

Dr. Boris Rubarth

First Steps in SAP® ABAP®

Second Edition

- ▶ Step-by-step instructions for beginners
- ▶ Comprehensive descriptions and code examples
- ▶ A guide to creating your first ABAP application
- ▶ Tutorials that provide answers to the most commonly asked programming questions

Table of Contents

Preface	7
1 Getting Started with ABAP	9
1.1 Hello ABAP	9
1.2 Creating a program with the ABAP Editor	10
1.3 Starting the report	14
1.4 Getting help from ABAP keyword documentation	16
1.5 Viewing existing code	17
1.6 Versioning your report	18
1.7 Conclusion	18
2 Using ABAP Statements	19
2.1 Using parameters as input variables	19
2.2 Using conditional branches	22
2.3 Using texts that can be translated	23
2.4 Working with string variables	24
2.5 Using elementary data types	25
2.6 Debugging your report	27
2.7 Conclusion	30
3 Using the ABAP Workbench	31
3.1 Starting the ABAP Workbench	31
3.2 Configuring ABAP Workbench options	33
3.3 Working with packages and transport requests	34
3.4 Transaction code for a report	37
4 Using data types from the ABAP Data Dictionary (DDIC)	41
4.1 Getting started with complex data types	41
4.2 Working with the ABAP Data Dictionary (DDIC)	44
4.3 Understanding internal tables	48
5 Modularization and reusing functionality	53
5.1 Using form routines	53
5.2 Using function modules	56

5.3	Using ABAP classes	71
5.4	Exception concept for classes	80
6	Accessing the database	83
6.1	Using ABAP statements to access the database	83
7	ABAP user interface technologies	91
7.1	Working with messages	91
7.2	Working with list processing	92
7.3	Working with screens and dynpros	101
7.4	Short overview of SAP UI technologies	115
8	Introducing the ABAP Development Tools	121
8.1	Installing ADT	122
8.2	Choosing the ABAP perspective	123
8.3	Creating an ABAP Project	123
8.4	Using the Project Explorer	125
8.5	Creating new objects	127
9	Conclusion	129
A	The Author	131
B	Index	132
C	Disclaimer	136

3 Using the ABAP Workbench

This chapter introduces the ABAP Workbench and explains the relationship between the ABAP Workbench and the development tools. We will take a closer look at the ABAP Editor, one of the most widely used tools in the ABAP Workbench. We will also look at how to customize some of the Workbench features.

You can start a report without using the ABAP Editor – if you know how to define a transaction code. Let's briefly cover how to define a transaction code.

3.1 Starting the ABAP Workbench

There are two ways to open the ABAP Workbench (also known as the *Object Navigator*).

One option is to start the ABAP Workbench directly with transaction code SE80. Select the object you want to work on (for example, your report) and then the Workbench will launch the appropriate tool (for reports, the ABAP Editor opens).

It works the other way as well: when you are working on your report using the ABAP Editor, open the ABAP Workbench using the menu path UTILITIES • DISPLAY OBJECT LIST. This will open a separate area on the left-hand side – as displayed in Figure 3.1 – which is called the *object list*.

The object list for a report shows the data objects used in the report.

You can open the object list not only in the ABAP Editor, but also from each ABAP tool. The object list is a separate window section with its own navigation; it is not always synchronized with the object displayed on the right-hand side. You can always synchronize both sections in one direction or the other: either use the menu path listed above to show the object list for an object displayed in the tool (right-hand side); alternatively, select an object in the object list (left-hand side) by double-clicking it and the appropriate tool will be displayed on the right-hand side of the screen. Later, we will see that it can be useful not to have both sections synchronized.

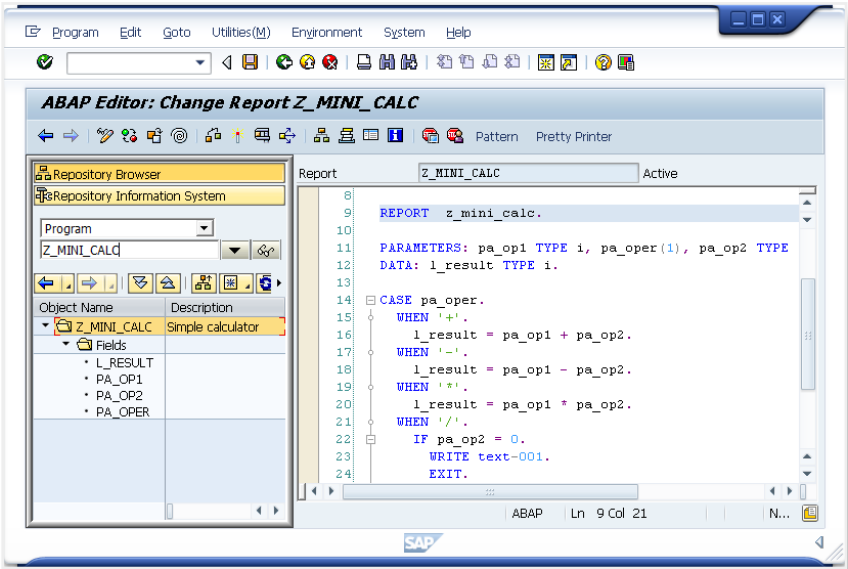


Figure 3.1: The object list for a report

Above the object list, three buttons with icons and text are visible: MIME REPOSITORY, REPOSITORY BROWSER (highlighted), and REPOSITORY INFOSYSTEM. The REPOSITORY BROWSER is currently selected and below that, a dropdown box shows *Program* and the name of our report. Our report is part of the *ABAP Repository*, like many other ABAP object types (for example, ABAP classes) as well.

← The ABAP Repository

All ABAP Repository objects are stored in the database. When we save any changes for our report, the changes are written to the database, into the repository area in the database.

Business data in the database is client-specific data, since a client is a unit that is self-contained in terms of business, organization, and data. However, repository objects are client-independent: a report created in one client can be used in the other clients of that system as well. More than likely, the business results of the report are different when started in different clients, since the report will be working on different business data (which is client-specific data).

You can change user-specific settings to determine which tools are displayed in the ABAP Workbench, as described in the following section.

3.2 Configuring ABAP Workbench options

The ABAP Workbench can display more tools than you will need in your daily work, so it is a good idea to display only the tools you need.

Select UTILITIES • SETTINGS to change your user-specific settings. The first tab WORKBENCH (GENERAL) allows you to set the BROWSER SELECTION of your choice. Use this tab to select only the REPOSITORY BROWSER and the REPOSITORY INFORMATION SYSTEM, as shown in Figure 3.2 below.

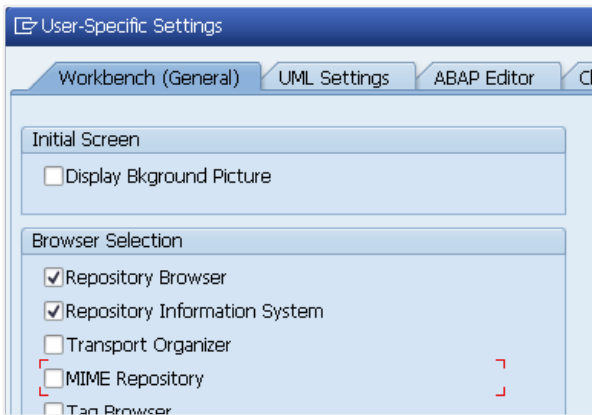


Figure 3.2: ABAP Workbench Settings

As you can see on the third tab, there is an area for user-specific settings for the ABAP EDITOR as well. Select that tab to see the PRETTY PRINTER tab and use the CONVERT UPPERCASE/LOWERCASE option to select KEYWORD UPPERCASE, as shown in Figure 3.3.

After closing the popup window, use the PRETTY PRINTER button to automatically convert ABAP keywords to upper case (and to properly indent the lines, if you selected that option). These options optimize the display of your report according to your preferences, they are not relevant for the run-time of your report.

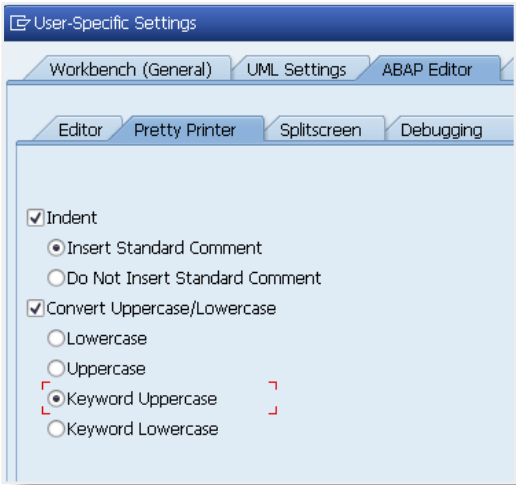


Figure 3.3: ABAP Editor settings for Pretty Printer

Now back to the selection *Program* in the REPOSITORY BROWSER: let’s take a look at the dropdown menu options.

3.3 Working with packages and transport requests

When you open the dropdown menu, you will see some of the existing repository object types, like *Program* and *Package* as displayed in Figure 3.4.

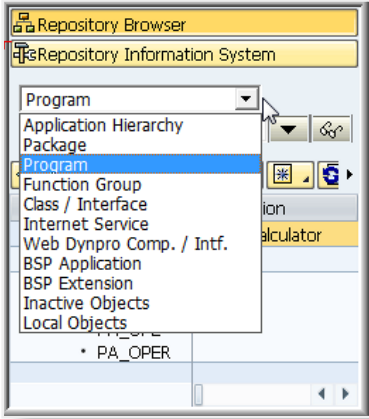


Figure 3.4: Repository object types in the Repository Browser

When we created our first report in Chapter 1, we skipped over an explanation of packages (Figure 1.3). Let's now examine packages in ABAP. Each object is assigned to a package, along with other objects that pertain to the same area of development or business functionality. A package bundles different types of objects, like reports, UI objects, and ABAP classes, as long as they belong together from the application view. Several developers may work on the objects in one package.

Each ABAP object has another assignment that we previously skipped over: transport requests. A transport request bundles objects that will be transported together along the transport path.

← ABAP transports

ABAP objects are developed in a development system. When a development phase is done, the objects are transported to the next system (typically a quality assurance or test system). The third step in a typical transport landscape is a production system, in which no development is allowed.

All three systems are separate installations of AS ABAP, with their own system ID.

A transport request may include objects from different packages, and vice versa. Generally, as a developer, you do not have to worry about transports. As soon as you are done with the current development cycle, you will have to change all of your objects to status *Final* (which is called *Release your transport request*). Typically, the project lead on your team will determine when to transport the objects. The transport is technically carried out by AS ABAP (and its tools).

To create a package, select the menu entry *Package* in the Repository Browser, and enter the name of your package, for example, ZFIRSTPACKAGE. Press : the ABAP Workbench will then display a popup, asking whether you would like to create the (not yet existing) package. Confirm with YES.

Add a short description of the package, as displayed in Figure 3.5.

B Index

A

- ABAP class 71
- ABAP Class Builder 72
- ABAP comments 12
- ABAP Data Dictionary (DDIC) 44
 - Customer namespace 48
- ABAP Development Tools (ADT) 121
- ABAP Editor 10
- ABAP keyword 9
- ABAP keyword documentation 16
- ABAP List Viewer (ALV) 96
- ABAP objects 71
 - Constructor method 79
 - Encapsulation 72
- ABAP Repository 32
- ABAP SQL 83
- ABAP statement 9
- ABAP transports 35
- ABAP Workbench 31
- ABAP Workbench options 33
- access to structure field 42
- architecture of AS ABAP 9, 49
- assignment operator 22
- AUTHORITY CHECK 90

B

- BAPI 58, 65
 - Exceptions 66
- BAPI_FLIGHT_GETLIST 66
- breakpoint 27
- breakpoint types 28
- BSP page 117
- business object 65
- Business Server Page (BSP) 117

C

- CALL FUNCTION 63
- calling class methods 77
- CALL METHOD 77
- CALL SCREEN 102
- CASE 26
- chain statements 20
- class exceptions 80
- client 32
- CLIENT SPECIFIC 86
- COLOR 96
- COMMIT WORK 57
- conditional branch 22
- CONSTRUCTOR 79
- conversion routines 26
- CREATE OBJECT 79
- customer namespace for module pools 110

D

- DATA 24, 43
- database table 44
- Data Browser 48
- data element 45
- data objects 24
- data object types 24
- data type
 - C 24
 - Complex 41
 - D 26
 - Decfloat 25
 - In DDIC 45
 - Local definition 42
- debugging 27
 - Start (/h) 28
- demo application 47
- DESCRIBE 98

developer key
 12
dialog transaction 101
domain 45, 48
drag and drop 78
dynpro 101
 Element list 104
 Flow logic 103
 Menu Painter 109
 Menus and buttons 109
dynpro program 101

E

elementary data types 25
Enhancement Framework 18
ERM 48
event 29, 93
 AT LINE-SELECTION 93
 AT SELECTION SCREEN 93
 INITIALIZATION 93
 START-OF-SELECTION 93
exception classes 80
exception concept for classes 80
exception object 81

F

flight data model 47, 68
Floorplan Manager (FPM) 119
FORM 53
FORMAT 96
form routine 53
 Call 55
 Definition 53
forward navigation 18, 61, 102
Function Builder 56, 59
function group 56, 59
function module 56
 Call 63
 Exception 62
 Testing 62

G

global structure definition 44, 47
Graphical Screen Painter 104
GROUP 95

H

HOTSPOT 95
HTMLB 117

I

ICON_LIST 96
INCLUDE 100
input variables 19
instance method 72
Intellisense 13
internal table 49
 ABAP statements 51
 Work area 49
Internet Transaction Server (ITS)
 116
itab 49

L

list 93
 Basic list 93
 Detail lists 93
 List buffer 93
list buffer 16
list formatting 95
list of inactive objects 21
list processing 92
local classes 72
local objects 37
Logical Unit of Work (LUW) 90
LOOP AT 50

M

MESSAGE 91
messages 91
 Message class 92

- Message type 91
- method parameter types 74
- modularization 53
- module pool 101, 102, 110
- MOVE-CORRESPONDING 42

N

- naming convention 25
- Native SQL 83, 84
- nested structures 41
- next screen 103

O

- object list 31
- Object Navigator 31
- object orientation 71
 - Instance of a class 71
- object selection 37
- OK_CODE 113
- Open SQL 83, 84

P

- package 34, 35
 - \$TMP 37
- PARAMETERS 19
 - Lower-case 20
- pattern function 63, 78
- performance for database access 89
- Pretty Printer 33
- private method 72
- PROCESS AFTER INPUT (PAI) 104
- PROCESS BEFORE OUTPUT (PBO) 104
- production system 35
- public method 72

R

- RADIOBUTTON 94
- RAISE EXCEPTION 81

- relational database management system (RDBMS) 83
- remote-enabled function module 58
- remote-enabled module 60
- reports 9
- report transaction 39
- Repository Browser 32
- RETURN parameter 66
- RFC 60
- RFM 60, 65, 72
- runtime object 46

S

- SAP Business Technology Platform (BTP) ABAP environment 14
- SAP Fiori 119
- SAPGUI for Windows 39
- SAP NetWeaver Application Server ABAP 9
- SAPUI5 119
- Screen Painter 103
- screen processing 101
- SELECT 84
- selection screen 21, 39, 94
- selection text 20
- sflight 87
- SFLIGHT 47
- SKIP TO LINE 95
- static method 72
- string variable 24
- subroutine 53
- sy-lilli 94
- sy-listi 94
- sy-lsind 94
- SY-SUBRC 46

T

- Table Control Wizard 109, 111
- table keys 51
- TABLE statement 115

- testing class methods 76
- test system 35
- text pool 23, 24
- text symbols 23
- TOP include 109, 110
- transaction code 37
- translatable text 23
- transparent table 47, 48
- transport request 34, 35
- try/catch 81
- TYPES 42
- TYPE TABLE OF 49

U

- user interfaces 91

V

- VALUE 25
- versioning 18

W

- Web Client UI framework 118
- Web Dynpro ABAP 118
- work area 50
- work process 83