Interface Programming in SAP® ABAP

- Overview of classic SAP ABAP interface techniques
- IDoc and ALE
- Remote Function Call (RFC) and BAPIs
- Using Remote Function Module (RFM) und BAPIs in SAP S/4HANA
# Table of Contents

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preface</td>
<td></td>
<td>9</td>
</tr>
<tr>
<td>1</td>
<td>Using Remote Function Call (RFC)</td>
<td>11</td>
</tr>
<tr>
<td>1.1</td>
<td>Implementing an RFC in three minutes</td>
<td>11</td>
</tr>
<tr>
<td>1.2</td>
<td>Creating RFC destinations (to ABAP systems)</td>
<td>14</td>
</tr>
<tr>
<td>1.3</td>
<td>Considering target system configuration</td>
<td>17</td>
</tr>
<tr>
<td>1.4</td>
<td>Checking the remote enablement of modules</td>
<td>19</td>
</tr>
<tr>
<td>1.5</td>
<td>Handling RFC exceptions</td>
<td>23</td>
</tr>
<tr>
<td>1.6</td>
<td>Remote debugging of RFMs</td>
<td>28</td>
</tr>
<tr>
<td>1.7</td>
<td>Monitoring connections on the gateway</td>
<td>31</td>
</tr>
<tr>
<td>1.8</td>
<td>Checking RFC traces</td>
<td>33</td>
</tr>
<tr>
<td>1.9</td>
<td>Searching for RFMs</td>
<td>35</td>
</tr>
<tr>
<td>1.10</td>
<td>Creating your own data objects</td>
<td>36</td>
</tr>
<tr>
<td>1.11</td>
<td>Creating RFMs to read data</td>
<td>45</td>
</tr>
<tr>
<td>1.12</td>
<td>Extending code with an RFC statement</td>
<td>57</td>
</tr>
<tr>
<td>1.13</td>
<td>Introducing asynchronous techniques</td>
<td>58</td>
</tr>
<tr>
<td>1.14</td>
<td>Using background RFC (bgRFC)</td>
<td>61</td>
</tr>
<tr>
<td>1.15</td>
<td>Configuring the system for bgRFC</td>
<td>65</td>
</tr>
<tr>
<td>1.16</td>
<td>Monitoring bgRFC processing</td>
<td>67</td>
</tr>
<tr>
<td>1.17</td>
<td>Checking bgRFC guaranteed delivery</td>
<td>68</td>
</tr>
<tr>
<td>1.18</td>
<td>Using queues for bgRFC</td>
<td>70</td>
</tr>
<tr>
<td>1.19</td>
<td>Creating RFMs to change data</td>
<td>72</td>
</tr>
<tr>
<td>1.20</td>
<td>Testing RFMs with database changes</td>
<td>80</td>
</tr>
<tr>
<td>1.21</td>
<td>Testing an RFM with eCATT</td>
<td>82</td>
</tr>
<tr>
<td>1.22</td>
<td>Comparing synchronous and bgRFC</td>
<td>92</td>
</tr>
<tr>
<td>1.23</td>
<td>Discussing external numbering</td>
<td>101</td>
</tr>
<tr>
<td>1.24</td>
<td>Using tRFC and qRFC</td>
<td>102</td>
</tr>
<tr>
<td>1.25</td>
<td>Changing and extending RFMs</td>
<td>103</td>
</tr>
</tbody>
</table>
2 Using BAPIs

If you ask experienced colleagues what a BAPI is, their answers may range from “Oh, just another RFM” to “A method to access business data inside SAP solutions encapsulated in a business object”. Both statements are correct: the colleagues just see things from different angles. We will start by looking at the shorter answer: a BAPI is a remote-enabled function module.

2.1 Calling a BAPI by RFC in three minutes

A BAPI is an RFM and its name starts with BAPI_. If, for example, you want to get a list of users from the target system, your coding may look as shown in Listing 2.1:

```plaintext
REPORT z_bapi_call.
PARAMETERS pa_dest
  TYPE rfcdes-rfcdest DEFAULT 'FIRSTDEST'.
DATA: t_user TYPE TABLE OF bapiusname,
    l_user TYPE bapiusname.

CALL FUNCTION 'BAPI_USER_GETLIST'
  DESTINATION pa_dest
  TABLES
    userlist = t_user.

LOOP AT t_user INTO l_user.
  WRITE: /, l_user-username.
ENDLOOP.
```

*Listing 2.1: Simplified RFC to BAPI example*

Was that all we need to know about BAPIs? Of course not … and again this introductory coding is not sufficient, so let us dig deeper.
2.2 Getting the overview on BAPIs

With the knowledge from the previous chapter on RFCs, you can already call a BAPI using RFC as the protocol, as the BAPI implementation is always an RFM. So why are you reading this chapter? BAPIs offer some very important and helpful features for your remote communication but also hold some traps—so it is worth examining some of the details. You may also be curious about what “BAPI implementation” means.

Let’s start with the name: BAPI is the abbreviation for Business API, in the sense of an interface to business data.

**BAPI or BAdI**

Do not confuse a BAPI with a BAdI: a BAdI is a Business Add-In for a modification-free extension of existing ABAP coding. And a BAPI may offer a BAdI—a point to extend the BAPI without modification, as we will discuss later.

If an RFM implements a BAPI, then you know how to examine the BAPI: you examine the RFM. Nevertheless, if the RFM is the implementation, there must be something beyond the implementation, some meta level that the RFM is implementing.

You can use the Function Builder selection screen to search for RFMs with the prefix BAPI_. (Later on, we will discuss a smarter way to search for BAPIs.) The next part of the name is the object to which the BAPI is related. Remember: the other colleague mentioned something like “data … encapsulated in a business object”.

The “object” belongs to the meta level beyond the implementation. This object is a representation of business data to which the BAPI shall offer an interface. A simple example is BAPI_USER_GETLIST: this BAPI provides a list of system users, so User is the object type to work on and getlist is the method—makes sense, doesn’t it? Figure 2.1 illustrates the business object with its methods and the related RFMs which are relevant for the RFC communication.
Open this BAPI (the RFM BAPI_USER_GETLIST) in the Function Builder and check the attributes: the function module is indeed remote-enabled. However, if you are looking for attributes revealing that this RFM is a BAPI, there is no clue available here, the only indication is the name.

Moreover, the source code of the BAPI-RFM is just like the source code for any other RFM. Therefore, we need another way to examine BAPIs and find out what makes them special.

**Object-oriented technique or module technique**

Although a BAPI relates to an object, the coding is not **object-oriented** in the sense that the object belongs to an ABAP class. No ABAP class exists for the object to which the BAPI belongs.

Nevertheless, the implementation inside the function module can use ABAP object statements, just like any other RFM implementation.
2.3 Exploring BAPIs

Instead of searching for the name of an RFM in the Function Builder, you can use the BAPI Explorer. This is a comfortable and well-structured tool that you can use to search for an appropriate interface for your scenario. You open the BAPI Explorer using the transaction code BAPI—it’s easy to remember. Nevertheless, the BAPI Explorer initially does not show the BAPIs but rather the objects, which we call business objects.

The BAPI Explorer lists these business objects and their interfaces (BAPIs) by application component hierarchy (HIERARCHICAL tab) or by name (ALPHABETICAL tab).

**BAPI benefit: Easy to find**

You can find appropriate BAPIs for your project in the BAPI Explorer using the application component hierarchy for orientation.

Switch to the ALPHABETICAL tab first and then scroll down to select the object USER. The DETAIL tab then displays details for this object, as shown in Figure 2.2.

The right-hand side of the screen offers additional tabs: DOCUMENTATION indicates that each business object, with its BAPIs, is well documented; the PROJECT tab offers a guided way to create your own business objects and BAPIs; the TOOLS tab offers several tools based on the object selected.

**BAPI benefit: Well documented**

You will find documentation for each BAPI. This is in contrast to RFMs, which may not be documented by SAP in each case.

Now select the method GetList of the object USER. The details reveal the name of the RFM that implements the BAPI, as shown in Figure 2.3.
Figure 2.2: Transaction BAPI shows details for the object USER

Figure 2.3: Details for BAPI_USER_GETLIST
B  Index

A
ABAP inline 82
ABAP List Viewer (ALV) 51
ABAP update technique 72
Acknowledgement 203
ALE audit responses 203
ALE configuration IMG 219
ALE message type 181
ALE port 193
ALE status monitor 187
Application Link Enabling (ALE) 181
Asynchronous processing 58
Asynchronous RFC (aRFC) 59
Authorization 40
Authorization object S_RFC 17

B
Background RFC (bgRFC) 61
  Using queues 70
BAPI 121
BAPI Explorer 124
BAPI extension concept 139
BasXML support 113
Best-effort 59
bgRFC destination 61
bgRFC unit 61
bgRFC unit ID 68
Blacklisted RFMs 231
Business Add-In (BAdI) 104
Business Object Builder 170

Business object types 126
Business objects 124

C
CATT 85
Communication interface 12
Conversion exits 52
Customer BAPIs 155

D
Decoupling 58
Deep data types 112
Distribution model 205

E
eCATT interface 86
eCATT system data container 82
eCATT test script 82
Electronic Data Interchange (EDI) 221
Exactly once (EO) 58
Exactly once in order (EOIO) 58
Extended Computer Aided Test Tool (eCATT) 82
External numbering 101

F
Fast RFC 115
Fast serialization test 116
Function group 46
Function module documentation 50

G
Gateway 31
Gateway Monitor 31
Gateway registration 107
Gateway service 31
Guaranteed delivery 58

I
IDoc 181
Communication IDocs 217
Master IDoc 217
Implementation Guide (IMG) 219
Implicit COMMIT WORK 57
Inbound ALE module 185
Interaction pattern 58
Interface 12
Intermediate document 181

L
Logical system 191
Logical unit of work (LUW) 57

M
Material Number Field Length Extension 232
Message class 42

N
Naming conventions 13
No direct response 58

O
Object Navigator 45
Outbound ALE module 185
Outbound scheduler 66

P
Partner profile 195
Performance analysis 34
Position orientation 111
Processing type 20
Profile Generator 40
Program ID 107

Q
Quality of Service (QoS) 59
Queued RFC (qRFC) 102

R
Registered server program 106
Release RFM 53
Remote debugging 28
Remote Function Call (RFC) 11
Repository Information System 35
Request and response 12
RETURN structure for BAPIs 129
RFC client and server 12
RFC destination 14
RFC destination 14
BACK 16
NONE 16
Type 3 14
Type T 106
RFC exceptions 23
Class-based 25
Generic 24
RFC library 226
RFC statement 11
RFC traces 33
RFM 20
RFM test
  Test sequence 81

S
SAP BC flow 224
SAP BW/4HANA 233
SAP Connectors 223
  SAP Business Connector 224
  SAP Connector for Microsoft .NET 227
  SAP Java Connector (JCo) 225
  SAP NetWeaver RFC Library 225
SAP Gateway 31
SAP HANA database 229
SAP S/4HANA 229
Secure network communication (SNC) 17
Serialization 109
  basXML 112
  basXML forced 114
  Fast serialization 115
  XRFC 111
Simplification Item 232
Simplification List 230
Stable interface 138
Suite-on-HANA (SoH) 229
Supervisor destination 66
Synchronous RFC 59
Synchronous RFC (sRFC) 57

T
Table appends 142
Terminal ID 29
TEST_RUN parameter 163
Transaction ID 101
Transactional RFC (tRFC) 102
Transfer protocol 110
Trusting/trusted RFC connection 17

U
Unified Connectivity (UCON) 19
Update function module 72

V
Versioning a BAPI 138
Versioning an interface 105

W
Wrapper RFM 104

X
X structure for BAPI 132