ABAP Modularization

Adrian Streitz, Johannes Rank, Borys Levkovskyi

Prerequisites

- Navigation experience with ABAP Workbench and Eclipse
- Advanced ABAP programming skills
- Basic knowledge in ABAP object-oriented programming

Agenda

I. Introduction

II. Includes

III. Function Modules

IV. Business Application Programming Interfaces (BAPIs)

V. Subroutines (FORMs)

Motivation and Fundamentals

- Process of **subdividing** a program into separate parts
- **Reuse** of certain functions without maintaining several copies
- Enables to create independent work tasks for a team of software developers → **collaboration**
- Facilitates to cope with higher complexity → **Divide-and-Conquer**
- Easy identification of **software errors**
- Easy way to **extend** and **substitute** existing program code
- There are different possibilities to make use of modularization for ABAP

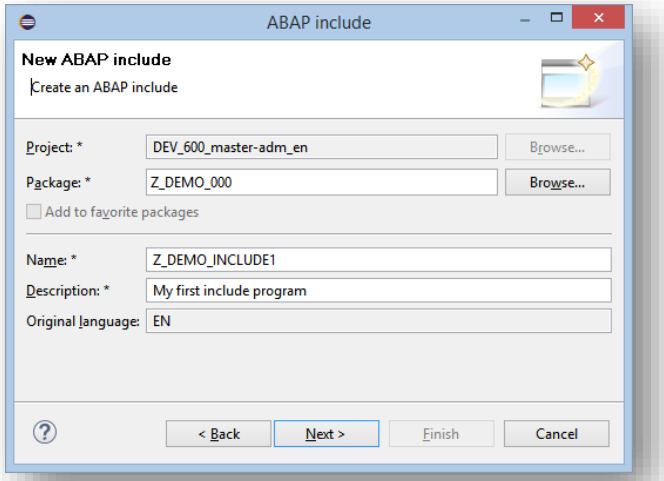
Forms of Modularization

1. Includes
2. Function Modules
3. Business Application Programming Interfaces (BAPIs)
4. Subroutines (FORMs)

First way to reduce complexity

Idea

- Outsource to external program
- Instruction INCLUDE integrates external program into main program



Drawbacks

- Bad maintenance: changes in include programs may cause syntax errors in main programs
- High memory consumption → multiple memory allocation of include program

Characteristics

- No independent run possible → must be integrated into other programs
- Include programs may contain other includes
- Include programs cannot call themselves
- Manual check of calling program by software developer necessary → does the include program fit logically?

INCLUDE	TOP-INCLUDE
no data declaration necessary	contains data declaration
operational code	considered for all syntax checks
	always integrated for compiler
	no operational code

Fundamentals

- Outsources functionality to external modules
- More than 100,000 function modules available by a default SAP ERP installation
- Function modules can be organized in function groups
- Function modules can be remote accessible
- Function groups may have own TOP-INCLUDE

Export-, Import- and Changing-Parameters

import (FM) = export (ABAP report)

```
flight_list
  importing value(cityfrom)  type s_from_cit
            value(cityto)    type s_to_city
            value(datefrom)  type s_date
            value(dateto)    type s_date
  exporting value(msg)       type string
  tables    flight_list      type standard table of wdr_flights with header line

Documentation
  DE-EN-LANG-SWITCH-NO-TRANSLATION

Parameters
  cityfrom   Departure city
  cityto     Arrival city
  datefrom   Flight date
  dateto     Flight date
  msg        Predefined Type
  flight_list Example Flights
```

export parameter

import parameter

table

report → FM

```
CALL FUNCTION 'FLIGHT_LIST'
  EXPORTING
    cityfrom    =     " Departure city
    cityto      =     " Arrival city
    datefrom    =     " Flight date
    dateto      =     " Flight date
*  IMPORTING
*    msg         =     " Predefined Type
  TABLES
    flight_list =     " Example Flights
.
```

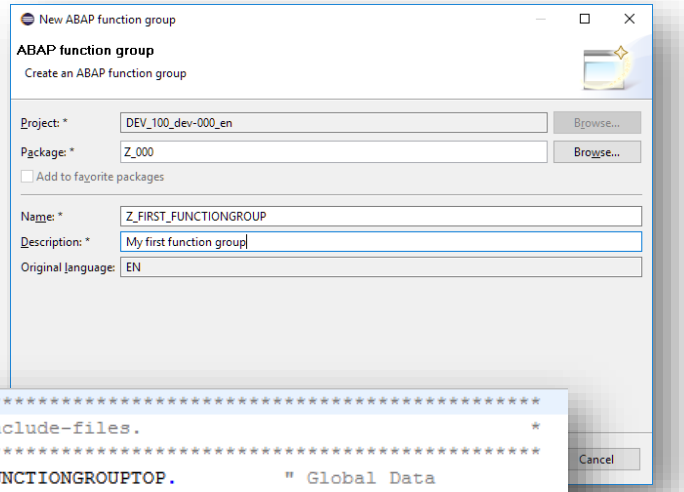
FM → report

report ←→ FM

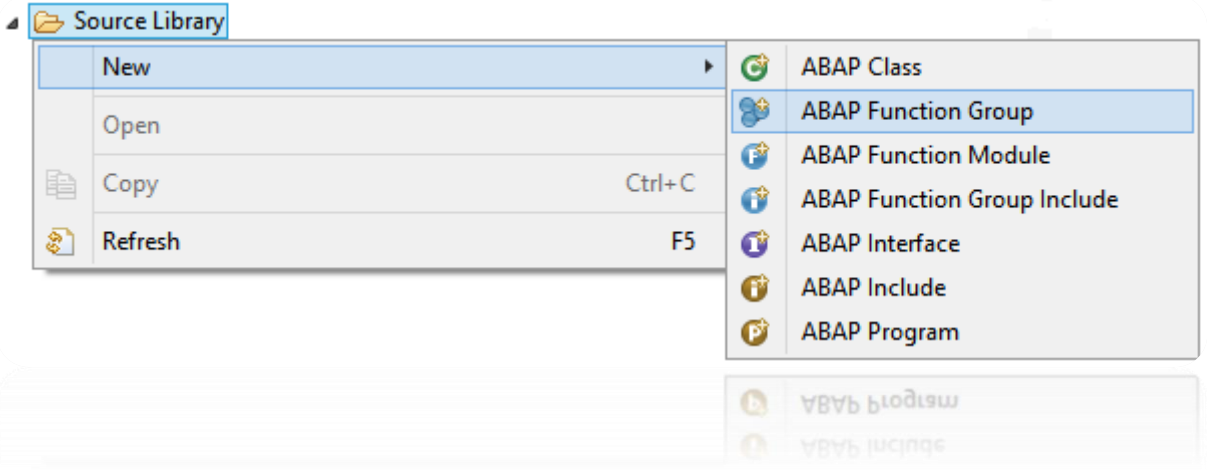
Exceptions are missing in this example, but may be raised by function modules!

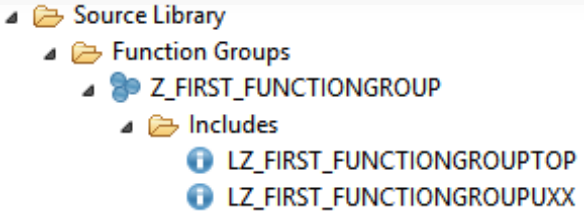
Function Groups

- Container for function modules
- Not executable
- Holds **global data** for all containing function modules



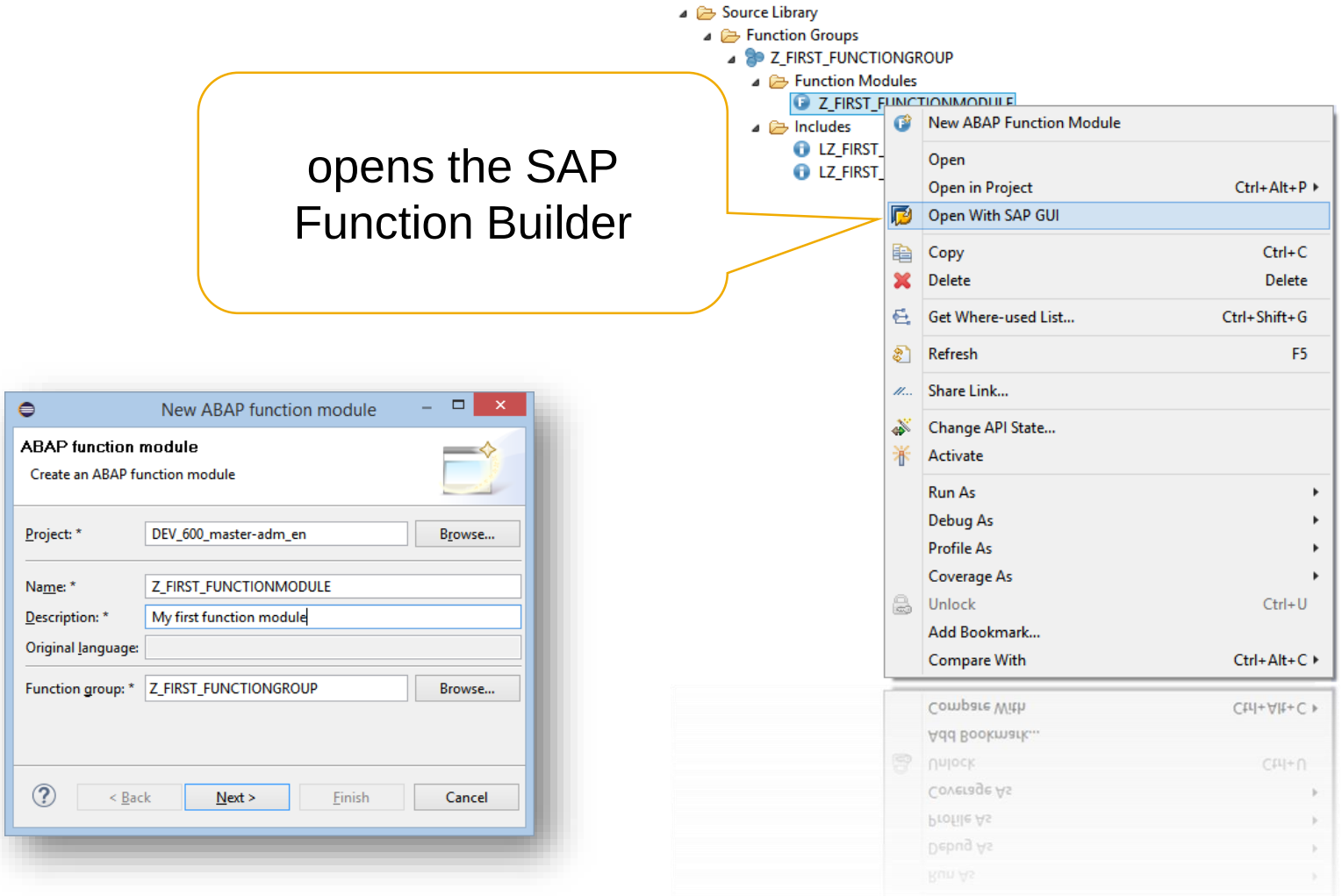

```
1  ***********************************************************************
2  *   System-defined Include-files.                                     *
3  ***********************************************************************
4    INCLUDE LZ_FIRST_FUNCTIONGROUPTOP.          " Global Data
5    INCLUDE LZ_FIRST_FUNCTIONGROUPUXX.          " Function Modules
6
7  ***********************************************************************
8  *   User-defined Include-files (if necessary).                        *
9  ***********************************************************************
10 * INCLUDE LZ_FIRST_FUNCTIONGROUPF...          " Subroutines
11 * INCLUDE LZ_FIRST_FUNCTIONGROUPO...          " PBO-Modules
12 * INCLUDE LZ_FIRST_FUNCTIONGROUPI...          " PAI-Modules
13 * INCLUDE LZ_FIRST_FUNCTIONGROUPE...          " Events
14 * INCLUDE LZ_FIRST_FUNCTIONGROUPP...          " Local class implement.
15 * INCLUDE LZ_FIRST_FUNCTIONGROUPT99.          " ABAP Unit tests
```





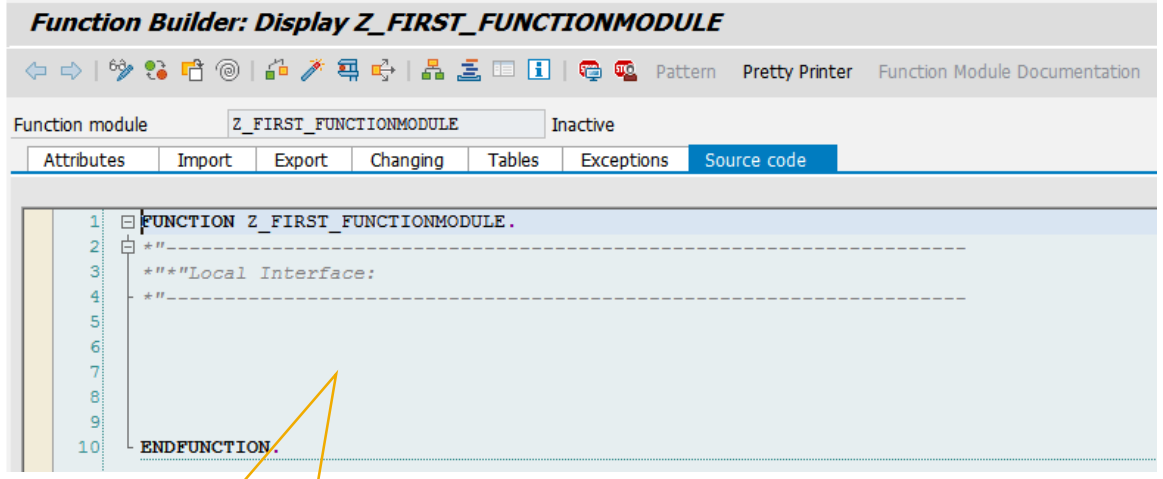
Create your first Function Module: Step 1

1. Check whether a suitable function module already exists. If not, proceed to step 2.
2. Create a function group, if no appropriate group exists yet.
3. Create the function module.
4. Define the function module interface by entering its parameters and exceptions (use the Function Builder)



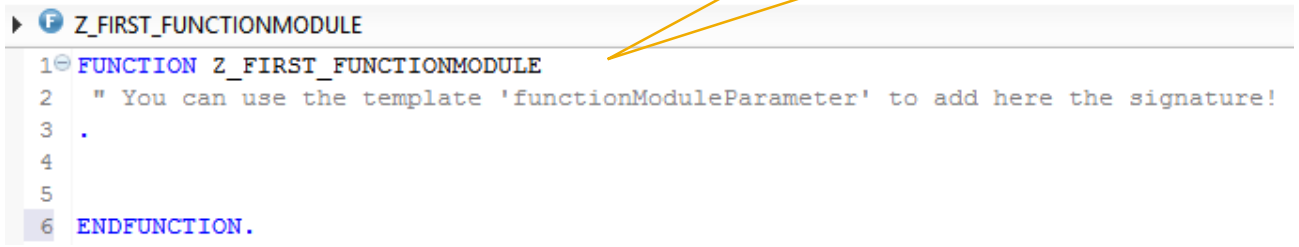
Create your first Function Module: Step 2

5. Write the actual ABAP code for the function module, adding any relevant global data to the TOP include.
6. Activate the module.
7. Test the module.
8. Document the module and its parameters for other users.
9. Release the module for general use.



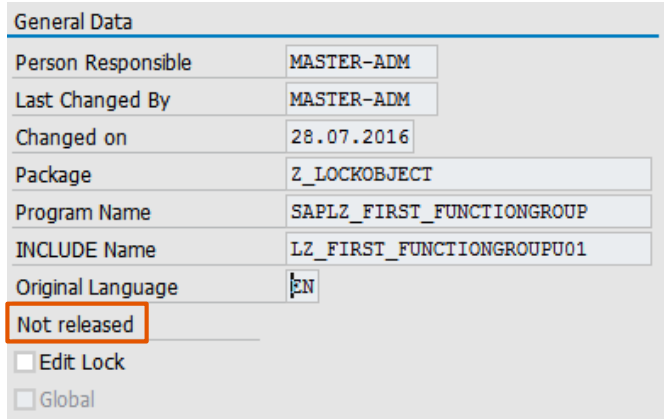
SAP GUI

Eclipse

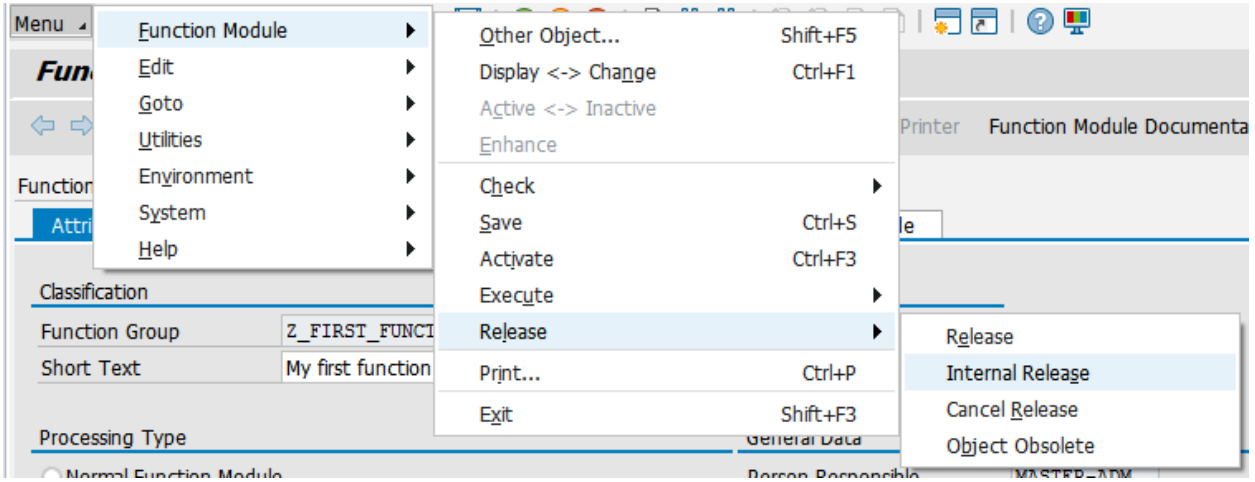


Release Status

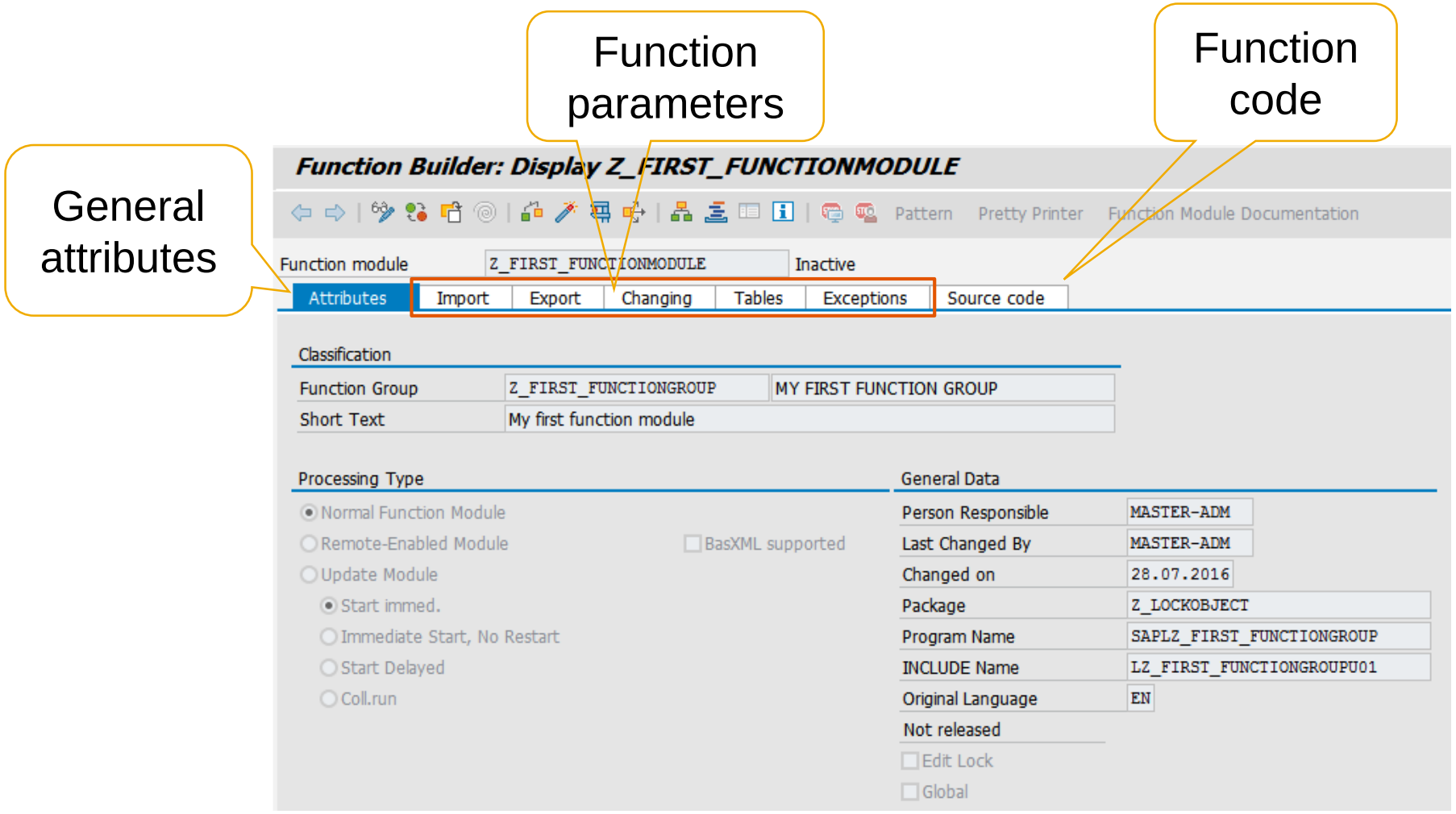
- **External release**
  - for customers
  - ensure upward compatibility
  - no future need to change calling statement ever



- **Internal release**
  - for internal usage
  - ensure upward compatibility
  - users have to be informed promptly when function module has changed



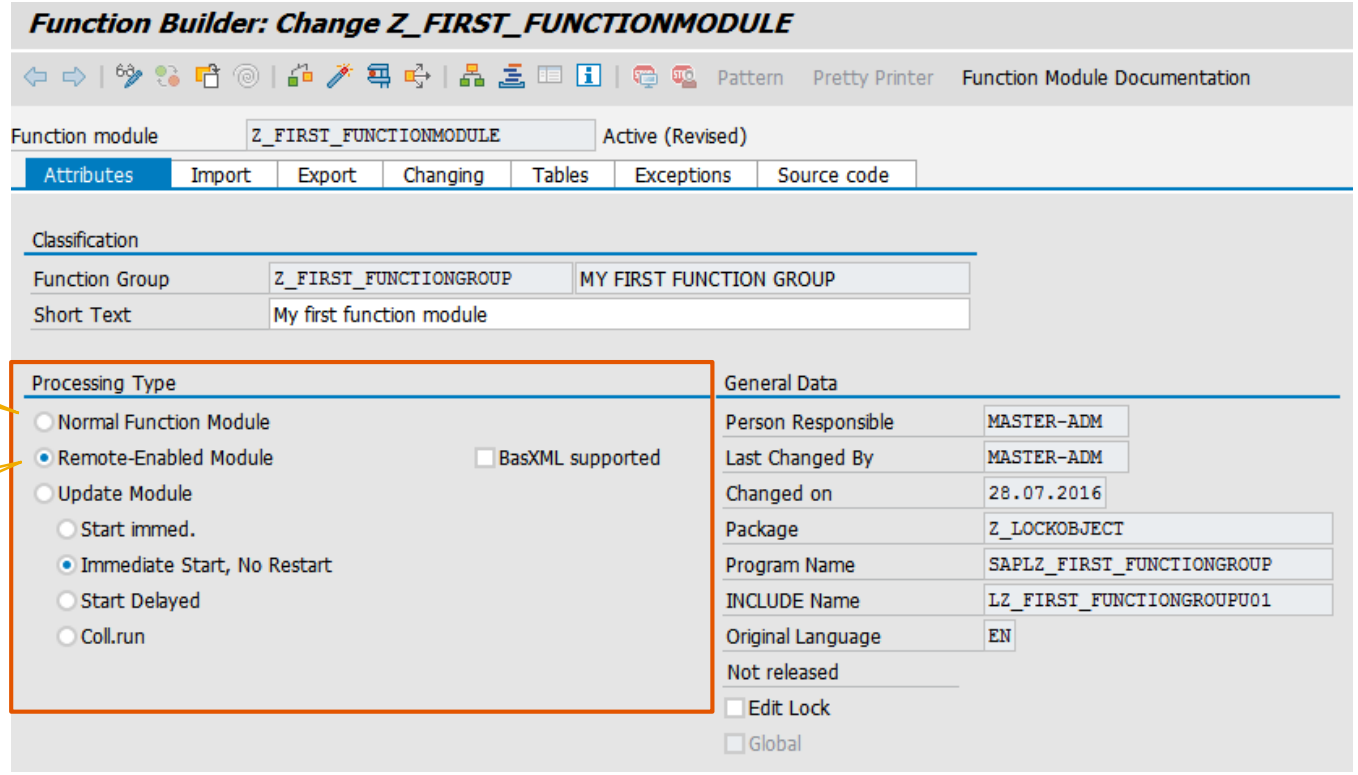
Function Builder



Function Module Types

local function

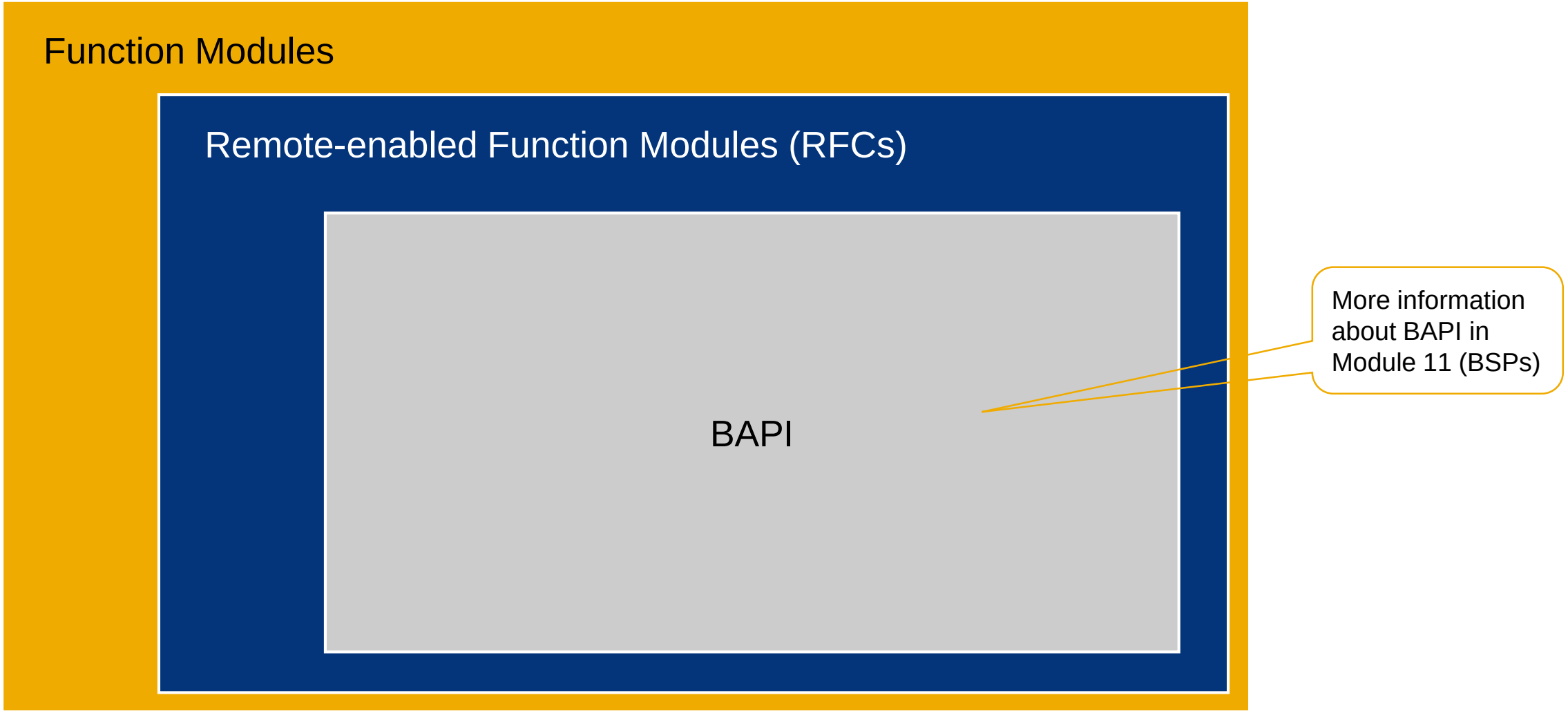
remote function



Introduction

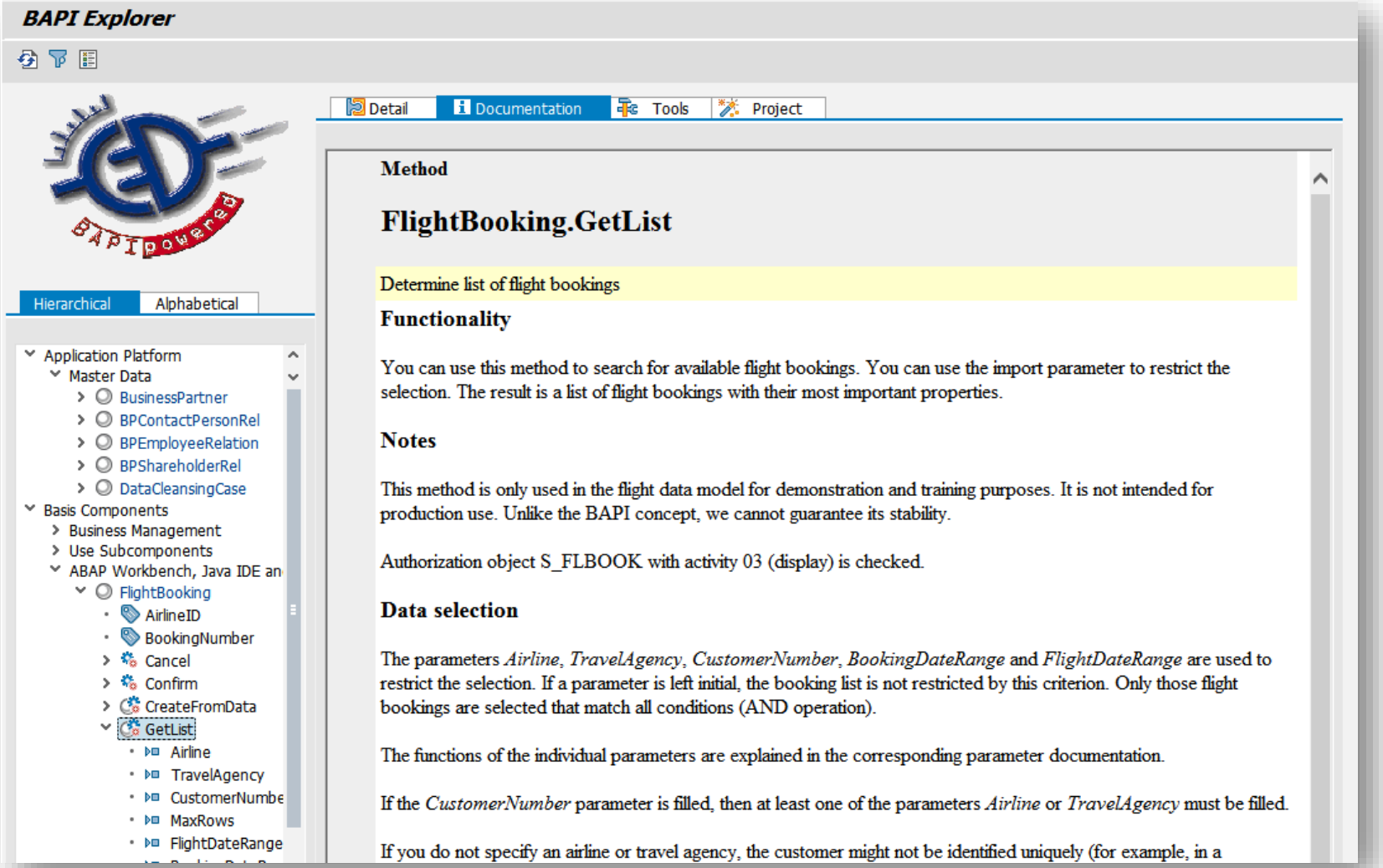
- Precisely defined interfaces providing access to processes and data of SAP business applications
- Special type of RFC enabled function modules
- May be accessed outside the SAP system (Visual Basic or Java applications)
- Object-based communication between components
- Set of interfaces → enables easy integration of third-party software into the proprietary SAP applications
- BAPI = Business Application Programming Interface

Classification



BAPI Explorer

- Most common to use existing BAPIs of the SAP system
- Repository of all available BAPIs: **BAPI Explorer**
- Open Transaction BAPI within the SAP GUI
- List sorted by:
  - Alphabetical order
  - Business areas



Usage and application

CTRL + Space

Parameter preview

```
CALL FUNCTION 'BAPI_fli'.
```

bapi_flight_checkavailibility
bapi_flight_getdetail
bapi_flight_getlist
bapi_flight_saveresplica

Press 'Shift+Enter' to insert full signature

Shift + Enter

bapi_flight_getlist

importing	value(**airline**)	type s_carr_id optional
	value(**destination_from**)	type bapisfldst optional
	value(**destination_to**)	type bapisfldst optional
	value(**max_rows**)	type bapimaxrow optional
tables	**date_range**	type standard table of bapisfldra with header line optional
	extension_in	type standard table of bapiparex with header line optional
	flight_list	type standard table of bapisfldat with header line optional
	extension_out	type standard table of bapiparex with header line optional
	return	type standard table of bapiret2 with header line optional

Documentation

Find list of flights

Longtext documentation

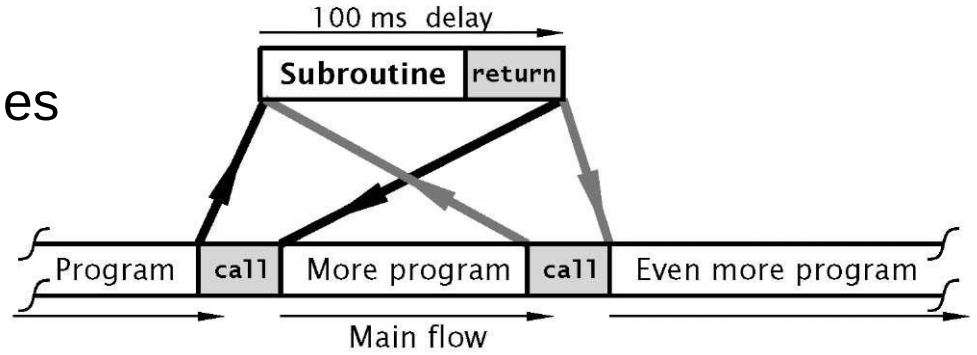
Parameters

airline	Select airline
destination_from	Select departure city
destination_to	Select destination
max_rows	Maximum Number of Lines of Hits
date_range	Selection range for flight date
extension_in	Import customer enhancements
flight_list	List of flights
extension_out	Export customer enhancements
return	Return Messages

```
CALL FUNCTION 'BAPI_FLIGHT_GETLIST'
*  EXPORTING
*    airline          =      " Select airline
*    destination_from =      " Select departure city
*    destination_to   =      " Select destination
*    max_rows         =      " Maximum Number of Lines of Hits
*  TABLES
*    date_range       =      " Selection range for flight date
*    extension_in     =      " Import customer enhancements
*    flight_list      =      " List of flights
*    extension_out    =      " Export customer enhancements
*    return           =      " Return Messages
  .
```

Fundamentals and Characteristics

- Subroutines are procedures in ABAP
- **Parameter without value declaration** means the variable points to the global variable
- **Parameter with value declaration** have their own values
- ABAP doesn’t allow nested subroutines
- Place your subroutines always at the end of your code



Defining Subroutine

passed by reference

passed by value

```
FORM <procedure name>
  USING <input parameter> TYPE <type>
  CHANGING <input/output parameter> TYPE <type>
 USING value<input parameter> TYPE <type>
 CHANGING value<input/output parameter> TYPE <type>.
ENDFORM.
```

For calling by reference, **USING** and **CHANGING** are equivalent. For documentation purposes, you should use **USING** for input parameters which are not changed in the subroutine, and **CHANGING** for output parameters which are changed in the subroutine.

Call an existing subroutine

```
PERFORM <procedure name>
[IN PROGRAM <program name>]
  USING <input parameter>
  CHANGING <input/output parameter>.
```

Call subroutine of another program (optional)

read-only parameters

Provide paramaters which may be changed

Now you know how create and use functional modules.

To consolidate your knowledge, you can do task 1 and 2 of the Modularization exercise.

Check your knowledge

Check your knowledge

- Using an INCLUDE results in a copy of source code at the calling position of the main program.

  ☐ True ☐ False

- Function modules are totally independent and do not need any other repository objects to be operable.

  ☐ True ☐ False

- Explain the difference between the two learned release states of ABAP Function Modules!

- Parameter without value declaration means the variable is locally created and only used within the processed subroutine.

  ☐ True ☐ False

Solution

Solution

- Using an INCLUDE results in a copy of source code at the calling position of the main program.

  ☒ True ☐ False

- Function modules are totally independent and do not need any other repository objects to be operable.

  ☐ True ☒ False

- Explain the difference between the two learned release states of ABAP Function Modules!

  See section *Release Status*

- Parameter without value declaration means the variable is locally created and only used within the scope of the subroutine.

  ☐ True ☒ False

References

- Schwaiger, R. (2016): Schrödinger programmiert ABAP: Das etwas andere Fachbuch, SAP PRESS, 2016
- Keller, H.; Krüger, S. (2006): ABAP Objects: ABAP-Programmierung mit SAP NetWeaver, Rheinwerk Publishing

© 2018 SAP SE or an SAP affiliate company. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP SE or an SAP affiliate company.

SAP and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP SE (or an SAP affiliate company) in Germany and other countries. Please see http://global12.sap.com/corporate-en/legal/copyright/index.epx for additional trademark information and notices.

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

National product specifications may vary.

These materials are provided by SAP SE or an SAP affiliate company for informational purposes only, without representation or warranty of any kind, and SAP SE or its affiliated companies shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP SE or SAP affiliate company products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.

In particular, SAP SE or its affiliated companies have no obligation to pursue any course of business outlined in this document or any related presentation, or to develop or release any functionality mentioned therein. This document, or any related presentation, and SAP SE's or its affiliated companies' strategy and possible future developments, products, and/or platform directions and functionality are all subject to change and may be changed by SAP SE or its affiliated companies at any time for any reason without notice. The information in this document is not a commitment, promise, or legal obligation to deliver any material, code, or functionality. All forward-looking statements are subject to various risks and uncertainties that could cause actual results to differ materially from expectations. Readers are cautioned not to place undue reliance on these forward-looking statements, which speak only as of their dates, and they should not be relied upon in making purchasing decisions.