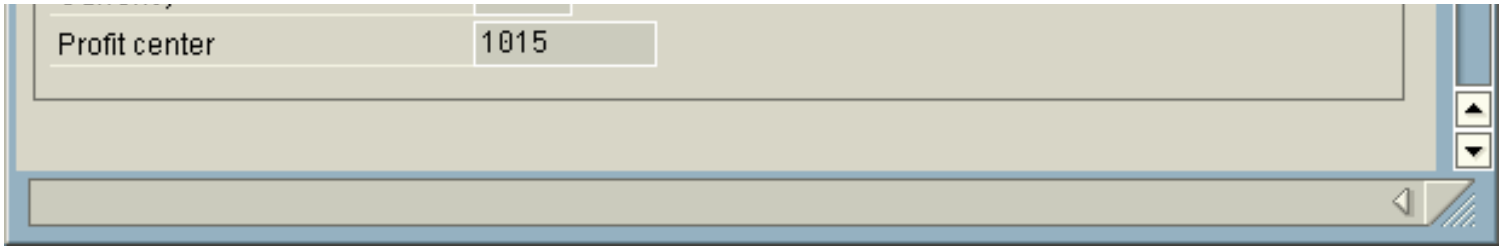
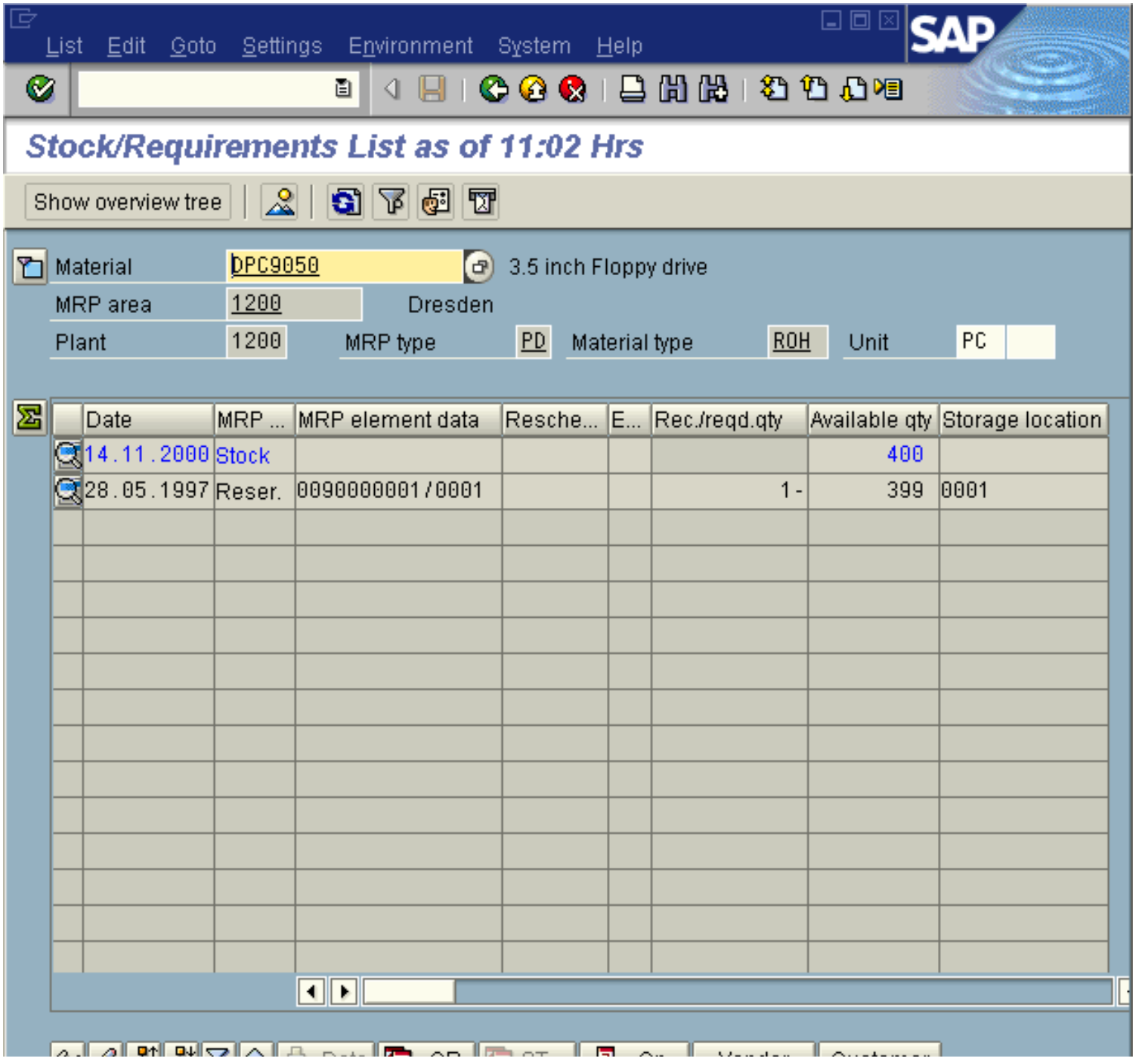


Figure 1: Tab Strip for Subobject Maintenance: Attribute Groups in Cost Center Maintenance

Combining Tabstrips with Other Screen Elements to Present Large Amounts of Information

Sometimes it is necessary and appropriate to offer a large amount of additional information on an object that is to be worked on. In the past this was typically achieved using dialogue boxes. If the additional information was very extensive, the only solution was often another primary window: The disadvantage was that the context of the task was lost. Stock/requirements lists provide a very good example of how a tabstrip can be used with other screen elements to present extensive header data on an object to be processed without losing the working context.



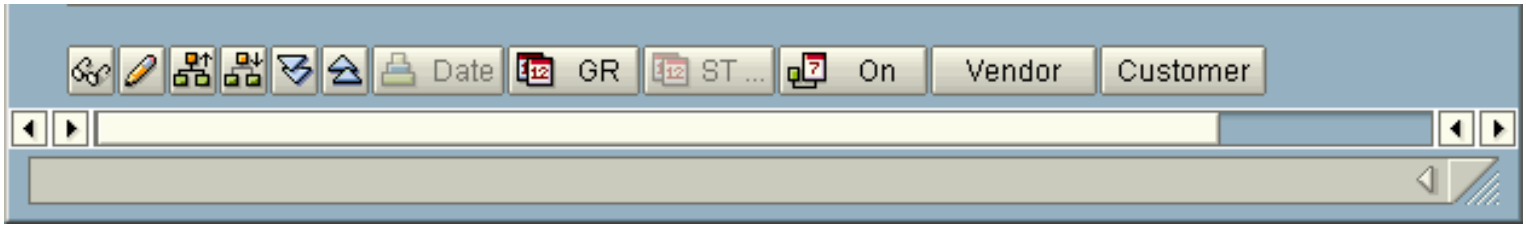


Figure 2: Overview Screen with Standard Header Data

The arrangement of the very extensive header data in the tabstrip means different views of the header data can be presented while retaining the working context.

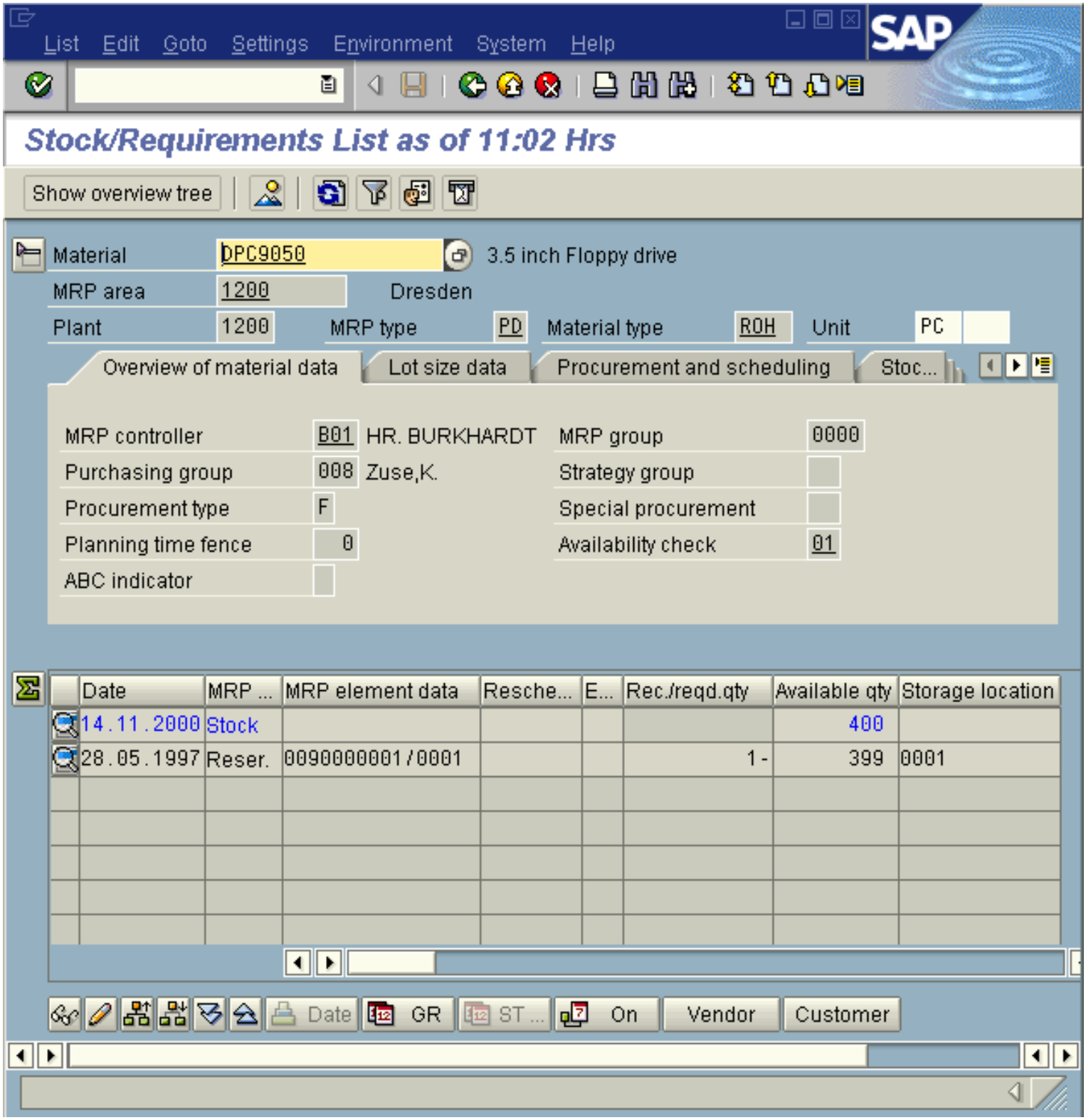


Figure 3: Overview Screen with Extensive Header Data

top

Source: SAP R/3 Style Guide

Functions

Accessing Functions | Keyboard Access

Functions are the low-level building blocks of the R/3 System. They provide the elementary business actions like creating or editing a document or printing a report. Functions differ in their complexity. Some functions perform only very simple tasks. These are, for example, the editing functions cut, copy or paste. The elementary navigation functions for moving from one screen to another are also simple functions. Other functions, however, may initiate complex business processes. As we already stated, while the menu tree provides the skeleton for an R/3 application, the functions fill it with flesh.

One important aspect of functions is to standardize them. This helps to limit the possible universe of different functions in the R/3 System to an overseeable number. These standard functions have a predefined behavior that users will be able to learn and understand. Thus, standardization of functions allows users as well as developers to use these functions in a specified manner and with a specified meaning. Standard functions are an effective means to make the R/3 System consistent.

For the R/3 System more than 50 standard functions have been defined. These standard functions are widely used in R/3 applications. So, users feel familiar with them and remember their meaning as well as how to access them. That is, they know where to look for them in the menus and what they are good for.

For an overview of the R/3 standard functions, their names and access have a look at the *SAP Reference Lists* on the *SAP Design Guild*. They tell you a function's function key, access character, and way to assign a pushbutton to it.

 top

Source: SAP R/3 Style Guide

 Functions

Accessing Functions

Functions may be accessed in a variety of ways in the R/3 System. They may be called by

- menus
- function keys or the function key menu
- pushbuttons, either in the standard toolbar, the application toolbar or by pushbuttons directly placed on the screen
- access characters (in the menus)
- fastpaths (sequences of access characters entered in the command field)

You define in the Menu Painter where a function is placed in the menu and how it can be accessed. If you do this according to your "personal taste", you may confuse the users. They rely on the correct placement and access mechanisms, because not only your application follows the guidelines described in this book, but practically every R/3 application as well.

Look out for some pitfalls, however:

- some functions do not appear in the menus, because this makes no sense
- access characters have to be unique within one pull-down menu
- the order of the functions in the application toolbar has to be observed
- some functions have function key directly "tied" to them, while others may be assigned to arbitrary functions keys; still others have no function key at all

In addition, there are a number of general questions that need to be answered, when you design the general menu structure and consider how the user can access functions:

- Which application-specific functions should be placed into the application toolbar for easy access?
- How well can the demand for a stable task menu bar be fulfilled? This is especially critical when parts from other developers are included in your own task.
- Are the function names intuitive? Will users know what they stand for? This is in particular important for rarely used functions!
- Which application functions can be represented by standard functions?
- How should the pull-down menus be structured, when there are lots of application functions to be put into?

These questions demand some experience with the issues involved. Some of them can only be resolved in close contact with end users. This way you make sure that a consistent user interface can be established across related tasks and applications.

 top

Source: SAP R/3 Style Guide

 Functions

Keyboard Access

There are two main methods for accessing functions by keyboard, the access characters and the function keys. The R/3 System provides another keyboard access method based on access characters, the fastpath. Furthermore, on certain computer platforms like Windows™ or Motif you can even access pushbuttons with the keyboard.

Access Characters and Fastpaths

Functions may be called from the keyboard via their *mnemonic access characters*. You access a function by pressing the ALT key in combination with the respective access character. The access character is shown in the pull-down menu as an underlined character. Underlining is done automatically once you defined an access character in the Menu Painter. For the standard functions the Menu Painter proposes access characters that conform to the SAP Style Guide. For other functions you have to choose the access character yourself. Make it unique (within a pull-down menu) and typical for the function. Often you can use the first character of the function name. If it is already used, try to find one that is easily associated with the function.

Access characters also represent a possible path, the so-called *fastpath*, for starting a task or initiating an action. To do this, you have to enter a period into the command field followed by a string of access characters. The sequence of access characters you enter in the command field begins with the menu bar item itself and ends with the desired function. It corresponds to the sequence of menu options you would have to choose with the mouse in order to initiate the desired action. Note that this, too, is a is a keyboard access mechanism for the function.

Function Keys

Function keys are another very fast and convenient keyboard access mechanism for functions, but with some differences to the preceding mechanisms. Function keys are special keys on a keyboard that are reserved for certain application functions. Therefore they are not mnemonics! As the number of function keys on a keyboard is limited, you have to observe some restrictions. In addition to pressing a function key these functions can be accessed via the *function key menu*. This menu pops up when the user depresses the right mouse button. The R/3 System supports up to 99 (V1-V99) function keys that are in part accessed via mnemonic shortcuts.

There is one important point to be observed: If you want to assign a pushbutton to a function which is located in the standard or application toolbar, you have to assign a function key to it first. In other words: You can only assign pushbuttons in the toolbars to functions with function keys assigned to them.

Every function key belongs to one of the following four function key groups:

1. SAP standard function keys (function always active, mandatory assignment)
2. predefined function keys, if active (mandatory assignment, if function is active)
3. recommended function keys, if active (recommended assignment, if function is active)
4. function keys with no specific function assigned to them (function keys without fixed assignment)

Look at the *SAP Reference Lists* on the *SAP Design Guild* for function keys, the list of virtual keys and the function assignments to virtual keys.

For the sake of consistency within an application, try to use standard function keys for functions active in several tasks of the application.

Pushbuttons

Pushbuttons in the Standard Toolbar | Pushbuttons in the Application Toolbar | Pushbuttons with no Fixed Position

Pushbuttons are "buttons" on the screen to easily and quickly access a function with the mouse.

Pushbuttons can occur at three locations in the R/3 System: in the application toolbar, in the standard toolbar, and in the work area. Therefore, you have to check for the proper location of the standardized functions. In addition, the following general guidelines apply, in particular to non-standardized functions:

- **Standard toolbar**: The standard toolbar contains cross-application, general so-called generic functions such as *Save*, basic navigation functions (*Cancel*, *Exit*, *Back*, scroll functions), local clipboard functions, search and print functions, and *Help*.
- **Application toolbar**: The application toolbar contains application-specific functions that refer to a screen as a whole, including navigation functions.
- **Pushbuttons with no fixed position**: The work area includes pushbuttons that refer to particular field contents, for example, pushbuttons for choosing a line or column of a table, for scrolling in a table, or functions which sort and total certain fields.

You define pushbuttons to appear in the toolbars in the Menu Painter. For these pushbuttons, the system decides automatically if a task-defined function is displayed in the standard toolbar or application toolbar.

You define pushbuttons that appear in the work area of a screen in the Screen Painter. Here you can also add an icon to the pushbutton.

The "Possible entries" pushbutton and the "More" pushbutton on selection screens are further examples for pushbuttons.

 top

Source: SAP R/3 Style Guide

Pushbuttons

Pushbuttons in the Standard Toolbar

The standard toolbar contains pushbuttons with icons for important cross-application functions (so-called generic functions) that can be quickly accessed with the mouse. It also contains the command field for entering a fastpath or transaction code.

The standard toolbar contains the following pushbuttons and controls in the order specified (from left to right):

- OK/Check = ENTER
- the command field
- *Save* (F11/V11)
- navigation functions: *Back* (F3/V3), *Exit* (F15/V15), *Cancel* (F12/V12)
- search and print functions: *Print* (F13/V86), *Find* (V71), *Find next* (V84)
- local clipboard functions(from 3.0d on): *Select mode* (V90; Windows™ and Apple only), *Cut* (V91), *Copy* (V92), *Paste* (V93)
- scroll functions: *First page* (F21/V80), *Previous page* (F22/V81), *Next page* (F23/V82), and *Last page* (F24/V83)
- system functions (from 3.0d on): *Session manager* (V94; Windows™ 95/NT only)
- *Help* (F1/V1)

Figure 1: The standard toolbar

All pushbuttons of a group are positioned close together

The corresponding functions are defined in the Menu Painter and automatically placed into the standard toolbar. Look on the list of Standard Functions for additional informations in the *SAP Reference Lists* on the *SAP Design Guild*.

top

Source: SAP R/3 Style Guide

Pushbuttons

Pushbuttons in the Application Toolbar

The application toolbar contains functions that are specific for an application. It provides a convenient access to these functions with the mouse. The application toolbar is located directly below the standard toolbar.

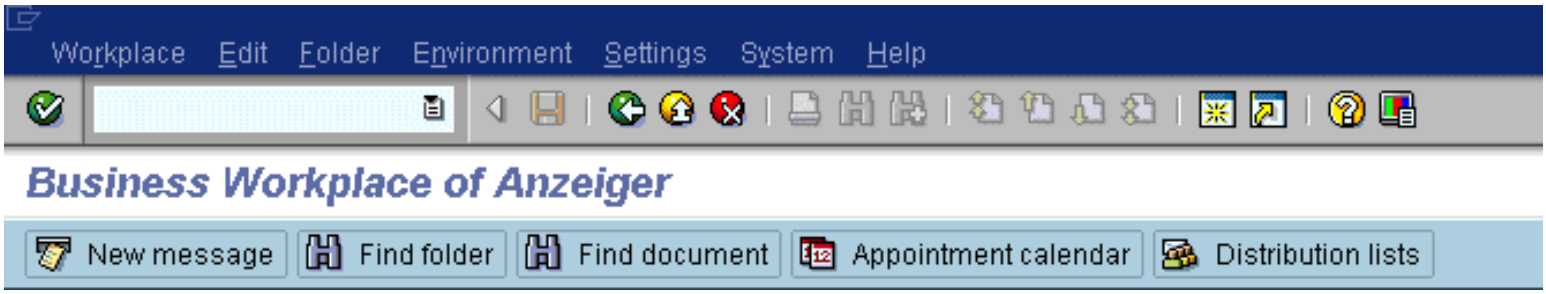


Figure 1: The application toolbar is located below the standard toolbar

Guidelines

- Try to display all functions as pushbuttons that are important for a task. To fit them into the toolbar, use appropriate abbreviations or icons.
- Put only functions into the application toolbar that apply to the whole screen. Present functions that apply to special fields or groups of fields on the screen as pushbuttons directly on the screen.
- Self-defined function keys are important for the application, but usually refer to less known functions. Display them in the application toolbar, because they are harder to learn and to remember. They should be presented visually to the user as a reminder.
- Furthermore, you can display navigation functions for changing the hierarchy level of a task in the application toolbar.

Icons on Pushbuttons in the Application Toolbar

If an icon is available for a function, include the icon in the pushbutton either with or without text label.

Figure 2: Example for functions with icons in the application toolbar

You assign an icon to a function that is positioned in one of the toolbars in the Menu Painter. The Menu Painter assists you and makes suggestions for icons. However, you should check these suggestions, before you assign an icon to a function.

Generally, most icons on pushbuttons should be used without a text label. In addition, use them only for the suggested meanings in order to not confuse the user. Avoid puzzles, that is, do not combine icons and text to form new concepts. This may only be acceptable for standard functions like *Display*, *Delete*, *Print*, *Header*, *Overview* etc. that the users know well or where the icons aid in classifying functions.

Icons without Textual Label

The following functions use icons without a textual label:

- *Execute*, *Find*, *Replace*, *Copy*, *Delete*, *Sort*, *Total*
- On detail screens: *Previous entry (item, record)*, *Next entry (item, record)*
- With screen sequences: *Previous <object component/screen>*, *Next <object/component/screen>*
- With reports or hierarchies: *Expand*, *Collapse, Previous/Next hierarchy level*
- With editing objects: *Other <object>*, *Create*, *Change*, *Display*, *Display <-> Change*
- For selections: *Select all*, *Deselect all*, *Select Block*

- Additional Functions (not yet standardized): *Compare*, *Filter*, *Rename*, *Find (next)*, *Transport*, *Export*, *Import*

Icons with Textual Label

If there is enough room, you can include icons with a textual label for special functions such as *Refresh*, *Convert*, *Skip*, or *Retrieve*. Add text to the icon, when the icon only classifies the class and a description is required to understand the full meaning of the function.

Check if the pushbutton can be positioned next to the object in the work area. Include only icons for application-wide functions in the application toolbar.

Order of Pushbuttons in the Application Toolbar

Try to observe the following order of pushbuttons in the application toolbar:

1. If applicable: icons for the functions *Previous screen*/*Next screen* and then *Previous <object component>*/*Next <object component>* (or *item*)
2. Icon for the function *Choose*=F2
3. Pushbuttons with icons alone
4. Pushbuttons with icons and text labels
5. Pushbuttons with text labels alone
6. If applicable: less important functions with any of these presentation forms

Try to present application-wide used functions with icon alone, that is, to put them into group 3.

If in doubt, an aesthetically pleasing overall impression decides if the icons should have a text label or not.

Icons on Pushbuttons in the Application Toolbar of Dialogue Boxes

Pushbuttons in the application toolbar of dialogue boxes can have icons, too. The following guidelines apply to pushbuttons with icons in dialogue boxes. They apply also to dialogue boxes without an application toolbar.

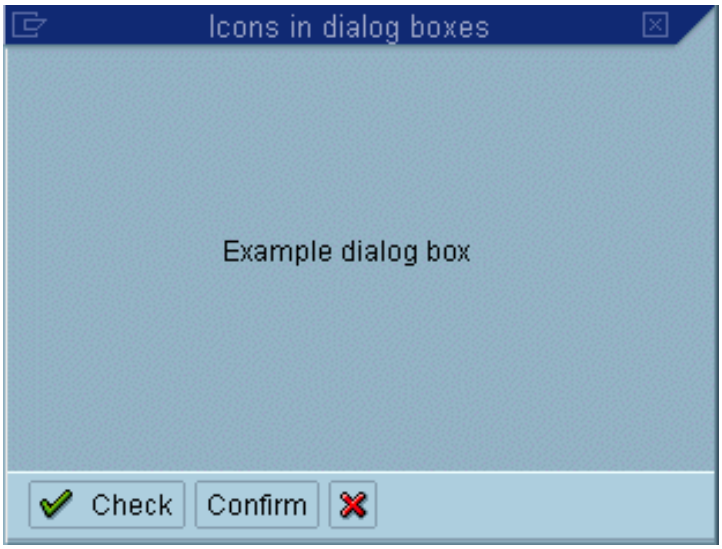


Figure 3: Example of pushbuttons with icons in the application toolbar of a dialog box

There are icons reserved for functions in dialogue boxes like the icons for the functions *OK*, *Cancel* and *Print*. All other functions use their standard icons.

Guidelines for Pushbuttons with Icons in Dialogue Boxes

- **The default key (ENTER)**: In general, the ENTER key is only assigned the checkmark icon for the functions *OK*, *Continue*, *Close* or *Execute*.
  When the *Choose* and/or the *Configure* functions are assigned to the ENTER key you should use the icon with a text label, for example "ICON_OKAY" plus text "Choose". For specific functions on the ENTER key such as *Delete, Check, Refresh* or *Find* use the checkmark icon plus text.
- **Cancel**: *Cancel* is only assigned an icon.
- **Other functions**: If there is enough room, add text to the icon. Otherwise use an icon without a text label.
- **Exceptions**: If the dialogue box displays text only (for example, confirmation prompts), you should use an icon with a textual label. Message dialogues do not have icons on the pushbuttons.
- **Sequence**: ENTER - dialogue box specific pushbuttons - *Cancel*

Position pushbuttons which have an icon but no text directly behind the ENTER button. All other buttons or icons follow.

top

Source: SAP R/3 Style Guide

Pushbuttons

Pushbuttons with no Fixed Position

Use pushbuttons with no fixed position to initiate functions that refer to individual fields, groups of fields, tables or lists and/or parts of them. You can use pushbuttons with no fixed position for example to show or hide information (texts, graphics, animation). On initial screens, use pushbuttons in the work area to call the most important functions that can be executed with the chosen objects.

Data Screens

Use pushbuttons on data screens for calling further information, execute immediate actions or, for instance, setting attributes. According to their positioning, we distinguish field-related, group-related and table related pushbuttons.

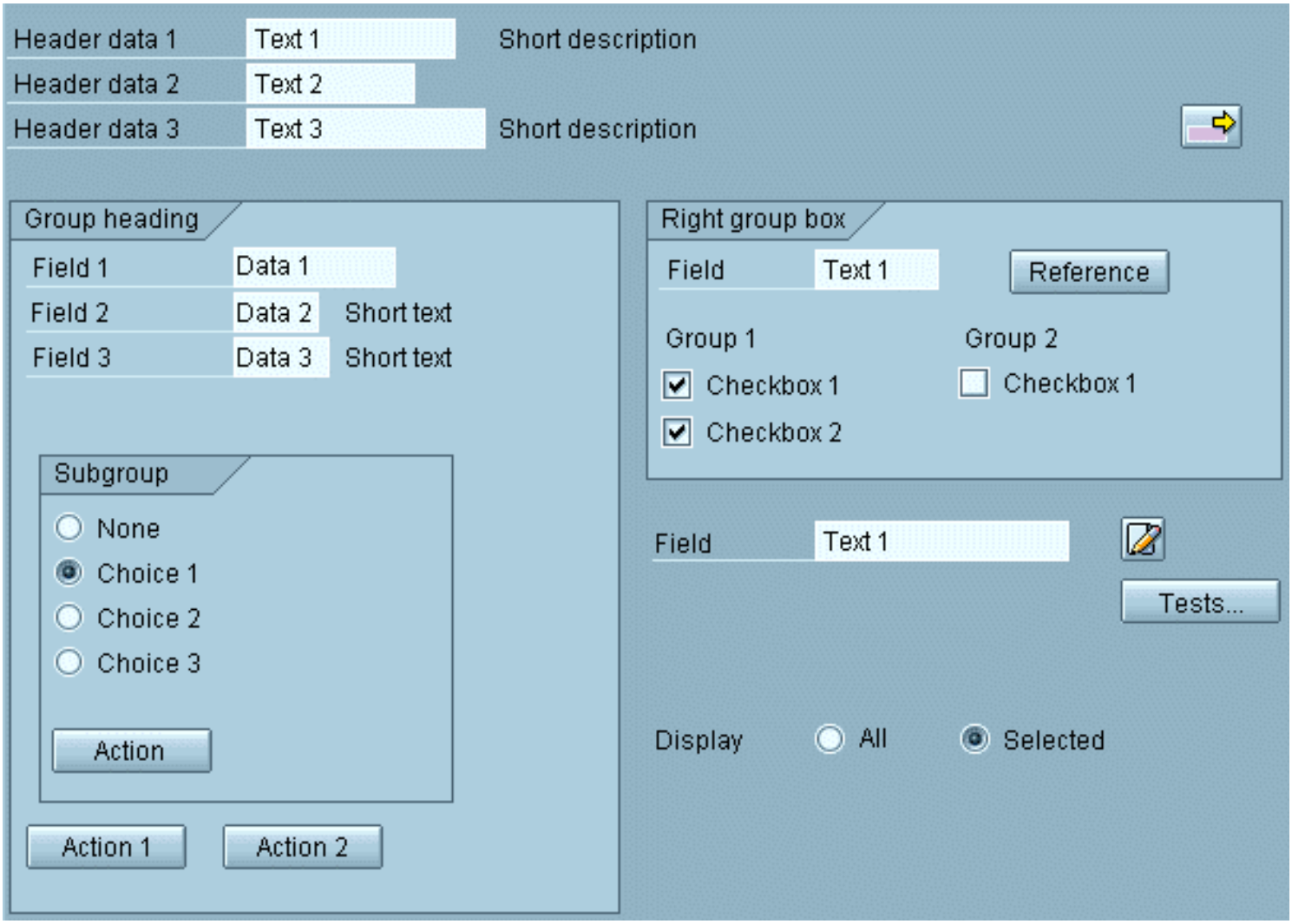


Figure 1: Examples for pushbuttons on screens for calling detailed information, for initiating actions and for setting screen attributes

Field-related pushbuttons refer to a single field or to a small number of fields. They are usually placed to the right of the field that they belong to. Align pushbuttons that are on top of each other. If placed in a group box, place them to the right of the group box, if possible. Yet, you should never place them too far away from the field, so that users would not know which field they refer to.

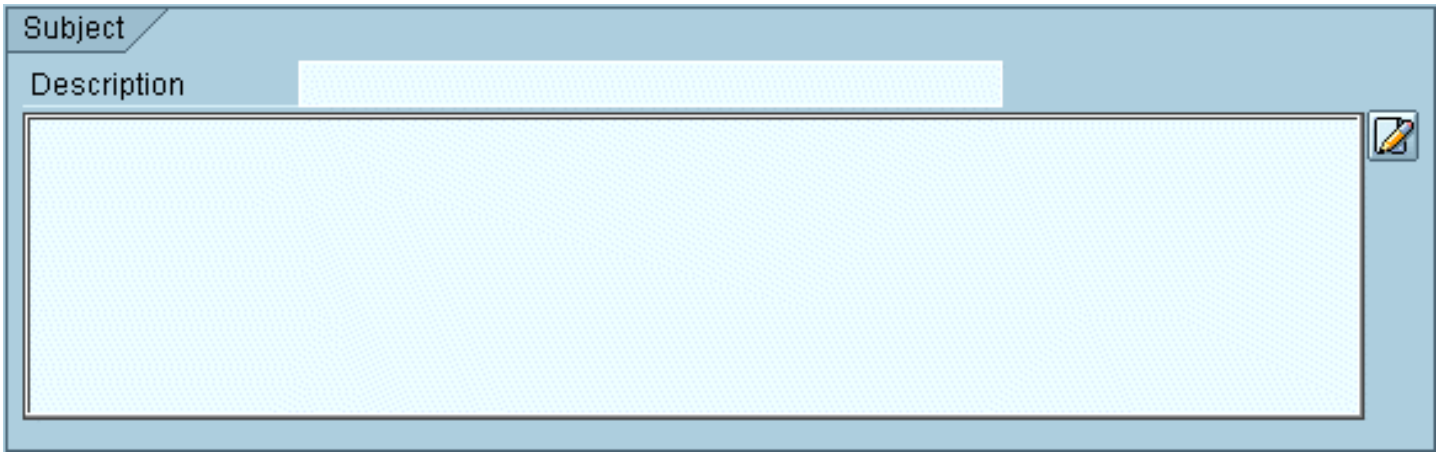


Figure 2: Pushbutton on the long text as an example of a field-related pushbutton

Group-related pushbuttons refer to a group of fields. Left-align them with the upper fields of the group box and let them generally start in the first three columns of a group box.

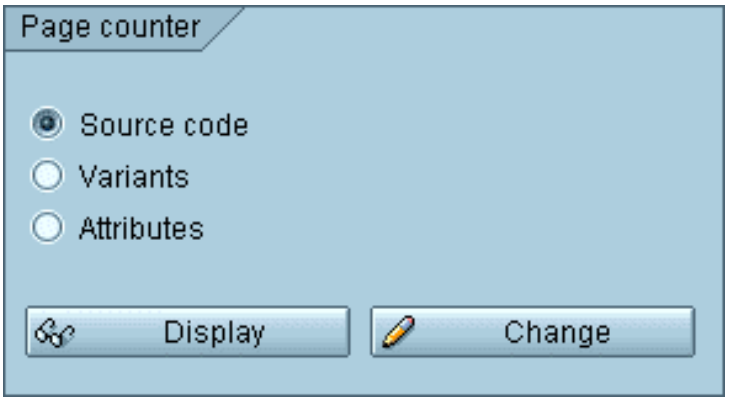


Figure 3: Example for group-related pushbuttons

Table-related pushbuttons refer to a table. Separate them by one line from the table and usually include them in the group box of the table if it is provided. Such pushbuttons can be scroll functions, edit functions or functions for the positioning.

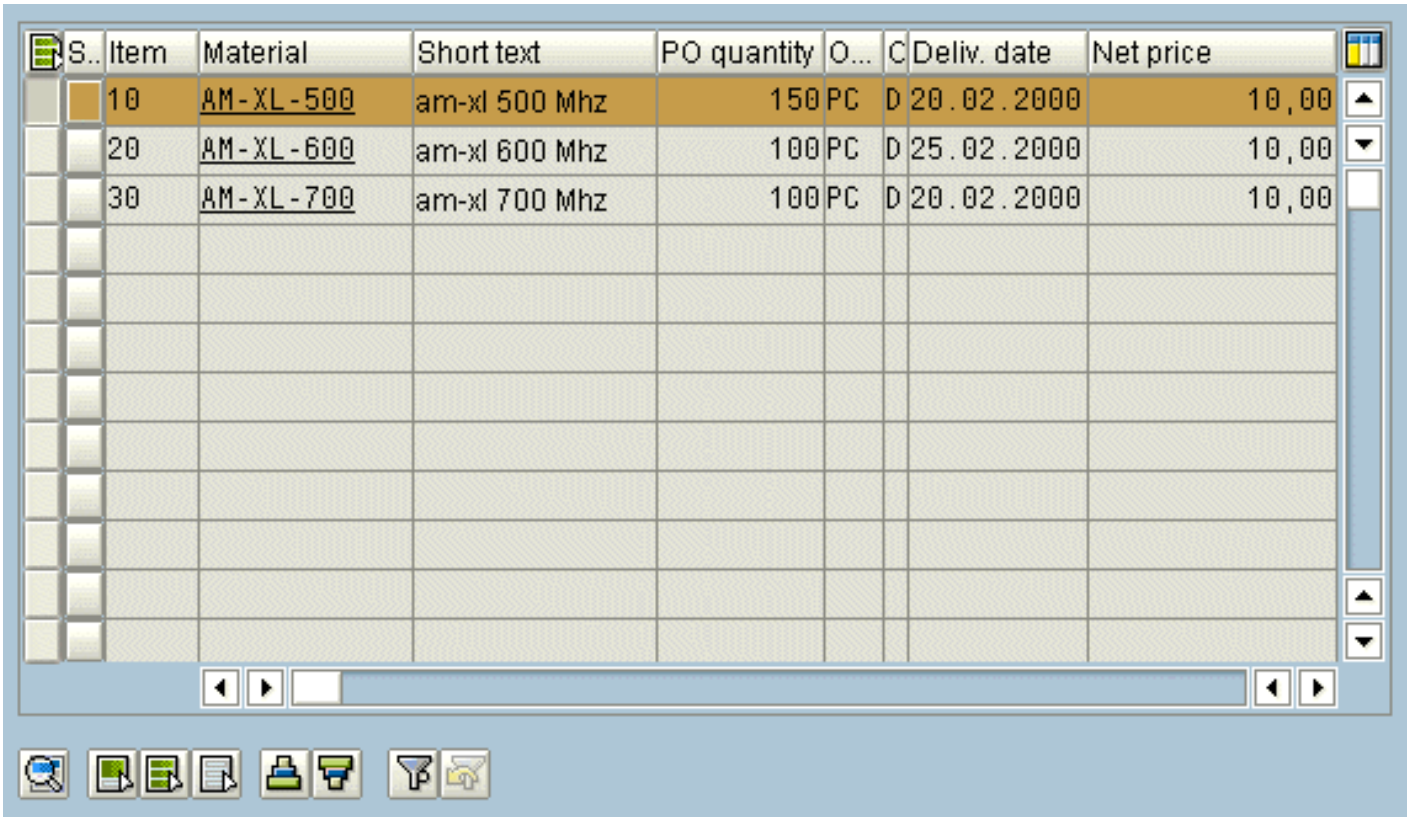


Figure 4: Pushbuttons for Table Control

See also Pushbuttons with Icons on Data Screens

Initial Screens

The initial screens of tool tasks differ from those of business applications in respect to the number of processing methods possible in a transaction. In tool tasks quite a number of object components or processing types are available. Therefore, these tasks differ in their initial screens from business applications by presenting the object components or processing types in the work area and not in the application toolbar. You may use the application toolbar for other less important functions, here. In addition, the object components or processing types are listed in the menu so that they are also available in the data screens.

If the user can only select one object component at time, design them as radio buttons, otherwise as checkboxes.

If processing types refer to the entire object, position the pushbuttons to the right of the input field for the object. If they refer to individual object components, arrange them within the group box that contains the object components or close to the object components.

If there is only one processing method, you need not provide an *Edit* pushbutton in the work area; in this case the ENTER pushbutton will do. On the other hand the number of pushbuttons should not exceed 10 in order not to lose transparency of the initial screen.

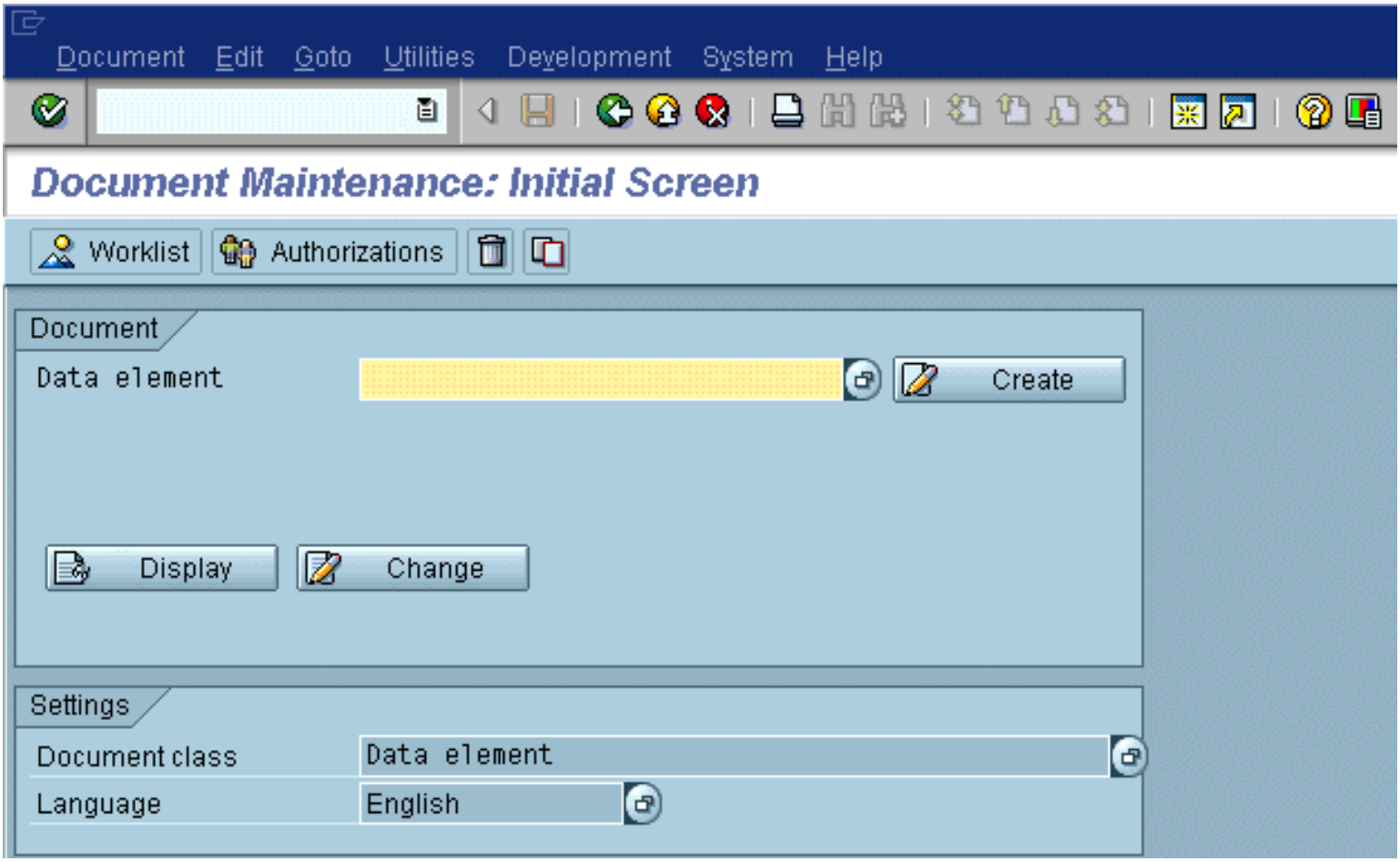


Figure 5: A typical initial screen

See also Pushbuttons with Icons on Initial Screens

top

Source: SAP R/3 Style Guide

Pushbuttons on Data Screens

Pushbuttons with Icons on Data Screens

On data screens, you can use icons for pushbuttons with no fixed position for a number of functions. Position the pushbuttons in the work area so that the intended reference is obvious (usually to the right of an object).

Generally, add text to an iconized pushbutton only, if absolutely required.

If the address icon is used for different types of addresses (e.g. shipping address, order address), complete the icon with a corresponding text.

Spin Buttons

A spin button is a control element for the sequential display and/or input of mutually exclusive alternatives. Use spin buttons to display or enter alternatives with a logical ascending order, for example, the months of the year.

You provide the functionality of a spin button using an input field and two triangle icons pointing up and down.

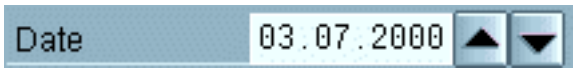


Figure 1: Spin button example

When choosing **alphabetical** entries (for example, month, weekday) the upwards pointing triangle means *Previous entry* and the downwards pointing one *Next entry*.

With **numeric** values, (for example, the rotation angle of an object, time) the upwards pointing triangle means *Next value* (*Increment*), the downwards pointing one *Previous value* (*Decrement*).

At the beginning or the end of the list the system cycles through the other end of the list.

Word Processing Functions

Text/Word Processing

- **Standard**: Usually, you use the *Change text* icon, and *Create, Change* and *Display* are not distinguished. If the user clicks this icon, the SAPscript word processing is called or an intermediary screen comes up from where SAPscript can be called. The user can enter formatted texts. This function may also call an OLE object (WinWord, Wordperfect etc.). It refers to the entire object (formerly often called *Long text*).
- **Extension**: There are icons with the meaning *Display text* and/or *Create text*. Use them to set the three text icons at runtime. This makes a separate status display superfluous. Follow this scheme for dynamic use of word processing icons:

Status of Text	Icons for Edit Mode	Icons for Display Mode
no text	*Create text*, **active**	*Create text*, inactive
no text, protected	*Create text*, inactive	---

text exists	*Change text*, **active**	*Display text*, **active**
text exists, protected	*Display text*, **active**	---

Table 1: Overview of text icons in edit and display mode

(Change) Note, Comments

- **Standard**: Use the annotation icon when the user can enter comments with reference to a particular action (object component, for example, order item) or a particular line of a report. Typically, these texts are entered unformatted.
- **Extension**: As with text (word processing): Create note and/or comment and/or Display note and/or comment

Other Services

For other services you can use the following functions:

- **Table calculation**: Calling, for example, the unit calculation from a previous application, for example, "Change part"
- **(Business/Statistics) graphics**: Calling the business graphics from a window in which displays a list with data
- **Variants**: Calling a dialogue box, in which layout variants can be specified (for example, in the Report Writer or in Logistics-Controlling) or display variants can be selected (for example in the line item display)
- **Info**: Information text from the application itself (that is, created by SAP or generated by the application or a data processing administrator). Texts can refer to the application or to individual columns in a report. They describe specific aspects of an application and cannot be changed by the user.
- **Short Message**: May be used in mail applications
- **Address**: Calling the address dialogue box
- **(Set) status:** Calling a window to set the status for an object (for example, entering release infos in the IMG)
- **Tools**: Callings tools inside or outside the R/3 System
- **Export/Import**: Saving files on a PC/loading files from a PC

top

Source: SAP R/3 Style Guide

Pushbuttons on Initial Screens

Pushbuttons with Icons on Initial Screens

On initial screens, assign the text of the following pushbuttons for generic functions an icon:

- Create
- Change,
- Copy <object>
- Delete (<object>)
- Display
- Display <-> Change
- Execute
- Replace
- Find and Find next

If there is enough room on initial screens, display these functions always as icons with a text label.

Arrange the pushbuttons like this:

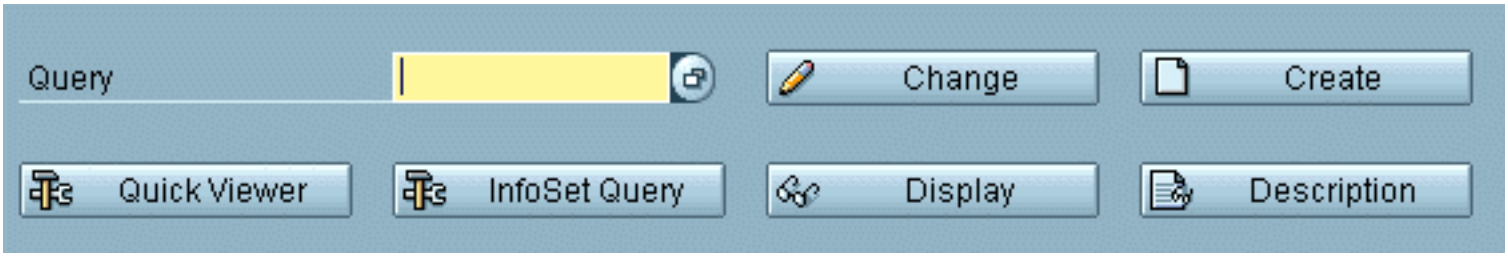


Figure 1: Initial Screen example

This applies basically to initial screens of tool applications, where the above mentioned functions are implemented as pushbuttons with no fixed position.

Application-Specific Functions

There are also quite a number of icons for application-specific functions, especially for the ABAP Workbench and for hierarchies.

Lists - Selection Screens

To improve the appearance of list selection screens, there are icons for operators and functions for use especially on these screens. In addition, you can use these icons in the *Selection options* dialogue box as field names with green, yellow or red background, depending on the context.

You can use these icons for pushbuttons and as a replacement for field names that appear flat on the template.

top

Source: **SAP R/3 Style Guide**

Context Menus

Context Menus and Menu Buttons

Context menus are shortcuts to often used functions.

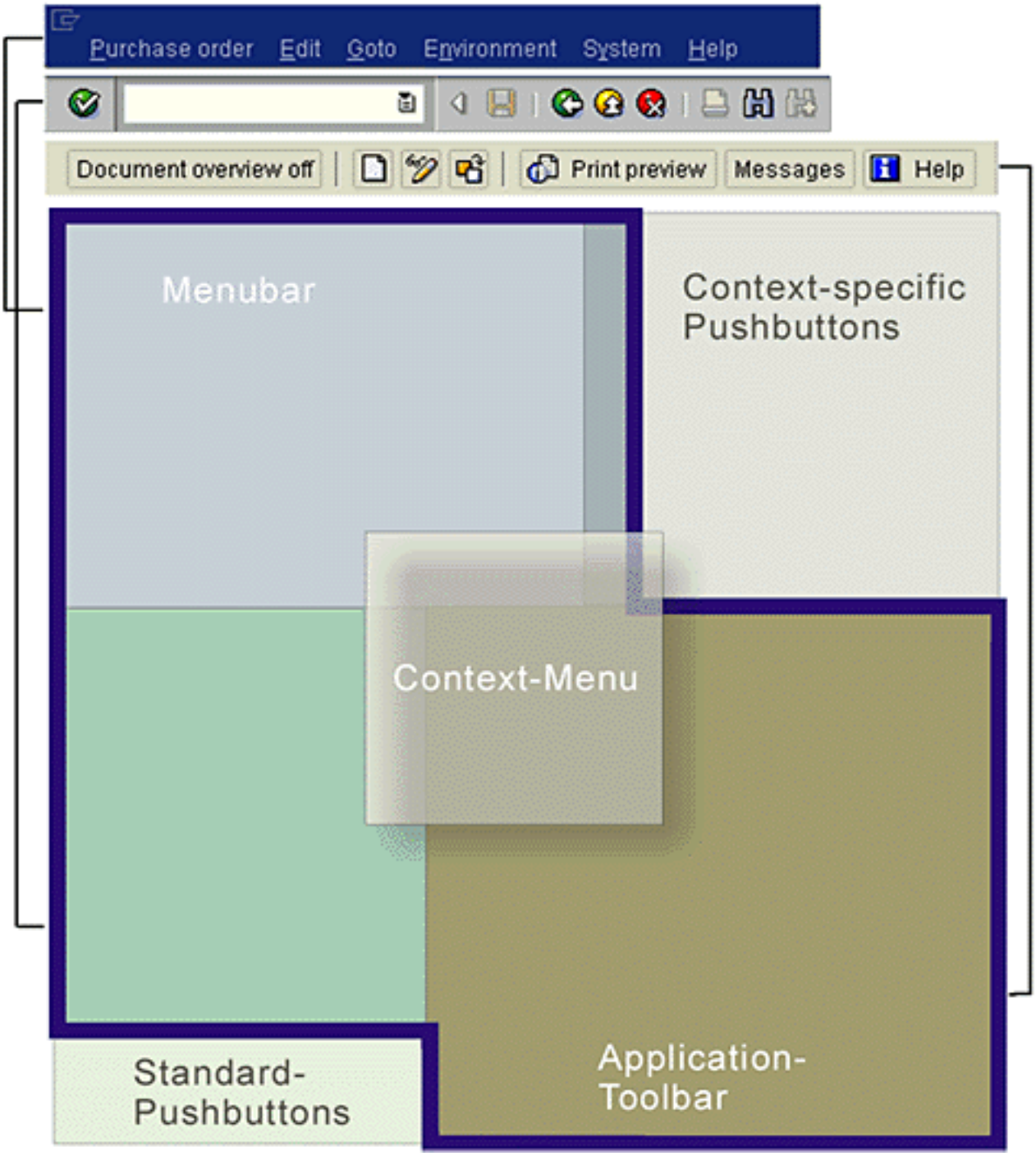


Figure 1: The diagram shows how context menus fit into the general framework of accessing functionality.

Recommendations for the Use of Context Menus

Structure

In principle context menus are structured like usual dropdown menus. They may contain the following elements:

- Menu entries (text),
- Icons (in front of the text),
- Ellipses (following the text, indicate that a dialogue box is called),
- Arrows (following the text; indicate that the menu contains a further menu level which is displayed as a cascading menu),
- Keyboard shortcuts (for example Ctrl-F2; follows the text),
- Separator lines for clustering the menu items into useful groups.

These elements can be combined according to the requirements of the control or application which supplies the context menu. The next two figures show examples for possible context menus.

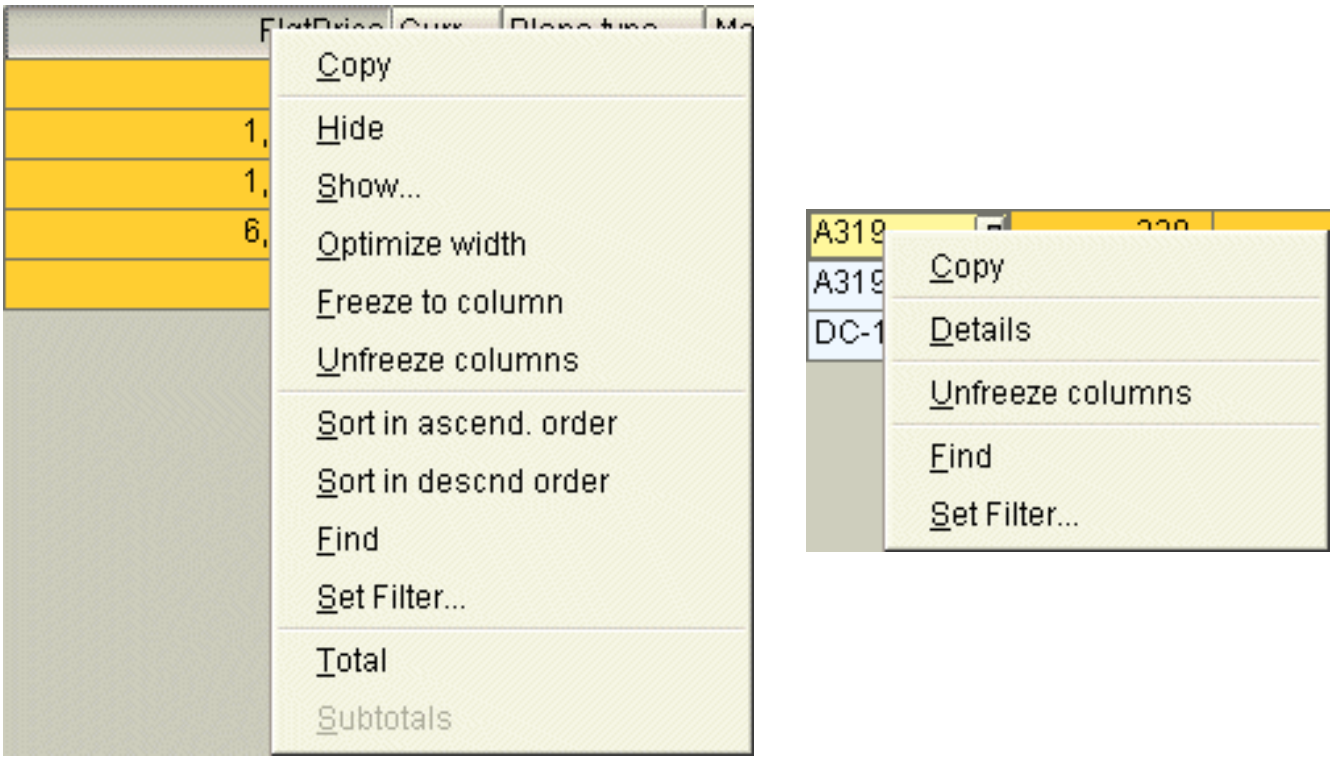


Figure 2: Examples for context menus

If there exits a default action for a context menu, this action appears as the first entry of the menu item list.

Context menus must not contain functions which are not also presented in other system locations. That is, a function must not appear in a context menu only.

Submenus

Context menus may contain submenus, but only one more level. Hierarchies with more than two levels are not allowed.

Number of Items

A context menu should not contain more than 10 items.

Completeness

A context menu should contain the complete set of pushbuttons that exists for a given object (e.g. tables / ALV lists).

"Default" Menu

On the screen background or on other undefined screen areas, the context menu only displays the item "No entry".

This way users receive feedback saying that there exists a context menu functionality, but that at this location no sensible actions can be provided.

Default Functions

All text fields (e.g. also table cells) contain at least the items "Cut", "Copy", "Paste" and "Select All".

Keyboard Shortcuts

Context menu items should display **mnemonics** for keyboard shortcuts, e.g. Ctrl-V.

For improving keyboard accessibility, all menu items should also have a "**hot key**" (accelerator key) assigned to them, that is, a letter key that directly calls a function. Accelerator keys are underlined; they must be unique for a given context menu.

Sequence

Functions specific for the object where the right mouse button was pressed have to be located at the beginning of the menu item list. Then transfer functions (*Cut*, *Copy*, *Paste*) and - if supplied - additional functions follow.

Inheritance

Menu items are only inherited, if a single element (e.g. a checkbox) does not have a context menu of its own. In such a case, for example, the surrounding group box supplies the context menu:

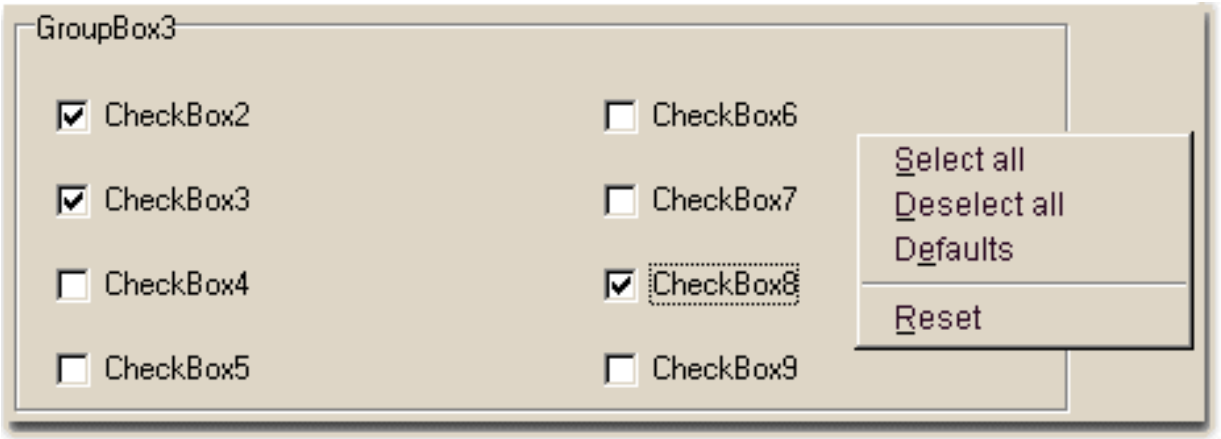


Figure 3: Context menu supplied by a group box

Functions assigned to the left mouse button (e.g. *Select*) should not be replicated for the right mouse button. **Exception**: *Select All*.

 top

Source: SAP R/3 Style Guide


Context Menus

Context Menus and Menu Buttons

Context Menus are suitable for expert users who want shortcuts. Do not use them for Options that cannot be chosen any other way and Users who are not computer experts and are not familiar with popup menus.

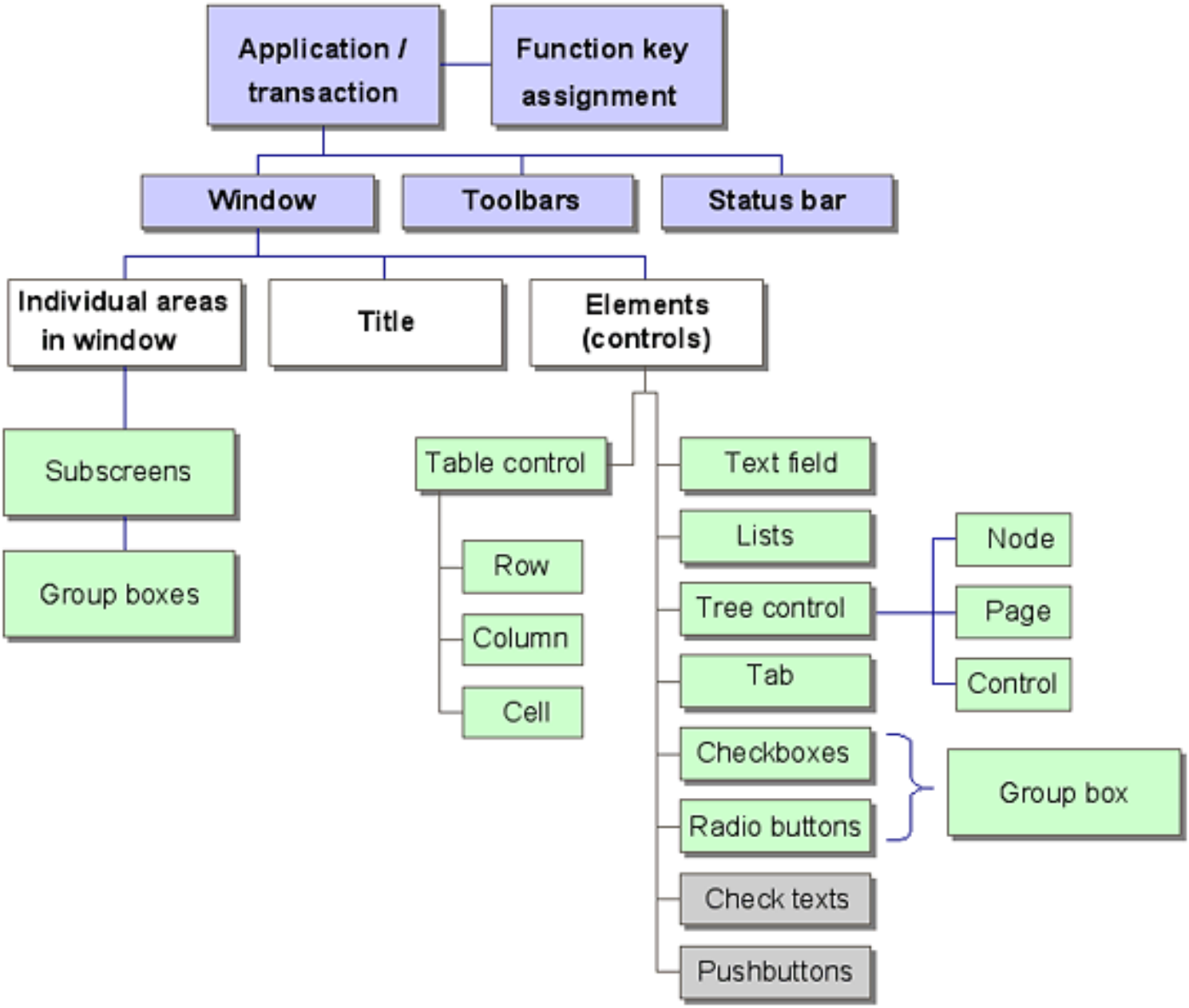


Figure 1: Possible assignments of context menus

Legend

- Violet: Application level
- White: Window level
- Green: Element level

Elements

Available

- Function texts with access character
- Shortcut keys
- Separators for contextual/visual structure
- Cascading menu options

Possibilities

- Icons (only for functions with icons elsewhere in the system)
- "Radio button items" for graphic representation of information about sessions or about a status

Figure 2: Example of a context menu with icons

And the F-Keys?

Objective: List of function key assignments

- Removed from the context menu
- Put in Help menu (for example)

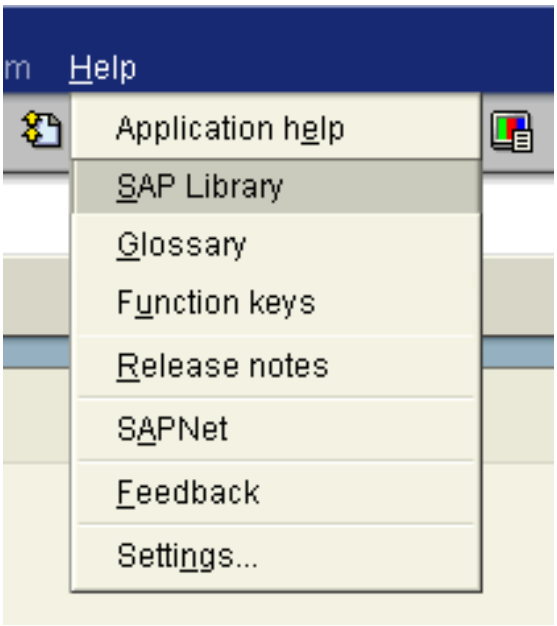


Figure 3: Functions listed in the help menu

Does and Dont's of Menubuttons*

Guidelines

- Menubuttons have more* functions
  - Display & Selection from within the menu
  - Define the default action
  - Use the button with the default action
- Menubuttons: What for ?
  - Selection from options palette
  - Differentiation of an action i.e.Delete (at button)
- Menubuttons must:
  - All functions must be contained within the menu
  - if default function has not been changed, the action stands first within menu
  - make default function visible on button

*) up to now only standard function 'pulldown menu' in ALV Tree/Grid

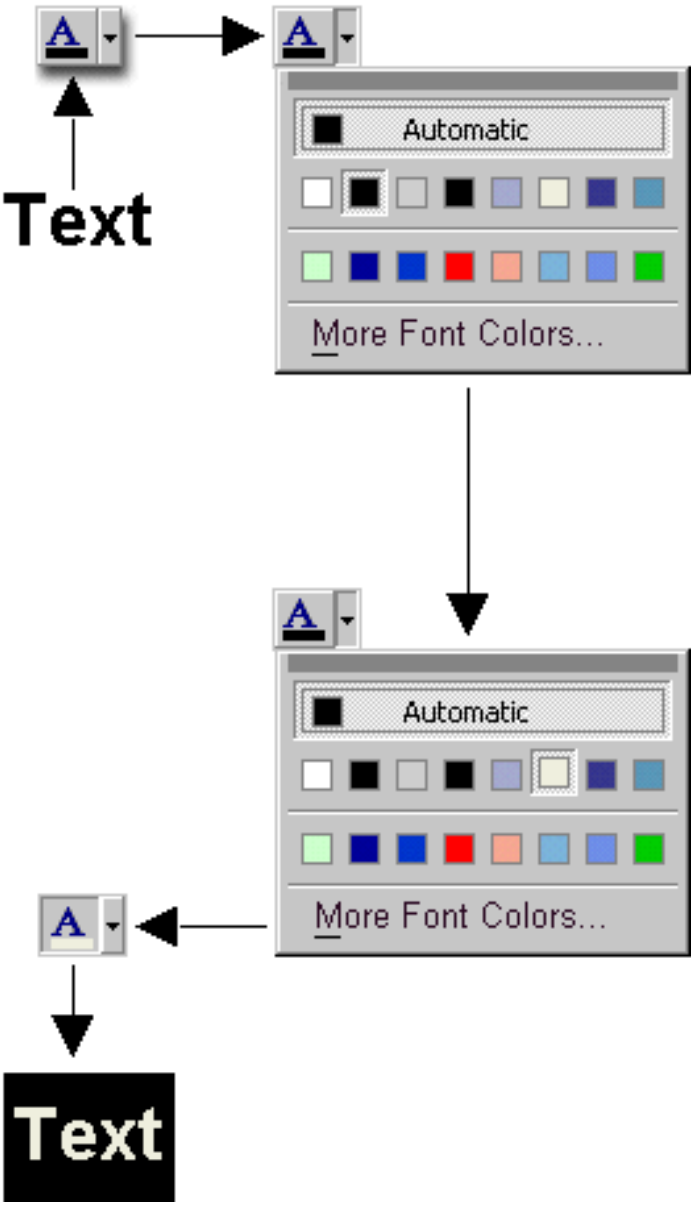


Figure 4: Illustration of actions triggered by a menu button

Other Functions

Application toolbar

Has only a few functions left:

a) For working with screen elements (example: tree)

b) For working on the whole object.

Caution: A menu with functions for using the application (even when the toolbar is hidden) is always required!

 top

Source: SAP R/3 Style Guide

General Guidelines for Dialogues and Texts

Although the R/3 System has a graphical user interface, texts, mainly single words, still play a very important role in R/3. Words give the screens and dialogues a "meaning", they tell what a screen is about and what it is good for. So, if you use some computer jargon or SAP jargon the users may not understand the screen, and they may not find for the things they want to do. As a result, they may not get their work done.

The ISO 9241 and many other regulations on user interfaces require that software systems are self-descriptive. This means among others that the system speaks the user's language and not some technical jargon. This aspect of user interfaces is far too often underrated. However, it comes up as a constant problem in user testing of the R/3 System. The following general guidelines should help you minimize terminology problems.

Speak the User's Language!

This is the most general guideline for designing field names, error messages, dialogues etc. Keep this guideline always in mind. Write it on a sticker and place it on your monitor!

Use Consistent Terminology!

Do not confuse users with different terms for the same thing. Do not name different things alike. It is also very important to keep terminology consistent between different applications and even between application areas.

Use Approved Terminology!

One means to keep terminology consistent between application areas is the SAP terminology database. This database is being constantly updated.

If there is an approved term in the terminology data base, use this term even if you personally dislike it. Users are familiar with this term and expect it in your application, too.

Think of the Context of Use and the Context of Translation!

Discussing interface terms often leads to heated debates, where personal preferences prevail. Often, the context in which a term is used is not taken into account. There may be similar terms in other applications or application areas with quite a different meaning. In some cases a term may already be used for other objects or actions. Sometimes the second best choice may be in the end the best one, provided all aspects are considered. The R/3 System is such a large system that you seldom can discuss a term in the context of a single application.

Translation is another important aspect that is often overlooked. Sometimes it is necessary that for instance English and German terms be quite different or even divergent. Different terminology may already have been established in these languages. Sometimes a direct translation has a different or even unexpected meaning in the other language. Often translators have to translate terms without reference to context of use. This may lead to a wrong translation if the terms in the original language have not been chosen carefully and unique.

 top

Source: **SAP R/3 Style Guide**

Dialogues

Prompts in Dialogues | Buttons in Dialogue Windows

The main purpose of dialogues is to ask the user for information or to provide him or her with information. Typical such examples are:

- The system needs information from the user to direct further processing.
- A critical situation arises where the user might lose data; the system therefore warns the user.
- The system informs the user on the success of an action or of the ongoing processing.

Make these pieces of information short and concise. It is important that the user understands them in order to prevent confusion or mistakes.

Dialogues take place in *dialogue boxes*. These are secondary windows that appear when the user activates a particular action in a primary or secondary window. They disappear from the screen when the user triggers a task-related action.

A dialogue box is classified as "modal" if the user must respond to the dialogue before continuing work in another window. Modal dialogues are useful, for instance, in situations where the user might lose data: The system informs the user of this situation in a modal dialogue box, and the user has to respond to this dialogue.

The R/3 System provides only modal dialogue boxes. In other systems there are also "amodal" dialogue boxes. With amodal dialogue boxes the user may continue work in the primary window, while the dialogue box is open. Typical such amodal dialogue boxes are dialogues for search and/or replace. In the near future the R/3 System will have amodal dialogue boxes, too.

The following explanation for dialogue boxes in the R/3 System takes into account that there are only modal dialogue boxes in the R/3 System. As a result of the modeness of dialogue boxes the menu bar in the primary window is locked, if a modal dialogue box is active on the screen. In addition, the dialogue box has no menu bar of its own. Actions are initiated via pushbuttons in the dialogue box solely.

Uses of Dialogue Boxes

The current main action within a transaction runs in a primary window to carry out the *Save or Post* action, for instance. Dialogue boxes supplement the main action by displaying additional information or data. They may also allow further entries, functional specifications, or selections in required-entry fields.

There must be no sequence of more than three dialogue boxes on the screen.
Exception: A dialogue box initiated with F4, that means possible entries may be displayed additionally.

Pushbuttons or Function Keys in Dialogue Boxes

Line pushbuttons up across the bottom of the dialogue box in the following order:

- ENTER=<Action>: always the leftmost button; always to be displayed
- Fx=Special functions: in ascending order from left to right
- F12=Cancel: always the rightmost button; always to be displayed, except in read-only dialogue boxes (but also active then)

The following buttons may be active but are **not displayed**:

- Scroll functions: F22=Previous page, F23=Next page, etc.
- Select operations: F4=Possible entries, F2=Choose (but see below!), F9=Select

The following buttons are **not allowed**:

- F3=Back, F15=Exit: Dialogues are side paths which can only be canceled with F12=Cancel.
- F11=Save: Saving can only be carried out in the main window. A dialogue box may prompt the user for saving, but the saving cannot directly be executed here.

Executing Dialogue Boxes

The user executes a dialogue box via the first pushbutton ENTER=<Action>. This is normally <Action>=Execute or ENTER=Continue. After pressing F4 from the primary window or from dialogue boxes that contain a selection list, it may also be ENTER=Choose. (**Note**: In this case, you have to reserve F2 additionally for *Choose* so that double-clicking with the mouse is also supported; F2, however, is not displayed.) Otherwise, you can also refer to the function executed directly (*Copy*, for example).

Provide dialogue boxes which contain a **question** with their own pushbuttons for the positive and the negative response. Do not display the ENTER key then. System messages that have to be confirmed receive a pushbutton with the name ENTER=Confirm, in the case of read-only dialogue boxes, ENTER=Continue.

Description of the Pushbuttons

ENTER Key

The dialogue box and the entries made are processed if the user presses the ENTER key. Then, the system either returns to the previous primary window or navigates to a new one. If needed, you may also display another dialogue box, while the previous dialogue box remains on the screen.

F12=Cancel

The dialogue box disappears and the program either returns to the primary window or to the last dialogue box (also if the user had to make required entries in the dialogue box). If possible, the previous status is fully restored, that is, the entries are "forgotten" and any processed entries are canceled.

Read-only dialogue boxes or dialogue boxes displaying information constitute an exception. F12=Cancel is actually active here, but it is not displayed.

F1=Help, F4=Possible Entries

F1 is always active and relates to individual fields, as in the primary window. After pressing F4, another dialogue box is displayed (the first remains where it is) with the respective possible entries.

Error Messages in the Dialogue Box

If the entries of a dialogue box are processed and the system detects an error, it can display an error message. Display the message in another dialogue box unless it is an S message which appears in the status bar. The user then has the option of correcting the input value in the dialogue box. That is, the dialogue box remains where it is, and the user receives the request to eliminate the error (or he cancels with F12).

Navigation in Dialogue Boxes

When displaying the dialogue box, place the cursor on the first field to be selected or in the first entry field. If no entry fields exist, place the cursor on the first displayed field which can have variables.

Positioning Dialogue Boxes

Position the first displayed dialogue box, if possible, so that the relevant information of the primary window is not covered up. If this is not possible, center the dialogue box on the screen if no other dialogue boxes can be displayed. Do not squeeze the dialogue box up against the edge of the screen.

Position further dialogue boxes to be displayed so that the title bar of the previous dialogue box is not covered up. That is, move them down at least two lines. In addition, move them a few characters to the left or to the right.

 top

Source: **SAP R/3 Style Guide**

 Dialogues

Prompts in Dialogues

Prompts are a special, and from the user's perspective an important type of dialogue box, because they help him or her to prevent mistakes in critical situations. For instance, if the user executes an action with potentially destructive or irreversible consequences, the system alerts the user to the critical situation and sends a prompt. This prompt informs the user about the current situation and provides him or her with possible actions to choose from.

In the R/3 System three categories of prompts are distinguished:

- prompts which the system displays every time a user action may cause a loss of data, so-called safety prompts,
- prompts which require the user to confirm or abort an action,
- prompts which require the user to choose between two alternatives.

Within these categories, you can use various forms of display and modes of operation, all of which are implemented in different function modules.

A distinction is made between destructive and non-destructive actions or response options. "Destructive actions or response options" result in a loss of data. If no data is lost, they are referred to as "non-destructive actions or response options".

Safety Prompts

The system displays safety prompts when the user executes one of the following functions:

- F15=Exit, F3=Back, F12=Cancel, F17=Other <object>, OK-Code=N
- another object class from the application area menu
- another action from the <*object*> menu

Display dialogue boxes with a safety prompt only if data entered may be lost. Use the system flag SY-DATAR to check whether this is the case.

Other System Prompts

The system requests the user, with or without messages, to confirm an action, for example:

- if entities - files, folders, tables, and so on -are to be deleted
- if entities are to be copied
- if a decision has to be made about how to continue processing

General Layout Guidelines

Dialogue boxes contain the following elements:

- a title bar
- an optional diagnosis text, explaining why the following question is asked
- a question which refers to the action previously performed, the possible loss of data, or the diagnosis text
- the "Yes" and "No" responses, displayed as two adjacent pushbuttons (exception see below); present the "Yes" response

first; the cursor is generally placed on the non-destructive option
- the "Cancel" pushbutton; position it to the right of the response buttons (exception see below)

For the most common prompts, there exist predefined dialogue boxes that are called via function modules (see below). Blank lines between the individual elements of the dialogue boxes are set automatically by these modules.

Safety Prompts after Returns

Display safety prompts when leaving the initiated action may cause a loss of data. There are two possibilities:

- Return with *Post/Save*: The user is able to save the entered data and then returns to the previous stage in the function.
- Return without *Post/Save*: The user is not able to save the entered data and only may return to the previous stage in the function.

With Post/Save

(function module POPUP_TO_CONFIRM_STEP or POPUP_TO_CONFIRM_WITH_VALUE)

Use this prompt as the standard return option, if possible. The user leaves the task function by activating the *Exit* function. The previously entered or changed data is either posted or saved when the user chooses this option in the upcoming dialogue box. Otherwise unsaved data will be lost. The user can also choose the "*Cancel*" option to close the dialogue box and cancel the initiated "*Exit*".

The question refers to the possible loss of data. Select the non-destructive option "Yes".

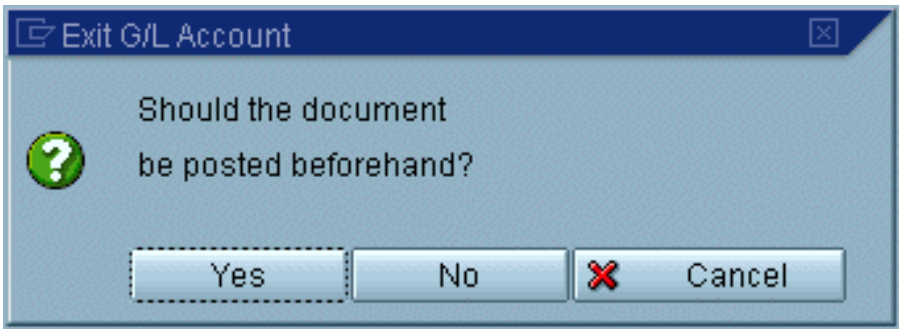


*Figure 1: Safety prompt for the function **Exit***

Show a similar dialogue box for the function *Back*. Note the different screen titles.

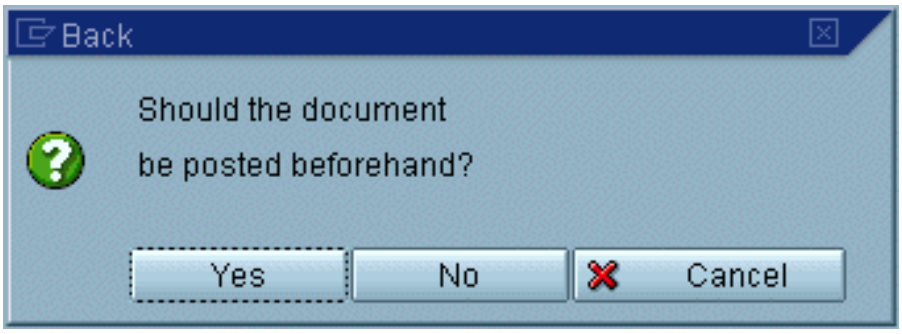


*Figure 2: Safety prompt for the function **Back***

Post/Save is Not Possible

(function module POPUP_TO_CONFIRM_LOSS_OF_DATA)

The second return option is used if data is inconsistent or if too costly checks need to be performed. This time the user activated the *Cancel* function in the primary window. He or she can respond to the safety prompt in one of two ways: Either leave the task function and lose the previously entered or changed data, or return to the task to proceed.

A diagnosis text is displayed. It always reads: "Unsaved data will be lost". The question refers to the initiating action. Select the "No" response option.

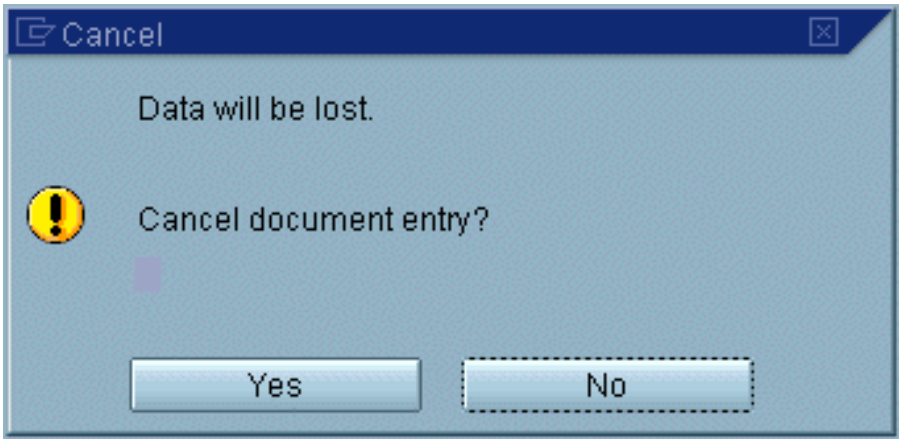


Figure 3: Safety prompt for the function ***Cancel***

Confirming Actions

If a dialogue box requires the user to confirm an action, there are two variants: The user is to confirm a non-destructive or a destructive action.

Non-Destructive Action

(function module POPUP_TO_CONFIRM_STEP or POPUP_TO_CONFIRM_WITH_VALUE)

In this case, check if you can do without a dialogue box because displaying a dialogue box will slow down processing.

The user has for example activated the "Change Currency" function. A dialogue box is displayed because the function has to be confirmed here.

The question refers to the initiating function. Select the "Yes" response option to speed up processing.

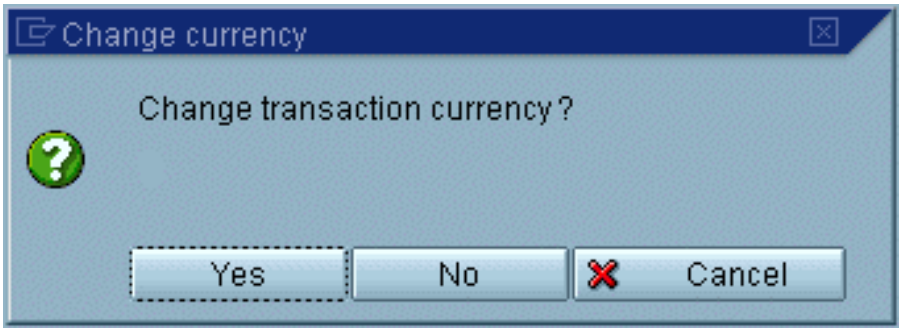


Figure 4: Example of a confirmation prompt for a non-destructive action

Destructive Action

(function module POPUP_TO_CONFIRM_STEP or POPUP_TO_CONFIRM_WITH_VALUE)

If the destructive operation may have irreversible consequences, always display a dialogue box. For example, the user has selected 20 lines in a list and then chooses "Delete". Display a dialogue box since the function has to be confirmed.

If a transaction offers several equally important operations which require the user to confirm the action, and these operations are of a destructive and non-destructive nature, select the "Yes" option as response for all operations. This explains why in extraordinary circumstances the "Yes" option is selected for a destructive action.

If, however, the destructive operation is a less frequently used function, then select the "No" response for maximum protection against a loss of data.

The question refers to the initiating function. In the example, the "No" response option is selected to protect from loss of data.

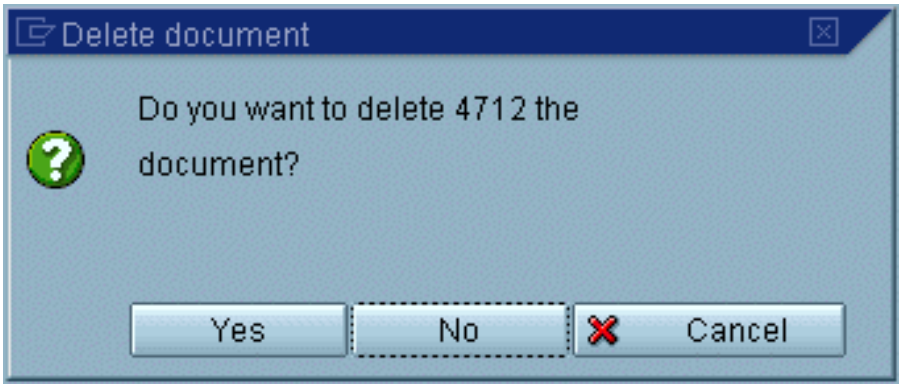


Figure 5: Example of a confirmation prompt for a destructive action

Confirming Actions after Previous Message

The options described in the previous section can be supplemented with a message, warning, diagnosis or information that precedes the actual question.

Message + Action

(function module POPUP_TO_CONFIRM_WITH_MESSAGE)

First, the system displays a dialogue box with a message specifying the critical processing state. Then the user is asked how he or she wants to proceed.

A message text (concluded with a full stop) is placed in front.

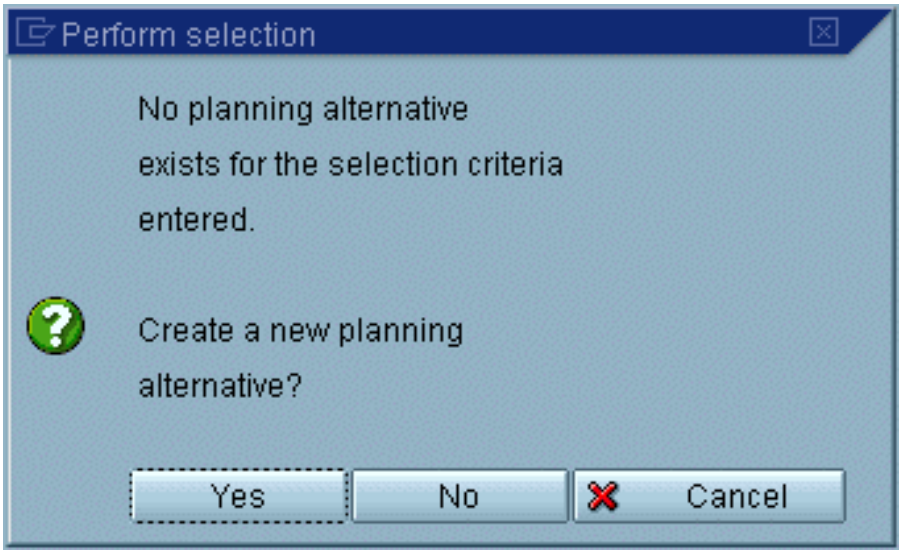


Figure 6: Example of a confirmation prompt with message

Choosing between two Related Options with or without Previous Message

The most general form of a dialogue box containing a prompt is the dialogue box which requires the user to choose between related options. It can be preceded by a message, warning, diagnosis, or information.

[Message] + Decision

(function module POPUP_TO_DECIDE and POPUP_TO_DECIDE_WITH_MESSAGE)

If the system asks the user an either/or-question which does not allow yes/no-answers, name the options offered to the user. Optionally, you can let the system display a message first. In the question, the two options are linked with "or".

Next to the question, separated by a blank line, present two corresponding response options and a canceling option in a horizontal button arrangement. As the following example shows, a vertical button arrangement is also possible.

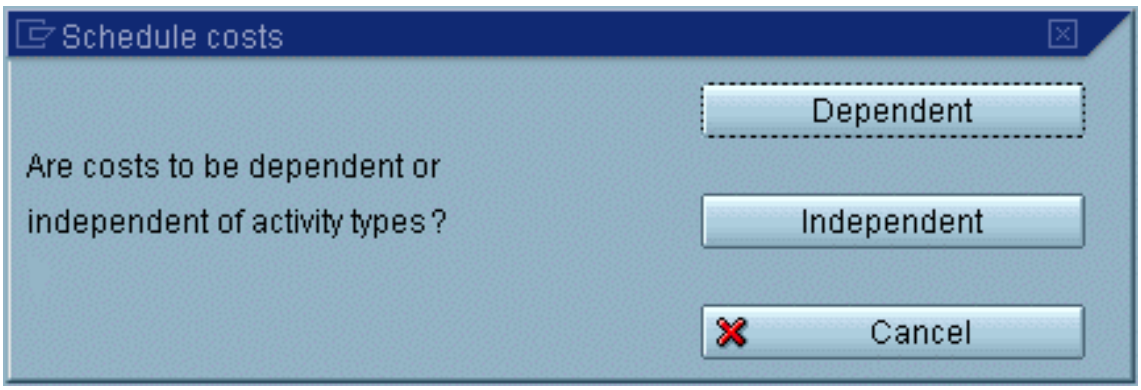


Figure 7: Confirmation prompt with alternatives (vertical arrangement, no message)

 top

Source: **SAP R/3 Style Guide**

 Dialogues

Buttons in Dialogue Windows

In the R/3 System there are only "modal" dialogue boxes. This means that once a dialogue box has been called, it only has to be closed again when you want to continue working in the primary window. At the moment it is not (yet) possible to switch back and forth between parallel windows during processing.

A dialogue box, however, can in itself consist of a number of elements like, for example, tabstrips, input fields, tables, and various functions. This standard regulates the layout of the application toolbar in instances when it is possible to make a large number of entries, or even to process a whole object in the dialogue box. This standard proposal includes function key settings from the standard proposal.

Layout alternatives are demonstrated using task situations. Simple standard cases are dialogue boxes which only display informations or which give the user the option to change settings.

If several fields and/or table entries in a dialogue box can be changed, it may be useful and/or necessary to check, confirm and apply this entries.

Source: SAP R/3 Style Guide

Buttons in Dialogue Windows

Simple Standard Cases

The enter key triggers that function which adopts the values or selections entered and closes the window (in the case of display only, the enter key simply closes the window).

Case 1: Dialogue Box with Change Option: Execute on "Enter", Cancel on the Last Key

- Standard function "execute" or "OK" on Enter (format: icon_okay without text)
- Alternative: special function on Enter (format: icon_okay + text, e.g. "find", "choose")
- Then possibly: functions which refer to the whole object in the window
- Last function: cancel on V12 (format: icon_cancel without text)

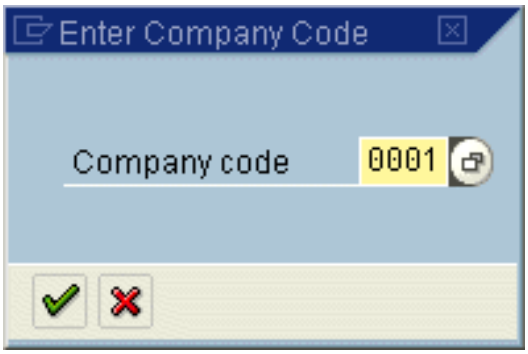


Figure 1: Dialogue box with change option

Case 2: Display only: Continue on "Enter", Cancel Isn't Displayed

The function "Continue" is on Enter (format: icon_okay without text, Quick info text: Continue).Cancel must be activated in the menu painter (V12), but is not shown in the application toolbar

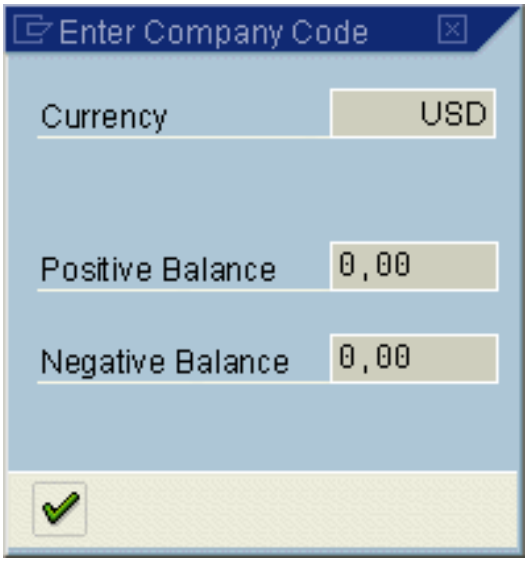


Figure 2: Dialogue box which only displays information

 top

Source: **SAP R/3 Style Guide**

Buttons in Dialogue Windows

Checking, Confirming, Closing and Applying the Dialogue Box

If several fields and/or table entries in a dialogue box can be changed, it may be useful to have entries checked when you hit the "Enter" key. In this case, the dialogue box remains open and must be closed explicitly using a separate function. The function "Check" can be performed on "Enter" (format: icon_okay + text "Check"). The dialogue box must then be closed with the function "Confirm" (case 1).

Conversely, the dialogue box can, of course, be closed with Enter, as before. An specific function, "Check", which has been added, then allows entries to be checked without closing the dialogue box (case 2).

Here are also cases where the entries in the dialogue box affect the underlying primary window (case 3). This case is similar to case 2, except that the function "Check" is called "Apply".

Case 1: Check on "Enter", Confirm on V8

In order to check the entries in the following dialogue box, the function "Check" is assigned to "Enter" (format: icon_okay + text "Check"). The dialogue box remains open. The function "Confirm" is on V8 (format: only text "Confirm"). This ensures that the entries are checked and the dialogue box closed. "Cancel" is the last function.

This type of arrangement is always useful when it is usual to make several entries which must first be checked to see if they are correct. This is especially useful if tables are also used in the dialogue box.

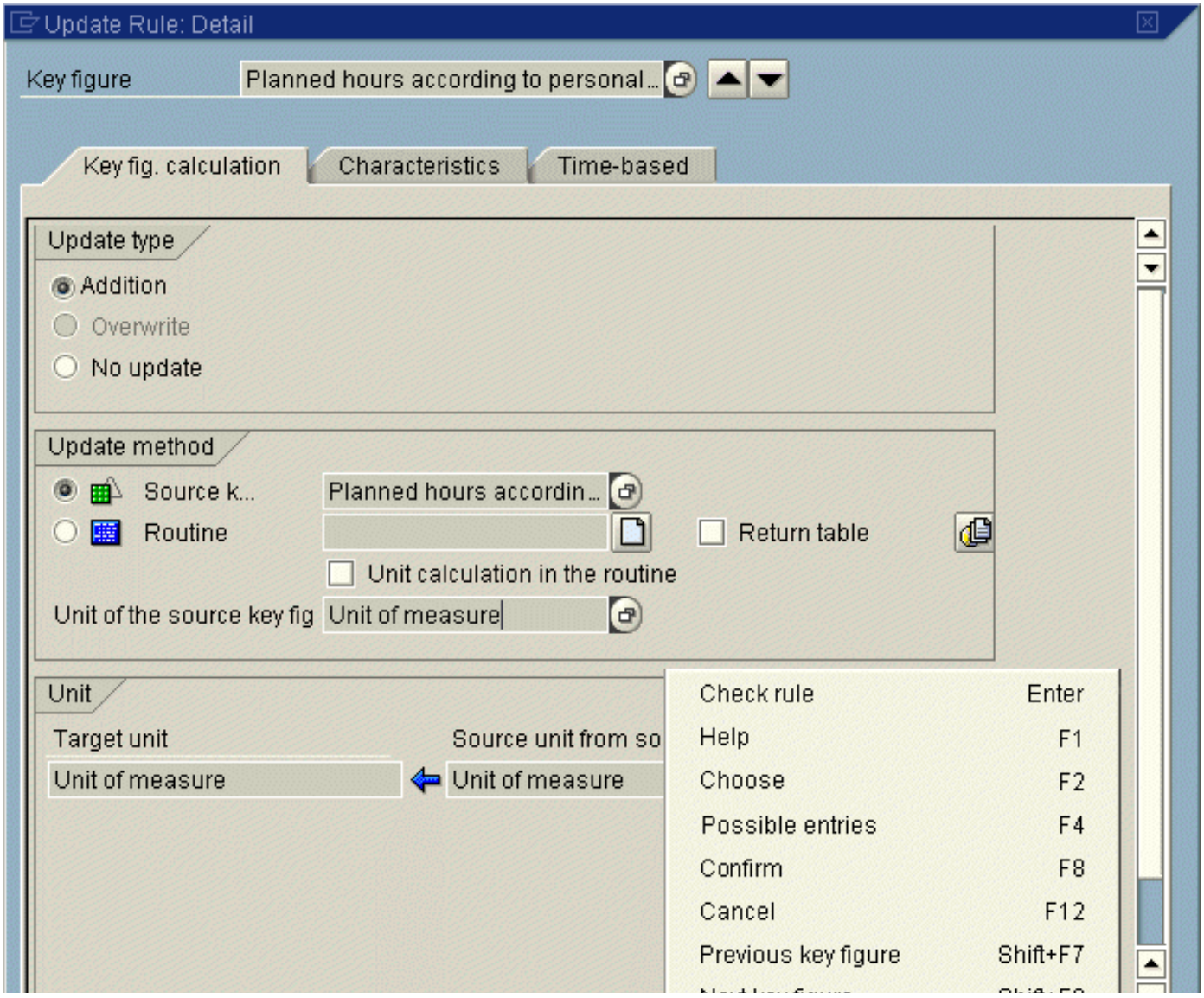


Figure 1: Dialogue box with the sequence of pushbuttons "Check" (Enter), "Confirm" (V8), <Other Functions>, "Cancel" (V12)

Case 2: Confirm on "Enter", Check on V7

In order to check the entries in the following dialogue box, the function "Check" is assigned V7 (format: only icon_check). The dialogue box remains open. The function "Confirm" is on the enter key (format: only icon_okay). This ensures that the entries are checked and the dialogue box closed. Cancel is the last function.

This type of arrangement is useful if it is usual to end the dialogue step immediately using "Enter" and when no errors are expected. The specific "Check" function on V7, however, allows the entries to be checked first, without closing the dialogue box.

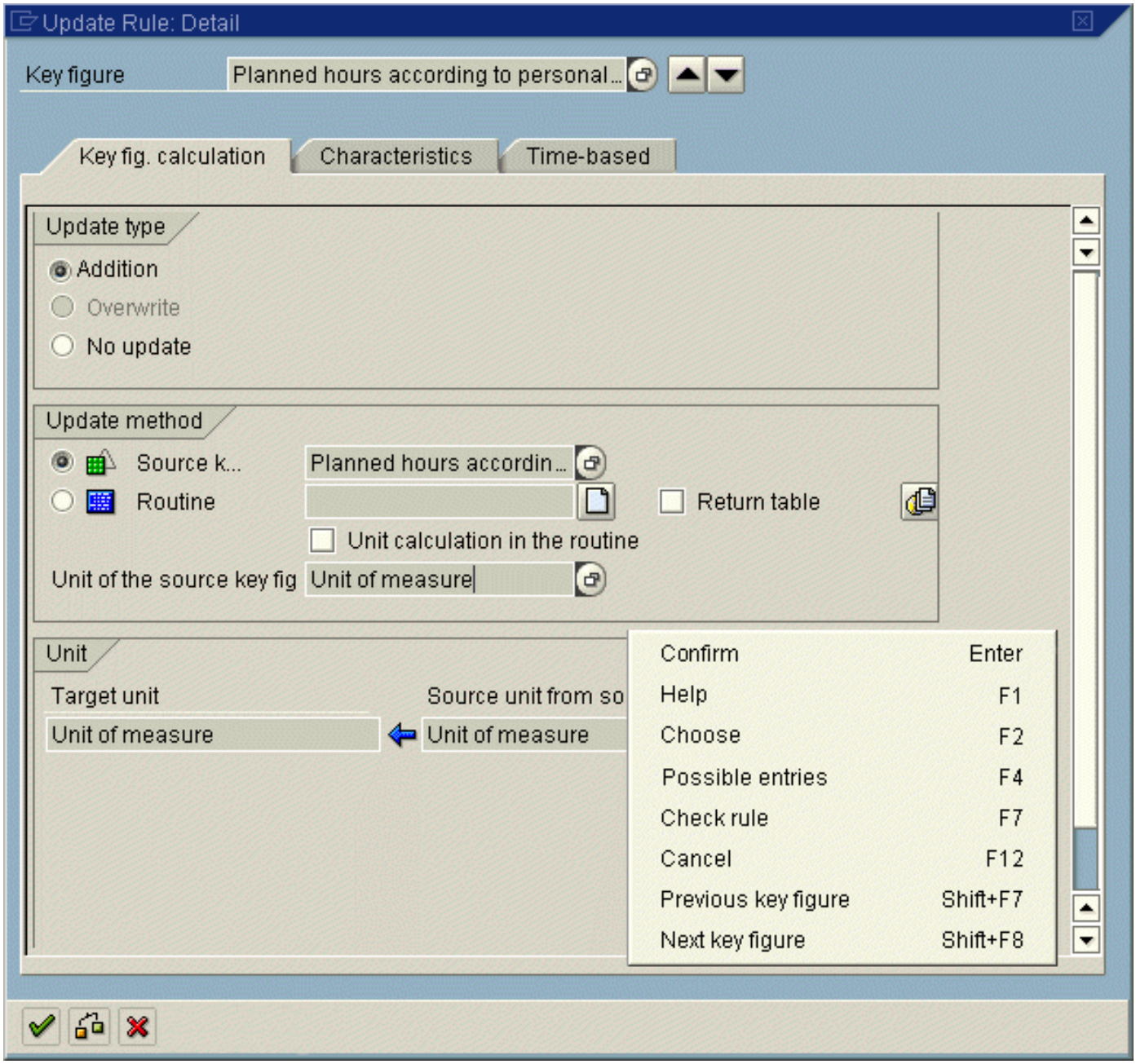


Figure 2: Dialogue box with the sequence of pushbuttons "Confirm" only with icon_okay (Enter), "Check" (V7), <...>, "Cancel" (V12)

Case 3: Confirm on "Enter", Apply on V7

This case differs from case 2 in that the function "Apply" (format: text only) causes a change in the underlying primary window. Thus settings will be made in the dialogue box which influence the primary window. "Confirm" on Enter (format: only icon_okay) adopts the settings and closes the dialogue box.

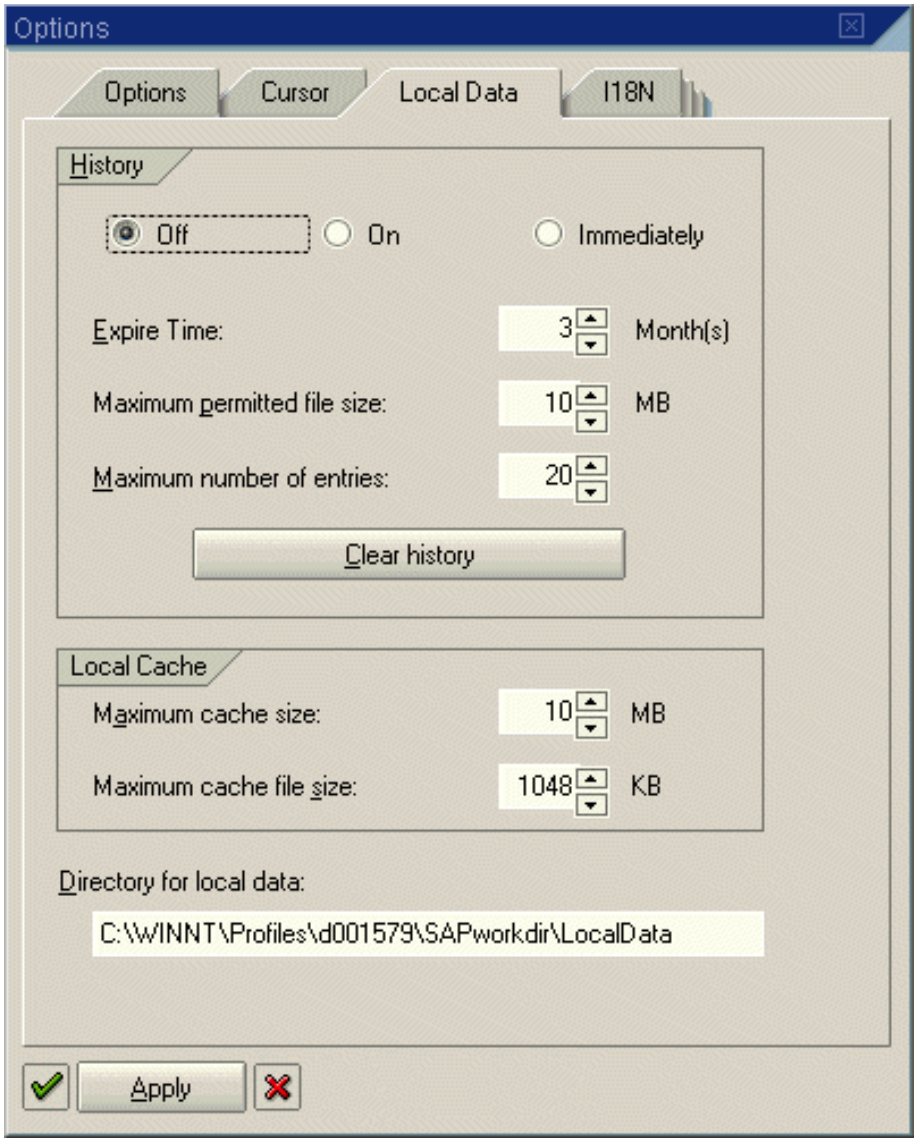


Figure 3: Dialogue box with the sequence of pushbuttons "Confirm" only with Icon ICON_okay (Enter), "Apply" (V7), <...>, "Cancel" (V12)

Exception: "Save" in the Dialogue Box

Normally, the whole object being processed would be saved for each transaction. To do this, the save function (ctrl-S), which is found in the standard toolbar, is used.

Alternatively, the "Save" function can be extended to "Save and Close". The text must then be moved to the relevant icon. If "Check" is omitted, this function would move to the enter key (with the icon icon_okay and text "Save and close").

Case 4: Check on "Enter", Close on V15 and Save on V11

In particular cases, the whole object can also be processed in a dialogue box. In this case, it is useful to be able to save the object directly. In this case the "Check" function is on "Enter" (format: icon_okay plus text), which means that you can continue

processing the object until it is correct. The "Close" function (or "Exit") is used to close the dialogue box. The icon "ICON_CLOSE" is also available for this purpose. "Close" and "Exit" are on V15. If the object has not yet been saved, a confirmation prompt will appear.

The "Save" function actively saves the processed object. A success message should appear. The dialogue box is not normally closed. It is then possible to process another object in the dialogue box.

"Cancel" is then, of course, displayed as the last key, allowing you to exit the dialogue step.

Note: This procedure undermines the strict old rule, "never use save in a dialogue box".

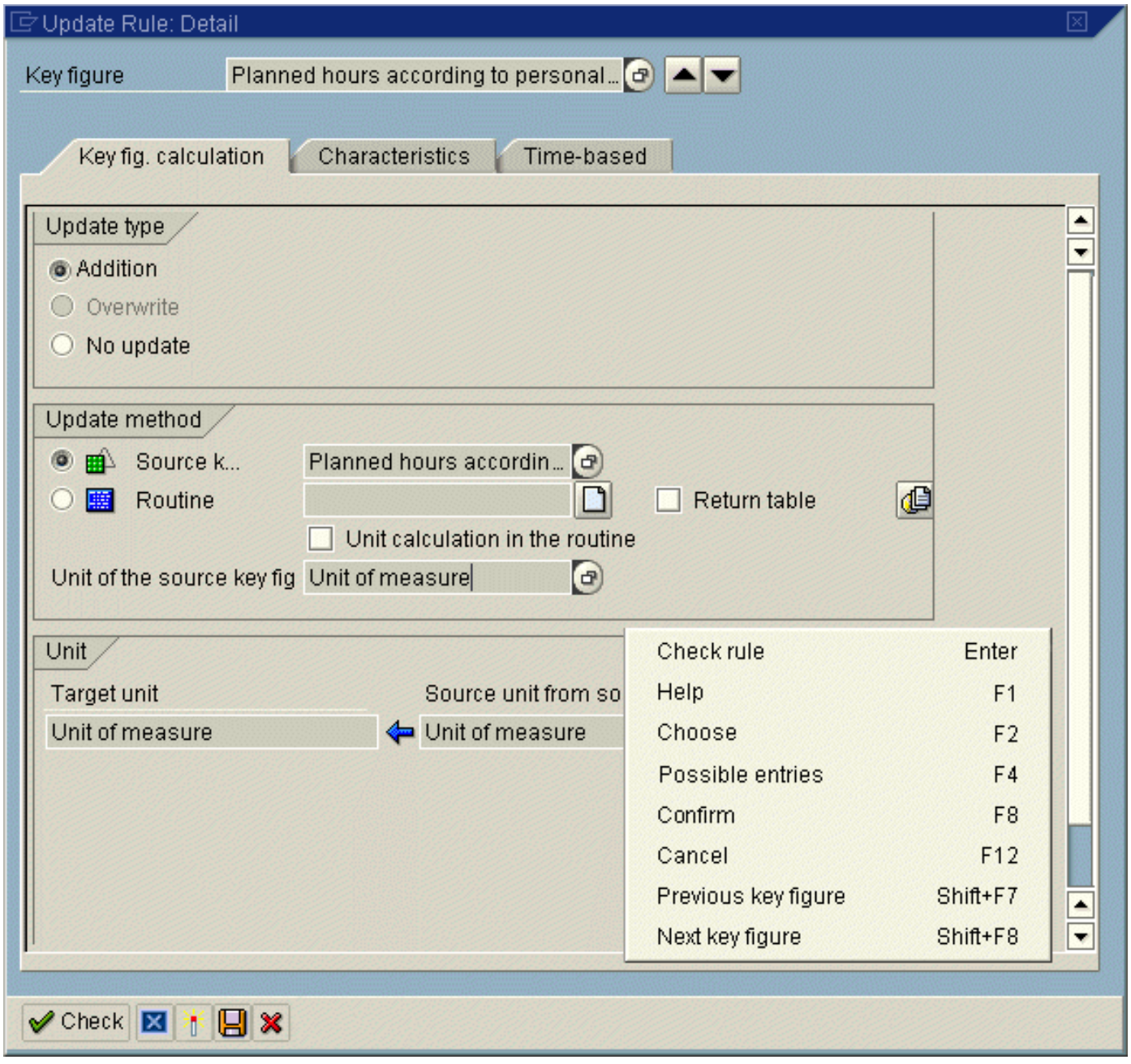


Figure 4: Dialogue box with the sequence of pushbuttons "Check" (Enter); "Close" (V15), <...>, "Save" (V11), <...Other Functions>, "Cancel" (V12)

top

Source: SAP R/3 Style Guide

Texts

Screen Titles | Messages | Message Short Texts | Capitalization

This section describes how to design texts. It deals with topics such as abbreviations, hyphenation, use of country-specific characters, and also highlighting. A further section discusses points to be borne in mind when translating the R/3 System into other languages.

Formulations

You have to provide the user with a work environment that is easy to understand: when choosing terms, when specifying pull-down options consisting of an object and an action or when creating the different messages. The following guidelines deal with the formulation of such texts.

Guidelines

Terms

If possible, use terms that the users know and that they understand.

Suffixes

Except for the word "personnel number", you can do without the suffix "number". For example, we talk about a "vendor" and not a "vendor number". Use suffixes such as "key" and "indicator" only if they are necessary to understand the field.

Texts in the Form of Questions

Texts containing questions should be formulated positive.

Good: Do you want to save the data?
Bad: Don't you want to the save data?

Shortened Sentences

If you use short formulations, use the sequence verb -> noun (English) or noun -> verb (German) without the article. Avoid making verbs into nouns.

Exception: If several short phrases appear one after the other (for example, in a pull-down menu), the part containing the information, that is the verb, is always written first. This also applies to German.

Abbreviations

Space restrictions may necessitate the use of abbreviations in many situations: in dialogue boxes, for field names in the applications, for lines in the menu bar or pull-down menus, and for function key names. Use Webster's Ninth New Collegiate Dictionary or for German the Duden as the reference for abbreviating words.

Do not use abbreviations if sufficient space is available or if the user rarely accesses a particular screen.

Guidelines

Using Accepted Abbreviations

If there is already an accepted abbreviation for a word, use it. Use abbreviations only if you are sure that the user can easily understand them. In general, do not use abbreviations that consist of only one letter.

Good: Dr.; Su.; min.; DM
Bad: Dctr; Snday; minmum; Deu. Mark

Creating New Abbreviations

If there is no accepted abbreviation available, words can be abbreviated according to the following guidelines arranged in order of priority.

1. **Omit the end of the word**

   If possible, create abbreviations for individual words by omitting the end of the word. Menu options and function key names should have a period at the end of the word.

   In a table column heading, also use the period, if possible.

   "abbreviation" becomes "abb."

2. **Omit vowels**

   Omit vowels only if you cannot form a meaningful word fragment. Vowels that do not influence the pronunciation of the word very much can be left out.

   "group" becomes "grp."

3. **Create an acronym**

   Create acronyms (artificially created new terms consisting of the first letters of the word fragments or individual words) only if this is clear from the application field (for example, GR for goods receipt in purchasing) or if this is required for table column headings.

   "Data Processing" becomes "DP", "Human Resources" becomes "HR", "General Ledger" becomes "GL"

   Never use a closing period if the last letter of a word is contained in the abbreviation.

   For example, "screen" becomes "scrn". The word fragment "number" is generally no longer used in the field name. An exception to this rule is "Personnel number" which can be abbreviated to "Personnel no.".

Abbreviating A Two-Word Structure

The guidelines for creating a new abbreviation apply here as well.

The individual word fragments are not separated by a period. Each new word fragment always begins with a capital letter if the

previous word fragment has been abbreviated.

"Start date" becomes "StrtDate", "Document type" becomes "DocType", "Tolerance key" becomes "TolKey", "Object group number" becomes "ObjctGrp."

Abbreviating Several Words

Proceed as follows when abbreviating several words:

1. If the previous word was abbreviated according to the guideline "Omitting The End of the Word", it should end with a period and be followed by a blank, if there is enough space.

   "Fixed vendor" becomes "Fix. vendor" or even "Fix.vendor",
   "Fixed capacity requirement" becomes "Fix. CapReq." or even "Fix.CapRep."

2. If the last letter of the word is used, insert a blank between the words.

   "Create new charge" becomes "Crte new chrg.", "Fixed vendor" becomes "Fixed vend."

3. If the second or a following word is written in lowercase, use the lowercase.

   "Fixed date" becomes "Fix. date"

Choosing From Several Words

If you look at the short descriptions in the ABAP Dictionary, you can also find complete phrases, such as "Fixed machine-related capacity requirements in hours". For the abbreviation, only choose the significant words. You may need to change the word order to form a suitable abbreviation. The guidelines 1. to 3. apply for the creation of abbreviations.

"Fixed machine-related capacity requirements in hours" becomes "Mach.fix" or "Mach. fix"

Compounds

In the case of different abbreviations for words which consist of more than two word fragments, make sure that individual word fragments are not completely omitted. Otherwise, it could result in the impression that two different things are meant. For example, "Minimum order amount" could be abbreviated to "MinAmount" and also to "MinOrdAmount". Here, it is better to use either "MinOAmt" or "MinOrdAmount", so that the word fragment "order" is not forgotten.

Further Text Issues

Hyphenation

Do not hyphenate words on a screen, so that they extend over two lines. Write words with a hyphen in the same line.

Good: Enter all your relevant data into this dialogue box.
The new object-oriented
user interface is fairly good.

Bad: Enter all your relevant data into this dia-
log box.
The new object-
oriented user interface is fairly good.

Exception: With long sections of continuous text (more than 50 characters), words can be hyphenated if this would otherwise result in large "gaps" on the right border.

Uppercase/Lowercase

Use lower and uppercase letters. Do not use uppercase letters as design elements to highlight headers or other elements.
Exceptions: Use uppercase letters for report names, transaction codes, names of tables and files.

Good: Software ergonomics is an interdisciplinary science.
Bad: SOFTWARE ERGONOMICS IS AN INTERDISCIPLINARY SCIENCE.

Country-Specific Characters

Use country-specific characters, for example, the umlaut and ß in German.

Good: Möhren ärgern Spaßvögel übrigens nie.
Bad: Moehren aergern Spassvoegel uebrigens nie.

Highlighting

You can achieve better legibility in the work area of a screen by highlighting important or structuring elements.

Highlighting Types

From the ergonomic point of view, the possible types of highlighting can be divided into the following three groups:

1. Tolerable highlighting: use it very sparingly and only in the case of an emergency (see below)
   - Lines as single lines, interrupted lines and double lines
   - Intensity
2. Hardly tolerable highlighting: avoid it, if possible
   - Semi-graphic symbols such as * ! #
3. Ergonomically bad highlighting: do not use it on a screen
   - Uppercase, spaced text, flashing, color

Highlighting Usage

The highlighting in the first two groups is to be used only if all means of spatial and conceptual structuring have been exhausted.

As a rule, not more than 10 % of the screen should be extra emphasized.

Translation

This section groups together some of the most important precautions that need to be taken to ensure that the R/3 System can be translated into other languages.

Basic Rule

Translation of texts from English or German into other languages may increase the length of the text by 30 percent or more. If you do not leave enough room on the screen, translators have to abbreviate terms beyond recognition.

As a rule of thumb, you should allow space for text expansion by about one quarter to one third during the translation.

User Interface Elements

Field Names

For field names, use the standard lengths provided by the system so that sufficient space is available for the translation. Sometimes, you can use the second largest length, but leave at least two blanks.

Short Texts

The same applies to short texts as to field names. However, due to the greater length, the translators have more room to find an appropriate word for the available space.

Menu Texts

For menu texts, the system standard is 20 characters. If the texts in the menu bar are very long and the entries no longer fit on one line of the menu bar, the menu bar is split into two lines. This causes problems with small monitors, as a screen line may get lost.

With pull-down menu texts, there are no length problems except that the limitation of 20 characters may cause a problem.

Pushbutton Texts

Pushbutton texts, like to menu texts, are restricted to a maximum of 20 characters. For pushbuttons, however, you have to keep in mind some special features, although these do not affect the translation. The system displays as many pushbuttons as can be accommodated in the window depending on the text lengths of the individual pushbuttons. Thus, if you want to make sure that a certain number of pushbuttons is displayed in the standard size window, you have to abbreviate the texts, if necessary. On the other hand, the number of pushbuttons increases when the window is enlarged, provided a sufficient number of pushbuttons is defined; correspondingly, if the window is reduced, fewer pushbuttons are displayed.

If you want to include an icon into the pushbutton, keep in mind that this will cost you another 4 characters! That means that you may have to abbreviate the pushbutton text. Do not drop the three punctuation marks that indicate a dialogue box, abbreviate instead.

Function Texts

During the enhancement of a user interface, problems may occur when replacing a function text. If this change is used to eliminate a guideline violation and if it does not change the nature of the function, it is useful. Otherwise, please always create a new function with a new function code! This is necessary, because the foreign language version does not necessarily have the same status as the current German or English version. This may lead to unwanted functions being called!

 top

Source: **SAP R/3 Style Guide**

 Texts

Screen Titles

Screen titles are an important aid for users to find their ways through the R/3 System. The title of a window is a unique label to identify the window. It is located in the title bar, the topmost line of a primary window or dialogue box.

The title bar helps the user to identify the current screen, that is, answer the question "Where am I?". Mostly, the name of the action the user has chosen and which is currently being executed is displayed in the title bar.

In some situations, it is necessary to make reference to the preceding actions. The title should then answer the question: "Where do I come from?" or "In which higher-level action am I currently?". This way, titles contribute to an efficient navigation through the R/3 System.

Guidelines for the Individual Levels

When accessing a particular task in the R/3 System, the user has to navigate through the system hierarchy. To make this hierarchy transparent, it is especially important that you enable the user to identify the individual screens and system levels. You can do so by creating the appropriate screen titles.

The following text gives the screen titles (if only one screen exists for a particular level) and/or design guidelines and examples (if several screens exist for a level) you can use to identify the different levels of the system hierarchy.

Syntax Conventions: Variable terms which must be replaced by a full identification appear in angle brackets. Expressions in square brackets are optional. Examples are in italics.

Main Menu Level

Title: SAP R/3

Application Level

If the user gets to this screen by choosing options from several cascading pull-down menus, the title of the screen displays the last chosen menu option (current menu option).

Syntax: <current menu option>
Title: *Accounts Payable*

To avoid ambiguities, you can also create the title of the screen from the concatenation of the higher-level menu option and the last chosen menu option.

Syntax: <higher-level menu option> - <current menu option>
Title: *Customizing - Extended General Ledger G/L*

Task Level

The title bar used to identify a task consists of three components:

1. the name of the higher-level task or object class (ObjCl, Object Class),
2. the chosen action which refers to the chosen task or the object class as a whole (Action),
3. optionally, the description of the current screen (Screen) and - in some cases - an object separated by three blanks (if identifiable), consisting of the field name and value (Object)

In an editor, this object would be the file name; for a posting document, for example, "document number 25".

If the current screen of the previous system status is initiated by choosing a menu option or activating a pushbutton, the <Screen> can also be created directly from the name of the menu option or the pushbutton text. You can also specify the path which brought the user to the current screen (in the form <Screen-2>-<Screen-1>-<Screen>).

Syntax: <Action> <Object Class>[: <Screen>[<Object>]]
or <Action> <Object Class>: [<Screen-2>-][<Screen-1->]<Screen>
Title: *Create unit costing: Item list*
Display FiAc document: Initial posting document number 1674

When naming reports, you must be aware that the user generally calls the report by choosing a corresponding menu option. The name of the menu option should match the title of the report, if possible. They should at least contain the same word elements.

Syntax: <Object Class>: <Screen>
Title: *Document compact journal: Result*

Note: The title bar of a report and the possible list headers that are located within the work area above the report name are different things!

Titles for General SAP Screens

The title bar at the task level generally consists of the following components:

<Action> <Object Class> [: <Screen>[<Object>]] or
<Action> <Object Class>: [<Screen-2>][<Screen-1>]<Screen>

Avoid redundancy and text components that do not provide further information. Thus, try to do without "screen" and "data" if the title is otherwise clear enough.

To use consistent names for the component <Screen> at task level, specific names have been assigned to general SAP screens.

Initial Screen

Syntax: <Screen> = "Initial screen"
Title: *Enter document: Initial screen*
Create material: Initial screen

Data Screen

Syntax: <Screen> = <specific contents of the screen>
Title: *Change purchase order: Header*
Change material: Purchasing
<Screen> = "General data", if the content is not specific
Title: *Change material: General data*

Entry Screen, Single/Collective Entry Screen

Syntax:	<Screen> = "Single entry"	<Screen> = "Collective entry"
Title:	*Create goods receipt: Single entry*	*Create goods receipt: Collective entry*

Entry vs. Edit

If the functions *Create* and *Change* occur in two separate tasks and these tasks have both a single entry and a multiple entry screen, you can formulate application-specific title bars.

Title:	*Create <object>: Single entry*	*Change <object>: Single change*
	Create <object>: Collective entry	*Change <object>: Collective change*

Generally, you distinguish between the single entry and multiple entry method. In this case, the entry method is specified in the task name and not in the screen name.

Title:	*<object> Single entry: Details*	*<object> Single change: Details*
	<object> Collective entry: Overview	*<object> Collective change: Overview*

If the entry and the change method are grouped together in a task (that is, there is only one editing method) and the emphasis is on the change option, change the task name correspondingly to "processing".

Title: *<object> Single processing: Details*
<object> Collective processing: Overview

Additional Data Screen

Syntax:	<Screen> = <menu option calling the screen or dialogue box>	
Title:	Examples of dialogue boxes:	Example of primary windows:
	Details: Short text	*Create material: Administration data*
		or shorter: *Administration data*

Detail Screen

Syntax: <Screen> = "Details"
Title: *Display table: Details*

Overview Screen

Syntax: <Screen> = "Overview"
Title: *Goods receipt for the purchase order: Overview*

Special case: Overviews can display several levels of detail. Specify these levels by an additional identifier. Do not replace the term "Overview" with another word.

Title: *...: Item overview*
...: Overview open items
...: Overview selected items

Result Screen

Syntax: <Screen> = Result
Title: *ABC analysis: Result*

Selection Screen

Syntax: <Screen> = "Selection"
Title: *ABC analysis: Selection*

Title Bar in Dialogue Boxes

Dialogue boxes can be displayed in different ways. The contents of the title bar varies correspondingly.

After Pressing F4 (Possible Entries)

The help system generates these titles automatically and enters them into the dialogue box. If you program the F4 functionality, you have to set the title conforming to the following rules.

The literal "PossEntr" indicates the function F4. The field name (FldID) names the field for which F4 help was requested. If an additional dialogue box is initiated from the first one (also to be programmed by you), copy the value selected in the first window to the title bar (Sel) and separate it by three blanks.

Syntax: PossEntr: <FldID>[<Sel>]
Title: *PossEntr: Process*
For an additional dialogue box depending on the first one:
PossEntr: Process balance sheet adjustments

After Choosing a Menu Option from the Pull-Down Menu

The options chosen from the menu bar (optional) and the pull-down menu are listed. Place the entry chosen from the menu bar (Menu) at the beginning (to increase understanding). In addition to the first pull-down menu (1. PM), display the second pull-down menu (2. PM) and the third pull-down menu (3. PM), if available, to indicate the chosen path.

Syntax: [<Menu>:]<1. PM>[-<2. PM>][-<3. PM>]
Title: *Insert*
Copy line
Goto: Graphic

If the title bar does not describe the window adequately and/or the title is too long, even though you have followed above mentioned design guidelines, you can define the title by specifying an optional action (Action) and the affected object (Object):

Syntax: [<Action>] <Object>
Title: *Specify date format*

If a Second Dialogue Box is Called from a Dialogue Box

The structure is similar to the one described in the previous section. In addition, display a freely-selectable one-word title (Title) for the second dialogue box. Alternatively, display a value selected from a list in the first dialogue box (Selection).

Syntax: <1. PM>[-<2. PM>][-<3. PM>]: <Title>
<1. PM>[-<2. PM>][-<3. PM>]: <Selection>
Title: *Copy line: Parameter*

Copy: Letter1

If a [<Action>]<Object> title was defined in the first dialogue box (see "After choosing a menu option of the pull-down menu"), the title can appear before the current one-word title (Title) with a follow-up dialogue box:

Syntax: [<Action>]<Object>]:] <Title>
Title: *Assign bank transactions: Bank charges*

Title Bar for Dialogue Boxes Displaying Messages

In certain situations, the program can display dialogue boxes which contain error messages, warnings, information as well as confirmation prompts for the user.

The title bar is to refer to the step which the user has last executed and which is responsible for the message.

The action (Action) last executed is to be specified optionally:

Syntax: [<Action>] <EdObj>
Title: *Exit transfers*
Delete line

The titles for some functions which cause returns and initiate a dialogue box are to be specified as follows:

Function	**Title**
Exit	*Exit <entire object>*
Cancel	*Cancel <object component>*
Back	*Back*
Other <object>	*Other <object>*
Create	*Create <object>*
Change	*Change <object>*
Display	*Display <object>*

General Notes on Screen Titles

Choose the freely-selectable terms accordingly: Describe the options from the user's point of view and not technical matters.

Keep to the maximum possible length of a title bar (60 characters). You may abbreviate the individual text components. Please note that the translation into another language may require more characters. Take blanks and the punctuation marks ":" , "-", "(" and ")" into account.

Note for translators: In some languages, the <Action> <Object Class> order may have to be reversed.

top

Source: SAP R/3 Style Guide

 Texts

Messages

Messages appear in dialogue boxes and in the status bar. They inform users on different aspects: error conditions, warnings, success, infos. Therefore, it is important that the users understand these messages.

General Messages

If the system detects an error, it displays a message either in the status bar or in a dialogue box, depending on the type of error. The message is to state the problem and to offer a solution so that the user can proceed without outside assistance.

Guidelines

Be as precise as possible! Messages should describe the situation which has occurred as unambiguously as possible. Be specific and address the problem in the user's terms. Avoid error messages which contain technical descriptions or explanations.

Offer the user direct support for the error recovery. Do not just state the problem, but rather indicate what needs to be done. If this is not possible, use a positive tone to explain the problem as precise as possible. You can provide additional support in parentheses, if known. Have a courteous and positive tone, rather than being imperious or condemning the user. Avoid the use of "may", "must", "should", "belong", "wrong", "insufficient", "incomplete", "not allowed", and so on.

Use a clear and simple sentence structure and only write complete phrases. Articles can be omitted; a verb (at least "is" or "are"), however, should always be present. Use present tense, if possible. Be consistent in your use of language; do not mix different languages.

Some more Don'ts...

Do not conclude the message with a period or with an exclamation mark. Suggestions for the error correction can be given in parentheses. Always leave a space after a period and never before. Do not use special characters such as < >, as they will not necessarily be known to the user; value specifications such as % can be used, however. Do not introduce or conclude messages with asterisks (***). Never put field names, report names and table names in quotation marks.

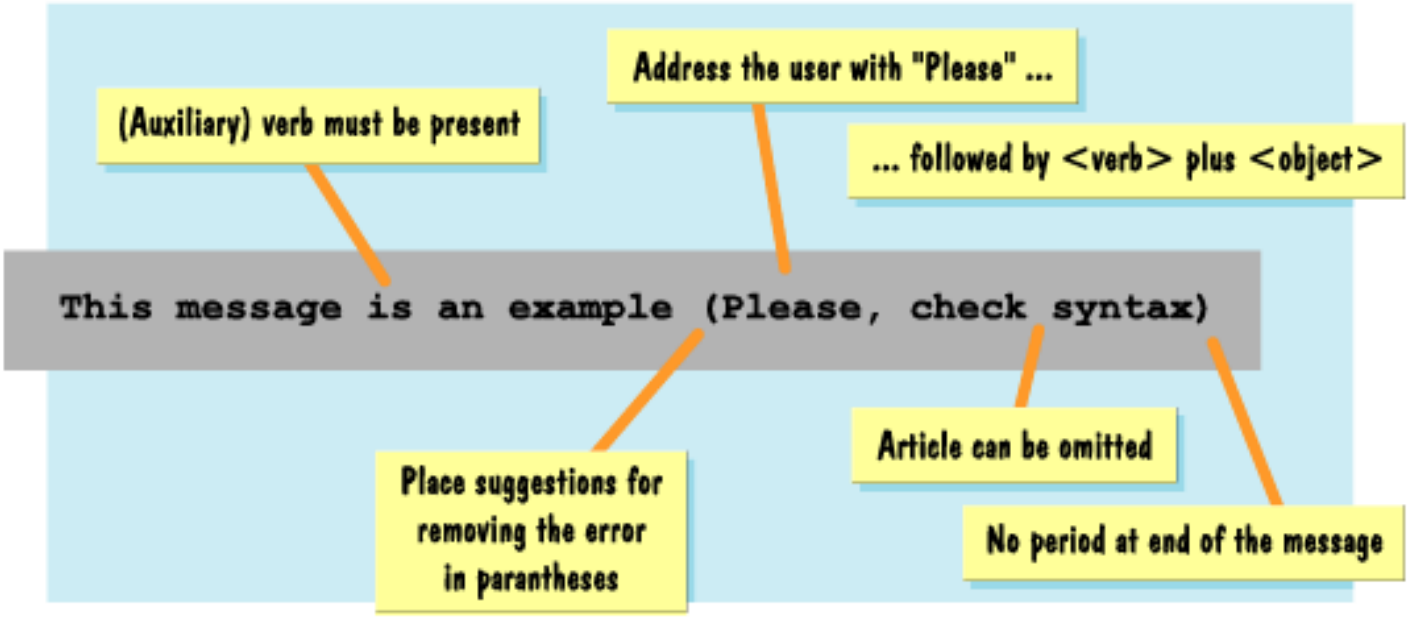


Figure 1: Structure of a message

Examples of Good and Bad Error Messages

The following topics list examples of bad and reformulated error messages (without comments) to illustrate the above-mentioned guidelines.

Messages Which can be Formulated More Specifically

Bad: Key is not in table
Good: Distribution channel is not expected (Please check entry)

Bad: The cursor is not placed on a line which can be selected
Good: For the selection, please position the cursor on a line of the table

Bad: Release order specifications incomplete
Good: Please complete release order specifications by specifying order type and distribution channel

Messages to Be Formulated Less Ambiguously

Bad: The line is not in table 134
Good: The combination of desired lines is not expected

It would be better to explain this message in detail in a dialogue box or to name the reference to the individual entry fields in the long text.

Bad: Entries incomplete
Good: Please complete entries

Bad: No change
Good: Entries were not copied, as they already exist

Messages Prompting the User

Bad: Deletion flag incorrect, only 'J', 'S', ' ' allowed
Good: Please enter 'J', 'S' or ' ' as deletion flags

Bad: Postal code not entered or has incorrect length
Good: Please enter postal code in defined length

Bad: Percentage rate must be smaller than 100%
Good: Please enter percentage rate between 0 and 100

Bad: No items selected
Good: Please select items in the list first

Bad: Release date entered incorrectly
Good: Please enter the release date in the form MM.DD.YYYY

Message Types and Error Groups

The messages in the R/3 System can be classified according to different types. The message type determines where the message is issued and how the system responds.

Messages which refer to errors, so-called error messages, can be grouped in terms of their contents on the basis of the events triggering them or the objects affected. Error messages are to state the problem and, if space permits, should offer suggestions for removing the error.

Message Types

S-Message (Success)

S-type messages are displayed in the status bar on the same or next screen. The message has no influence on the user's work. It only informs about the successful execution of system functions.

I-Message (Information)

This message has a modal character, that is, the user must acknowledge the message. Therefore, display I-messages in a modal dialogue box. The user can continue the processing by pressing the ENTER key.

W-Message (Warning)

So-called W-messages interrupt the processing and allow the user to make corrections. For this reason, fields are ready for input. Display W-messages in the status bar if the messages have been issued by a primary window. If a dialogue box issues the message, display the message in a separate dialogue box.

With regard to the system response, warnings can be compared with E-messages. E-messages, however, require the user to change the entry.

E-Message (Error)

When the system detects an error, use E-messages. Incorrectly completed fields must be ready for input. Do not only make the incorrectly filled field ready for input but also those fields whose entries have contributed to the error.

Depending on whether the E-message was issued by a primary window or a dialogue box, display it either in the status bar of the primary window or in a separate dialogue box.

A-Message (Abend)

A-messages are also of a modal character and are displayed in modal dialogue boxes.

A-messages do not allow the user to make any further entries. The user can only confirm the message. The task is abruptly terminated and the system returns to a higher-level menu.

Issue A-messages only in extraordinary circumstances, for example, when a system-related error occurs or if the error can no longer be controlled by the task.

Error Groups

The following error groups form a classification of possible errors while working with R/3. They are intended to aid you in designing messages for possible error conditions.

Errors in Tables

Generally, the user is not authorized to make changes in a table. Therefore, the short text of an error message which refers to tables should not contain a reference to a table. This information is contained in the long text, together with the possible hint to contact the system administrator.

If the display of the message was initiated because a system table does not include an entry which was expected to be listed, the short text should not contain a reference to the table but at most a note on the entry:

E-Message: Order type is not expected (Please check entry)

If the error is related to tables, which are only of importance for the SAP System itself or if the cause for the error cannot explicitly be assigned to a particular entry, the message should refer to the system administrator:

A-Message: No further processing possible (Please contact your system administrator)

The long text should always contain a detailed error diagnosis which can also be of a more technical nature (which, however, is marked correspondingly).

If special reference is made to lines which are not contained in a table and if the table is important for the user, that is, if he has maintenance authorization for the table, use the following message pattern:

A-Message: Please enter country in country table first

Input Values Aren't within the Valid Value Range

The user has entered an entry which was not within the valid, usually numeric value range.

E-Message: Please enter posting period in allowed value range

If the value range is known or predefined, the error message should ask the user to enter a value within a specified range:

E-Message: Please enter a posting period between 1 and 12

Meaningless Entries

In some situations, the user attempts to carry out an action which is meaningless from a business point of view (the system should be able to prevent these actions).

E-Message: Please create bill of material first

E-Message: Order type of outline agreement cannot be selected for &

System Error, System Limits Are Exceeded

If a non-user error occurs in the system, the message should explain that a system-related problem has caused the error. It should also point out how the user can respond. Technical descriptions should be avoided.

For temporary problems, the following "good" example can be used:

Bad I-Message: Global lock table is full. Locking is currently not possible
Good I-Message: At present, the function cannot be carried out

Give a detailed explanation of this message in the long text.

If a more severe system error interrupts the user's interaction with the system, you can display the following message:

A-Message: No further processing is possible (Please contact your system administrator)

Again, give detailed explanation of this message in the long text.

No Entry Made in a Certain Field

If particular entries are required to continue processing, the required specifications are, however, missing or incorrect:

E-Message: Please enter the batch number for material

E-Message: Please enter required-entry field material

Control Error

If an operating error occurs, for example, the user chooses a function which requires a selection beforehand, the message should specify the initiated action and the error diagnosis, if possible:

W-Message: You have chosen Copy (Please select an order first)

W-Message: You have chosen Delete (Please delete all items individually)

Collisions Due to Certain Conditions Set in the System

If records are locked, the user is missing the required authorization, etc.:

I-Message: You are not authorized for printer & (printer destination)

I-Message: Vendor master record & is currently locked

Invalid Result After Executing an Action

If a calculation would lead to a meaningless or invalid result, do not only give the diagnosis but also a note on how to eliminate the error.

E-Message: The specified quantity is smaller than the delivered quantity (Please enter again)

Success Messages

We talk about a success message if the system has executed a function and the user receives a confirmation message on the activity (for example, due to its importance).

S-Message: New record with field was added to list X

S-Message: Physical inventory document with the generated number has been posted

Plausibility Error

To avoid contents-related errors early on, the system checks entries for their plausibility, if possible.

W-Message: Please check pricing

 top

Source: SAP R/3 Style Guide

 Texts

Message Short Texts

Standards and Guidelines*

- Use precise and unequivocal language. Write concisely to allow for a potentially longer translation. Provide a maximum of information in a minimum of space.

Use	Do Not Use
Enter a valid unit of measurement	Entry not permitted
	Unallowed entry

- Use sentence style. The first word of the message should begin with a capital letter.
- Do not use an article at the start of the sentence.

Use	Do Not Use
Purchase order unit must not be deleted	The purchase order unit must not be deleted

- In longer message texts, omit articles within the text unless this creates ambiguity.

Use	Do Not Use
Unable to display status; period is already completed	Enter a period that ends in future

- Always include a verb.

  Exception: Error messages (no verb required) and search result messages (no auxiliary verb required)

Use	Do Not Use
Material is not available	Material not available

- Use the present tense wherever possible.

Use	Do Not Use
Unable to determine previous period	Previous period could not be determined *[past tense inappropriate here]*

- Exception: Success or failure messages should reflect the actual status.

Use
Master data saved

- Do not use punctuation within a message, unless the message contains a statement followed by a *reason for the error* or *recommended user action*. In that case, separate the two with a **semicolon**, and observe standard capitalization rules after the semicolon.

Use	Do Not Use
Data is not available; define the result areas first	No result areas are defined

- *Questions:* In that case, separate the two with a **period**.

Use	Do Not Use
Unsaved data will be lost. Do you want to save?	Data will be lost! Save?

- Do not use special characters such as angle brackets (< >), dollar signs ($), quotes ("), or asterisks (*).
- Do not use ALL CAPS or s p a c e d t y p e for emphasis (also applies to variables).
- Avoid abbreviations.
- Avoid using formulations that cause problems in translation, especially in texts containing variables.

Use	Do Not Use
Entry 0002 of the entries to be copied...	The 0002nd entry to be copied...

- Use vocabulary that the user will understand. Focus on the task at hand, not the technical function performed by the system. Write from the point of view of the user, not the system.

Use	Do Not Use
Enter the company code	The BUKRS field is not complete

- Avoid using technical terms.
- Use direct speech wherever possible — that is, address the user directly. Avoid using the passive voice.
- Do not use "please" when asking the user to do something.
- If possible, and space permitting, provide assistance; do not simply diagnose the error. Formulate the text in a positive tone, rather than reprimanding the user.

Use	Do Not Use
Enter a material class	Material class does not exist
Enter a valid company code	Company code incorrect
Enter a period that ends in the future	You cannot enter a period end in the past
Error during data transfer; check the log	Errors occurred during data transfer
Order is not available; check your entry	Order is not available

Standard Message Short Texts*

The following standard message texts will be included in the Online Text Repository, a tool to assist developers when writing message short texts.

I. Instructions for Entering Data

Enter <object>

Use	Do Not Use
Enter a valid period	Maintain the period
Enter a short text for the action	Enter short text for action

II. Inability to Display Data

Unable to display <object>; <reason>

Always provide a reason. For example, if the selected revision status cannot be displayed because it is located in a period that was already completed (that is, the period entered is not applicable here), say so in the short text. If you do not have enough space in the short text, insert the reason in the long text.

Use	Do Not Use
Unable to display revision status; period is already completed	You cannot display inventory data here
	Displaying inventory data here is not possible

III. Occurrence of Error(s)

Error while <operation>; <instructions>

In this type of error message, do **not** use a verb. Always indicate how to correct the error.

Use	Do Not Use
Error while processing long text; check the long text indicator	Errors were found
Error while posting; check the log	Error occurred during update

IV. Check for Saving Data

Do you want to save the <object>?
Do you still want to save the <object>?
Do you want to save the <object> first?

Use	Do Not Use
Do you want to save the changed values?	Should the data be saved?
Do you still want to save the data?	Do you want to save before exiting?
Do you want to save the entries first?	Value was changed. Should it be saved?

V. Check for Saving Data, with Data Loss Warning

Unsaved data will be lost. Do you want to save?

Use	Do Not Use
Unsaved data will be lost. Do you want to save?	Unsaved data will be lost
	Data will be lost! Save?

VI. Inability to Execute Operations

<Action> is not permitted; <reason>
<Action> is not possible; <reason>
You cannot <action>; <reason>

Always provide a reason. If you do not have enough space in the short text, insert it in the long text.

Use	Do Not Use
Function is not permitted for user status; <reason>	Function prohibited for user status
You cannot delete here; actual values exist	Deleting not possible because actual values exist
Deletion is not permitted; <reason>	Deletion unallowed
You cannot configure a planning variant here; vendor address is invalid	Material number invalid
	Material notice cannot be maintained here
	Valid selection screen does not exist
	Operation & is not supported

VII. Lack of Authorization

You are not authorized to <do this>

Avoid using transaction codes. State the function instead.

Use	Do Not Use
You are not authorized to initialize program <program name>	No authorization to delete status
You are not authorized to display master data	You do not have change authorization for this control data
Your authorization profile & does not allow changing data	

VIII. Selecting Objects

Select <object>
Additional <objects> are not selected

Use	Do Not Use
Select the lines to be deleted	Select end of block
Select the row above which the entry should be inserted	

Additional status schemes are not selected	

IX. (Non-)existence or (Un-)availability of Objects

<Object> does not exist
<Object> is not available
<Object> already exists
<Object> is already available

In certain cases, there is a difference between an object that *does not exist* and one that *is not available.* For example, if a budget "does not exist," it has not been created. However, a budget may not be "available" even though it "exists"; in that case, it has been created but not allocated.

Check whether the object is unavailable or nonexistent because the user has made an incorrect entry. If this is the case, provide instructions.

Use	Do Not Use
Info object does not exist; define class type & first	Class type & is not defined
Data is not available; define the result areas first	No result areas are defined
Alternative unit does not exist; define an alternative unit first	No alternative unit entered

X. Search Results

<Object> not found
No <object> found

In search result messages, do **not** use an auxiliary verb.

Use	Do Not Use
No documents found	Documents were not found
No suitable entry found	No suitable entry was found

XI. Continuing Processing

Processing continued
Do you want to continue processing?

<Process> continued
Do you want to continue <process *(gerund)***>?**

If the operation or activity is self-evident to the user, use the first or second of the above standard texts; do not specify the technical name of the operation. For example, if the context is the posting of a material master record, use "Processing continued," not "Posting of material master record continued."

Use	Do Not Use
Processing continued	Continuation of processing

Do you want to continue processing?	Do you want to continue?

XII. Terminating Processing

Processing terminated
Do you want to terminate processing?

<Process> terminated
Do you want to terminate <process *(gerund)***>?**

If the operation or activity is self-evident to the user, use the first or second of the above standard texts; do not specify the technical name of the operation. For example, if the context is the posting of a material master record, use "Processing terminated," not "Posting of material master record terminated."

Use	Do Not Use
Processing terminated	Termination of processing
Update terminated	Do you want to terminate?

***Note**: In the examples, the text on the left does not always correspond to that on the right -- that is, it does not necessarily mean "Use <...> instead of <...>."

top

Source: SAP R/3 Style Guide

 Texts

Capitalization

Various studies of scenarios and usability have determined that the interface of applications must also support users in their work. Consistency in the texts on the user interface is an important part of how users perceive our software. This applies not only to terminology, but also to writing style, to ensure that SAP's software has a professional appearance and a consistent look and feel. Users should not notice whether the texts they see are the original language of the interface or a translation, and should see no difference in style regardless of whether the texts were written in Germany, the United States, India, or wherever.

While capitalization is not something that end users pay much attention to, consistency of capitalization not only improves the look and feel of the software but also can help to save costs considerably. English, for example, is the source language for 10 other languages at SAP. If the same text is written in two different ways, even if the difference is only in capitalization or punctuation, it becomes two distinct texts for the languages that translate from English. This slows down translation and increases the risk of inconsistent terminology in those languages, which can confuse the end users.

If you write system texts in English - regardless of whether you are a developer, technical author, or translator - you can help to ensure a consistent interface and keep costs down by adhering to the capitalization guidelines below.

Capitalization of English Screen Elements at SAP

The English language generally distinguishes between two styles of capitalization - title case and sentence style. (Title case is also often referred to as "initial caps", "init caps", or "headline style".)

Title Case

Title case means that the first letter of each word is capitalized, except for certain small words, such as articles and short prepositions. For more detailed information about what is meant by title case, see the Quick Guide to Capitalization in English at SAP below.

Use title case for the following screen elements:

- Menu options and pushbuttons
- Field, checkbox, and radio button labels
- Frame, screen, and dialog box titles
- Tab descriptions
- All hierachy nodes and folder names (roles and so on)
- Possible entries, options in drop-down lists
- Column and row headers
- Titles of reports and lists
- All technical short descriptions (names of programs, function groups, transactions, CUA statuses, and so on)

Sentence Style

In sentence style, only the first letter of the sentence or phrase is capitalized. All words after that are written in lower case, except for proper nouns.

Use sentence style for the following texts:

- Error messages
- Status messages
- Complete sentences and questions on the user interface (such as *Do you want to save?* or *All unsaved data will be lost.*)

Quick Guide to Capitalization in English at SAP

These standards apply to **all** cases, in **all** documentation, where "initial caps" are to be used ("headline style") as opposed to sentence style (first word of sentence or phrase capitalized only), regardless of the specific type of title, heading, header, or interface text.

Capitalize

- Nouns
- Verbs (including *is* and other forms of *be*)
- Participles
- Adverbs (including *than* and *when*)
- Adjectives (including *this*, *that*, and *each*)
- Pronouns (including *its*)
- Subordinating conjunctions (*if*, *because*, *as*, *that*)
- Prepositions and conjunctions with five or more letters (*between*, *without*, *during, about*, *because, through*).
  Remark: *Chicago Manual of Style* (*CMS*) rule is not to capitalize any prepositions.
- First and last words, no matter what part of speech they are
- Prepositions that are part of a verb phrase (*Logging On*, *Setting Up*)
- Both elements of hyphenated words (*How-To, Country-Specific*).
  Remark: This rule differs slightly to the *CMS* rule but is more appropriate for our documentation and user interfaces.
- Words and phrases in parentheses as you would capitalize them if they did not appear in parentheses
- Any words, phrases, fragments, or sentences after a colon or semicolon

Do Not Capitalize

- Coordinating conjunctions (*and*, *but*, *or*, *nor*, *for*)
- Prepositions of four or fewer letters (*with*, *to*, *for*, *at*, and so on)
- Articles (*a*, *an*, *the, some*) unless the article is the first or last word in the title
- The word *to* in an infinitive phrase
- Case-specific product names, words, or phrases (*mySAP.com*, *README* file, *e-Business*, and so on)

Examples

Languages Using the Latin Alphabet	Considerations When Hiring Contractors
What Is the ABAP Dictionary?	Using the Drilldown Function
How to Set Up Your Computer	Setting Up Accounts
How to Use the SAP System	Logging On to TRADOS
mySAP.com: The Future Is Now	Where Do You Want to Go Today?
Showing/Hiding Text (Columns or Rows)	Cross-Client Capability
Automatic Distribution Can Only Use One Style	Small- to Medium-Sized Companies
Deleting from the Cross-Reference Library	Client-Specific Settings
Item in Follow-On Document	Templates and Add-Ins
Save As... or Save All	Include Only Terms a Reader Is Likely to Look Up
Log Off	Searching For Online Guides
Maintaining Substance Data: An Example	Phrases: Where-Used List
How to Format Titles with Microsoft Word	Formatting Titles Without Using the Menu


top

Source: **SAP R/3 Style Guide**