Sergey Korolev

**ABAP™ Development for Financial Accounting**

Custom Enhancements
Contents at a Glance

1 Enhancement Types ................................................................. 17
2 Master Data Enhancements .................................................... 39
3 Posting to Accounting ............................................................. 111
4 Enhancements in Reports ........................................................ 151
5 Inbound Scenarios in Financial Accounting ............................ 167
6 Outbound Scenarios in Financial Accounting ........................ 197
7 Workflow as a User Exit .......................................................... 227
Contents

Introduction ............................................................................................................. 13
Acknowledgments .................................................................................................. 15

1 Enhancement Types .......................................................................................... 17

1.1 Customer Enhancements (CMOD/SMOD) .............................................. 17
  1.1.1 Function Module Exit ................................................................. 18
  1.1.2 Menu Exit ................................................................................ 19
  1.1.3 Customer Exit Subscreen ........................................................ 20
  1.1.4 Finding Customer Enhancements ............................................. 21
  1.1.5 Enhancements Summary .......................................................... 23
1.2 Business Transaction Events (BTE) .......................................................... 24
  1.2.1 Events and Processes ................................................................. 24
  1.2.2 Configuration ......................................................................... 25
  1.2.3 Finding Business Transaction Events .................................... 30
  1.2.4 Business Transaction Events Summary .................................. 31
1.3 Business Add-In (BAdI) ........................................................................ 31
  1.3.1 Classic BAdI ........................................................................... 32
  1.3.2 Kernel-Based BAdI ................................................................. 34
  1.3.3 Filtered BAdIs ........................................................................ 35
  1.3.4 BAdI Subscreen and Function Codes ..................................... 36
  1.3.5 Finding BAdIs ........................................................................ 37
  1.3.6 BAdI Summary ....................................................................... 37
1.4 Implicit Enhancements .............................................................................. 38
1.5 Summary ..................................................................................................... 38

2 Master Data Enhancements .............................................................................. 39

2.1 General Ledger Accounts ........................................................................... 39
  2.1.1 Main Transaction Codes for General Ledger Account Master Data ........................................... 40
  2.1.2 Data Enhancement of General Ledger Account Master Data Tables .................................. 41
  2.1.3 Screen Layout Enhancement ...................................................... 52
2.1.4 Other Enhancements Available in General Ledger
   Account Master Data ..................................................... 65
2.1.5 General Ledger Summary ........................................... 67
2.2 Accounts Payable and Accounts Receivable ...................... 68
   2.2.1 Maintenance Transactions ........................................ 68
   2.2.2 Data Enhancements .................................................. 68
   2.2.3 Screen Layout Enhancements ..................................... 71
2.3 Accounts Receivable (Customers) ..................................... 72
   2.3.1 Define Your Own Subscreen ...................................... 72
   2.3.2 Define Tabstrip Layout (Customer Screen Group) ......... 74
   2.3.3 Activating a Screen Group via a BAdI Implementation ... 76
   2.3.4 Linking Your Own Subscreen ..................................... 79
   2.3.5 Making the Screen Field Transaction Mode Aware and
         Updatable ...................................................................... 81
   2.3.6 Calling Moments of BAdI Methods ............................. 85
   2.3.7 GUI Status Enhancement with Open FI (BTE) .............. 87
   2.3.8 Other Open FI (BTE) Events ..................................... 90
   2.3.9 Function Module Exits ............................................. 91
2.4 Customer Credit Management Data and Screen Enhancement ... 91
   2.4.1 GUI Status Enhancement .......................................... 92
   2.4.2 Data Enhancement .................................................... 93
   2.4.3 Status Screen Enhancement ....................................... 93
   2.4.4 Defining and Activating Partner Products in
         Transaction FIBF ....................................................... 96
   2.4.5 Setting External Partner Functions ............................ 97
   2.4.6 Further GUI Status Enhancement with Table T061V ..... 99
   2.4.7 Additional Credit Management Data User Exits .......... 102
2.5 Accounts Payable (Vendors) ........................................... 102
   2.5.1 Screen and GUI Status Enhancement with Function
         Group FARI ............................................................. 103
   2.5.2 BAdI Definitions .................................................... 106
   2.5.3 Business Transaction Events ..................................... 109
   2.5.4 Function Module Exits ............................................. 109
2.6 Summary ........................................................................ 110

3 Posting to Accounting .......................................................... 111
   3.1 The Technical Structure of an Accounting Document .......... 111
      3.1.1 The Header .......................................................... 112
[contents]

3.1.2 Items .............................................................................. 113
3.1.3 Parked Document Tables ................................................ 115
3.1.4 Secondary Indices .......................................................... 116
3.1.5 Total Tables ................................................................... 117
3.2 Core Program Modules of Accounting ......................................................... 121
  3.2.1 Screen Enhancement of Accounting Posting Transactions ......................................................... 121
  3.2.2 Screen Enhancement of General Ledger Posting Enjoy Transactions with BAdI ......................................... 127
  3.2.3 Screen Enhancement of Customer or Vendor Enjoy Transactions with BAdI ................................................... 130
3.3 Accounting Document Data Enhancement ......................................................... 135
3.4 Data Processing Enhancements during Dialog Processing .......................... 137
  3.4.1 Data Processing BTEs ..................................................... 137
  3.4.2 BTE Processes ................................................................. 138
  3.4.3 BAdI .............................................................................. 139
  3.4.4 Substitutions and Validations ......................................... 140
3.5 Data Processing Enhancements during Document Saving .......................... 143
  3.5.1 BTE Events ..................................................................... 144
  3.5.2 BTE Processes ................................................................. 145
  3.5.3 BAdls ............................................................................. 146
3.6 SAP Internal Techniques for Processing Accounting Data Flow (RWIN) ......................................................... 146
  3.6.1 RWIN Summary ............................................................. 148
3.7 Differences in Data Processing between Dialog Transactions and Program Functions ......................................................... 148
  3.7.1 Additional BAdl AC_DOCUMENT ................................... 149
  3.7.2 BTEs That Are Not Called ............................................... 149
  3.7.3 Ending BTE 00001050 (POST DOCUMENT: Accounting Interface) ......................................................... 149
3.8 Summary .................................................................................... 149

4 Enhancements in Reports ................................................................. 151
  4.1 Technical Architecture of the Line-Item Report ............................... 151
    4.1.1 Header and Footer Output Enhancement ............................................ 153
    4.1.2 Menu Enhancement with BTE Events ............................................... 156
    4.1.3 Menu Enhancement with BAdI ..................................................... 157
## Contents

4.1.4 Output Layout Enhancement ............................................. 161

4.2 New SAP General Ledger Account Line-Item Report

  4.2.1 Header and Footer Output Enhancement ..................... 164
  4.2.2 Extended Authorization Check ................................. 164
  4.2.3 Menu Enhancement .............................................. 164
  4.2.4 Enhancing the Output Layout ................................. 165

4.3 Summary .......................................................................... 166

5 Inbound Scenarios in Financial Accounting ......................... 167

  5.1 Master Data Migration and Distribution ...................... 167
    5.1.1 Batch Input ..................................................... 167
    5.1.2 HR Master Data .............................................. 177
    5.1.3 ALE/IDoc ..................................................... 178
  5.2 Postings Inbound Scenarios ......................................... 186
    5.2.1 Batch-Input or Direct Input ............................... 186
    5.2.2 Payroll Results ............................................. 187
    5.2.3 Postings via IDoc .......................................... 188
    5.2.4 Electronic Bank Statement ............................... 189
  5.3 Summary .......................................................................... 195

6 Outbound Scenarios in Financial Accounting ...................... 197

  6.1 Master Data Distribution ............................................. 197
    6.1.1 Batch Input ..................................................... 197
    6.1.2 ALE/IDoc tools .............................................. 198
  6.2 Dunning ............................................................................ 203
    6.2.1 BTEs in Transaction F150 .................................. 203
    6.2.2 BTEs during the Dunning Run ......................... 207
    6.2.3 Dunning Summary ............................................ 216
  6.3 Payment Program .......................................................... 216
    6.3.1 User Exits in Transaction F110 ......................... 217
    6.3.2 User Exits in Payment Program SAPF110S ............. 219
  6.4 Summary .......................................................................... 224
## Contents

### 7 Workflow as a User Exit ........................................................... 227

#### 7.1 Workflow Events: Linking System Actions with External Applications .............................................................. 228

- 7.1.1 Event Handling ................................................................................ 228
- 7.1.2 Event Creation Options .............................................................. 231
- 7.1.3 Application Development Implications ................................. 231

#### 7.2 Practical Example ........................................................................ 232

- 7.2.1 Prerequisites ................................................................................. 232
- 7.2.2 Workflow-Enabled Class ........................................................... 233
- 7.2.3 Standard Task .............................................................................. 236
- 7.2.4 Event Creation ............................................................................ 240
- 7.2.5 Now Test! ................................................................................. 241

#### 7.3 Summary .................................................................................... 243

The Author ............................................................................................... 245
Index ......................................................................................................... 247
As today's corporate ERP system landscape becomes more and more distributed, you have to be prepared for different kinds of data that can flow to and from external systems. With this in mind, the focus of this chapter is inbound scenarios in Financial Accounting.

5 Inbound Scenarios in Financial Accounting

In this chapter, we consider data processing scenarios when the SAP system receives accounting data from external systems. This can be master data from legacy systems or posting data from, for example, an external payroll system. This chapter describes how you can intervene in this process using various user exits.

5.1 Master Data Migration and Distribution

There could be no SAP ERP implementation project without an initial data migration procedure. Imagine how painful it would be if a company started its trading activity by implementing SAP ERP and then entered its existing customers and vendors one by one. As a rule, the moment a company implements SAP ERP, the customer/vendor list (which is in some other legacy system) has to be prepared. There are also scenarios in which accounting master data are loaded from external systems on a regular basis.

In the following subsections, we’ll discuss several ways to load master data into an SAP system and how to seamlessly penetrate the standard data flow to address specific requirements.

5.1.1 Batch Input

If you are familiar with the SAP Legacy System Migration Workbench (LSMW) and have completed data migration projects, you probably recognize these standard SAP programs for the mass uploading of customer and vendor master records: reports RFBIDE00 and RFBIKR00.
Both reports have the same selection screen as shown in Figure 5.1. Input data for the report must be presented as a flat file located on the application server.

**Note**

You can also pass a logical file name into the report by passing it through invisible parameter LDS_NAME, which can be used in a SUBMIT statement. In this case, the value of the visible file path name parameter is ignored.

![Selection Screen of Report RFBIDE00](image)

Figure 5.1 Selection Screen of Report RFBIDE00

By default, the maximum length of an input file line is 2,000 characters—this is the length of dictionary structure BDIFIBIWA. If your input file has longer lines, you can extend structure BDIFIBIWA by using customer include structure CI_BDIFIBIWA.

Keep in mind, however, that structure BDIFIBIWA only defines the length of an input file line, whereas the actual structure of the data being processed is defined according to the first 31 characters of the line (see the structure shown in Figure 5.2).

The first character of each file line is a record type, which can take one of three values: 0, 1, or 2. Record type 0 marks the beginning of a session, record type 1 is the beginning of one customer (or vendor) data for one transaction code, and record type 2 is a data record. The next 30 characters of a file line contain a dictionary structure name. For record type 0, the structure name is always BGR00; for record type 1, the structure name is always BKN00 for customers and BLF00 for vendors.
In the record with structure BGR00, you can denote the transaction code that will be used to process the data. The record with structure BKN00 contains the customer number and corresponding organizational assignment, such as company code, sales market data, credit control area, and so on. In the record with structure BLF00, the data contains information for the vendor number, company code, purchasing organization, and so on.

File lines with record type 2 can contain standard and nonstandard structures. Standard batch-input structures mainly comply with the following naming convention: character B followed by one of the master data table names. For example, BKNA1 is a batch-input structure for Table KNA1, BLFA1 is the batch-input counterpart for LFA1, and so on.

Note
The full list of all standard batch-input structures and supported transactions can be found in SAP online help for reports RFBIDE00 and RFBIKRO0.

In the next subsection, you’ll see learn to extend data and amend its processing using BAdIs. You’ll also see a step-by-step example of loading extended data with a standard SAP program.

Data Enhancement
You can enhance batch-input data either by defining your own fields in the corresponding customer include, which you can find in all standard batch-input structures (e.g., CI_BKNA1 in BKNA1), or by defining your own data structures.

If you choose the second option, follow the same conventions found in the standard structure:

- The first two fields of the customer include should be the same as in the standard structure (STYPE and TABNAME).
- All fields must be characters (no numbers).
Note
To make the customer-defined batch-input structure available in SAP LSMW, you must insert a corresponding entry in the customizing table SXDA2.

Using BAdIs
If your custom-defined fields are part of an additional screen layout (see Chapter 2, Master Data Enhancements), then you have to apply user exits to make the system process additional data in customer or vendor loading reports.

Customer loading report RFBIDE00 uses the following BAdI definitions and methods:

- **Definition:** CUSTOMER_ADD_DATA
  - Method CHECK_ADD_ON_ACTIVE is called in the initialization phase of the report. Other BAdI methods are called only if at least one add-on is active.

- **Definition:** CUSTOMER_ADD_DATA_BI
  - Method CHECK_DATA_ROW is called for any nonstandard file line with record type 2 and an unknown structure name. The method can be used to check the input contents for nonstandard structures.
  - Method FILL_FT_TABLE_USING_DATA_ROWS is called at the end of transactions processing (only for Transactions XD01 and XD02). The method can be used to amend or extend generated batch-input screens and field sequences to incorporate add-on screens and fields.

Vendor loading report RFBIKR00 uses the following BAdI definitions: VENDOR_ADD_DATA and VENDOR_ADD_DATA_BI. Method names and their purposes are the same as in report RFBIDE00; and logical method FILL_FT_TABLE_USING_DATA_ROWS is only called for Transactions XK01 and XK02.

Example
To illustrate the enhancement usage in our IDES system, let's incorporate the example from Chapter 2, where we enhanced customer master data, into the standard loading program, RFBIDE00. We extended the company code view of the
customer master data by an additional field: Custom Account Class (with technical name KNB1-ZZCUST_CLASS).

First, we extended the dictionary structure (BKNB1) by defining the customer include (CI_BKNB1). As a result, the BKNB1 definition in Transaction SE11 should look like Figure 5.3.

![Extended BKNB1 Dictionary Structure](image)

Figure 5.3  Extended BKNB1 Dictionary Structure

When preparing the example for Chapter 2, we implemented BAdI CUSTOMER_ADD_DATA. Now we need to use BAdI definition CUSTOMER_ADD_DATA_BI. Because we haven’t created our own batch-input structure, but extended a standard structure instead, we don’t need to implement the CHECK_DATA_ROW method. We do need to code an addition to the screen and field sequence, which will save our data into the customer master record. To do this, we need to examine how the screen sequence might look by using an old batch-input recording, which can be found in Transaction SHDB.
We record the following actions of Transaction XD02 with the following steps:

1. Enter the customer number and company code.
2. Select the enhanced screen layout (defined in Chapter 2).
3. Change the value in the CustAccClass field (no matter from which to which; we just need a value change).
4. Save.

Figure 5.4 shows the combined sequence of screenshots of these steps.

Figure 5.4  Recorded Screen Sequence of Transaction XD02
The result of the recording is shown in Figure 5.5.

Figure 5.5  The Recording of Transaction XD02

As you analyze the recording, you see that on the starting data screen SAPMFD02/200, we executed function code AO05, which has taken us into the enhanced screen layout. There we entered a value of 3 into the field KNB1-ZZCUST_CLASS and clicked Save (function code UPDA).

Now we are ready to implement the code of method FILL_FT_TABLE_USING_DATA_ROWS.
The Interface of Method FILL_FT_TABLE_USING_DATA_ROWS

Figure 5.6 shows the interface of method FILL_FT_TABLE_USING_DATA_ROWS. You can see that we have current BKN00 data (with customer number and other organizational assignment data) as input parameter I_BKN00; we also have all file lines related to the current transaction in input parameter IT_DATA_ROWS. Finally, we have one export table typed parameter, ET_FT, which we will amend according to our logic. ET_FT has line type of BOCDATA structure, which is a well-known structure used in batch-input statement CALL TRANSACTION USING.

The algorithm should do the following:

- Find the first entry of structure BKNB1 in the file data.
- Insert function code AO05 into the previous screen: BDC data.
- Start a new screen in BDC data.
- Set new field values according to BKNB1 contents that were found.

Always keep in mind that there can be other active BADI implementations, so you shouldn't include any function codes in the batch input because this can end the transaction. In our example, we don't insert the function code UPDA, which is seen in our sample recording (refer back to Figure 5.5).

Listing 5.1 shows the source code of our method implementation.

```
METHOD if_ex_customer_add_data_bi~fill_ft_table_using_data_rows.
  FIELD-SYMBOLS: <wa> TYPE bknbl.

  DATA: ft TYPE bocdata.
```
LOOP AT it_data_rows ASSIGNING <wa> CASTING.
   CHECK <wa>-styp = ’2′ AND <wa>-tbnam = ’BKNB1’.

*  Insert function code to select Enhanced screen layout
*  This will be added to the last processed screen in BDC data
   CLEAR ft.
   ft-fnam = ’BDC_OKCODE’.
   ft-fval = ’=AO05’.
   APPEND ft TO et_ft.

*  Start new screen
   CLEAR ft.
   ft-program = ’SAPMF02D’.
   ft-dynpro = ’4000’.
   ft-dynbegin = ’X’.
   APPEND ft TO et_ft.

*  Enter field value on the custom defined screen
   CLEAR ft.
   ft-fnam = ’KNB1-ZZCUST_CLASS’.
   ft-fval = <wa>-zzcust_class.
   APPEND ft TO et_ft.

EXIT.
ENDLOOP.
ENDMETHOD.

Listing 5.1 Method FILL_FT_TABLE_USING_DATA_ROWS Source

After activating the BAdI implementation, we can now test the new fields with a small SAP LSMW project. The goal of this project is to update field KNB1-ZZCUST_CLASS using the batch-input loading program RFBIDE00. After defining the appropriate target object and source structure, you can see in the SAP LSMW field-mapping step that our field is included in the target structure (see Figure 5.7). Note that all uninitialized fields are turned off to make the view more compact.
The **CREATE BATCH INPUT SESSION** step in the SAP LSMW project is actually a call of the program RFBIDE00. We tested it with only one record in the input file to update customer T-L63A02 in company code 1000. Now change the **CUSTACCCLASS** field to 3. After generating the batch-input session, we can inspect it in Transaction SM35. Figure 5.8 shows the screen list of the session with an opened field value list. Our added field is in its place.

**Figure 5.7** LSMW Field Mapping View for the Customer Master
5.1.2 HR Master Data

In some HR payroll instances, an employee has his own HR master record, which generates a corresponding vendor master record or customer master record for that employee in the financials department of the company. From the formal accounting point of view, when the company pays the salary to that employee, he should be treated as a company vendor because that employee sells his services to the company (in the form of an everyday job). If HR Payroll and FI are installed as separate systems, you must set up a task of regularly distributing HR employee data into an FI system to form vendor or customer master records.
In brief, the standard process of HR data distribution, which is based on ALE (application link enabling) technology, looks as follows:

1. Several structures of employee data (called *infotypes*) from the external HR system are copied into the FI system, in the form of an IDoc (depending on the HR system version, it can be an IDoc type from `HRMD_A01` to `HRMD_A07`).

2. The receiving FI system regularly runs report RPRAPA00, which prepares the locally available HR data for loading with the standard report RFBIKR00.

3. Inside report RPRAPA00, a BAdI definition `BADI_EXITS_RPRAPA00` is used to intercept the standard logic when preparing a data file for the following run of the report RFBIKR00. The list of available BAdI methods is shown in Table 5.1.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SET_VALUES_FOR_BLFBW</td>
<td>Exit for BLFBW: Vendor master, withholding tax types</td>
</tr>
<tr>
<td>SET_VALUES_FOR_BLF00</td>
<td>Exit for BLF00: Vendor master</td>
</tr>
<tr>
<td>SET_VALUES_FOR_BLFA1</td>
<td>Exit for BLFA1: Vendor master, general data part 1</td>
</tr>
<tr>
<td>SET_VALUES_FOR_BLFBK</td>
<td>Exit for BLFBK: Vendor master, bank details</td>
</tr>
<tr>
<td>SET_VALUES_FOR_BLFB1</td>
<td>Exit for BLFB1: Vendor master, company code data</td>
</tr>
<tr>
<td>SET_VALUES_FOR_BLFB5</td>
<td>Exit for BLFB5: Vendor master, dunning data</td>
</tr>
<tr>
<td>SET_VALUES_FOR_BGR00</td>
<td>Exit for BGR00: Batch-input structure for session data</td>
</tr>
</tbody>
</table>

Table 5.1 Interface Methods of the BAdI Definition `BADI_EXITS_RPRAPA00`

Each method has an employee number (PERNR) as an input parameter and a respective batch-input structure as a changing parameter. The structure name is clearly shown by the method name.

Because report RPRAPA00 works on the local HR data, you can use standard HR functionality to access employee infotypes. All the BAdI methods are called in the end of each employee number processing.

5.1.3 ALE/IDoc

The batch-input data loading techniques discussed earlier are based on a file as a data carrier. This is a somewhat outdated technology, and while it is robust and
stable, it's less flexible and less secure compared to ALE/IDoc technology. IDoc processing logic is completely separated from the data transferring media, which is much more suitable to the modern distributed environments with its variety of data transferring protocols. In essence, ALE/IDoc technology is more welcome in modern integration projects involving B2B (business to business), A2A (application to application), and mobile scenarios.

When it comes to making a decision on what type of technology to employ in an integrating project of almost any nature, we recommend choosing IDocs over files. ALE/IDoc technology is highly configurable, and depending on corporate-specific requirements, you can completely intercept the IDoc processing of any individual type.

### The Structure of an IDoc in a Nutshell

The structure of an IDoc is identified by its basic type, which is an ordered set of segments. For simplicity, the notion of an IDoc segment can be treated as an equivalent of the dictionary structure. Basic type defines not only a simple order of its segments but also their hierarchy relations, cardinality, and necessity. In other words, the basic type defines the syntax of IDoc, which is controlled by the runtime ALE system layer. The IDoc basic type structure can be displayed using Transaction WE30.

For the sake of simplicity, we can also say that a pair of objects—logical message code and basic type—together define IDoc processing logic via assignment to a specific ABAP function module, workflow template, or task. These assignments are stored in configuration table EDIFCT, which is accessible via Transaction WE57.

SAP delivers the following logical messages for master data distribution via ALE: CREMAS and CRECOR for vendors, and DEBMAS and DEBCOR for customers. Figure 5.9 shows the IDoc processing module configuration for customer-related messages and IDoc types.

If you look into the default IDoc configuration table EDIFCT (via Transaction WE57), you can see that standard processing logic for inbound IDoc transferring customer and vendor master data is hidden in two function modules: IDOC_INPUT_DEBITOR and IDOC_INPUT_CREDITOR. These function modules are assumed to process IDoc basic types from CREMAS01 to CREMAS05, and from DEBMAS01 to DEBMAS06, CRECOR01, and DEBCOR01. In this notation, the numeric suffix is the version of the IDoc structure.
Both function modules work the same way. They first analyze the system type; if it's an ERP system, the function modules call an ERP-specific function: `ERP_IDOC_INPUT_CREDITOR` for a vendor and `ERP_IDOC_INPUT_DEBITOR` for a customer. There is also a function call for a standalone HR system, but it's quite simple. Because HR doesn't need any advanced customer or vendor master data manipulations, you'll find just a direct update of the corresponding tables.

The main secret of standard IDoc processing logic is that it updates or creates individual master record by means of batch input. If you dive into the source code of `ERP_IDOC_INPUT_DEBITOR` or `ERP_IDOC_INPUT_CREDITOR`, you'll find the corresponding `CALL TRANSACTION` statement. In a way, they repeat the logic of reports RFBIDE00 and RFBIKR00; but instead of a flat file, these functions process IDocs, and each segment can be treated as an equivalent of a file line. You can also see that

Figure 5.9  The Contents of Table EDIFCT
after processing IDoc segments, the function gathers information into an internal table of structure **BDIFIBIWA**.

**Note**

In IDoc processing, SAP provides calling moments for the same BAdI definition as in RFBIDE00 and RFBIKR00.

Next, we’ll discuss working with IDoc data structures—segments—and how you can affect the processing logic in standard SAP functions.

**Working with Segments**

The structure of the IDoc type you are planning to process can be displayed in Transaction WE30. Figure 5.10 shows the structure of IDoc basic type **CREMAS05**. As you can see, there are three levels of segment hierarchy.

![Figure 5.10 The Structure of IDoc Type CREMAS05](image-url)
By double-clicking on an arbitrary segment name, you can drill down to the segment editor where you can see the list of segment fields. You can see an example of segment structure in the segment editor in Figure 5.11.

![Figure 5.11 The Structure of the Segment E1LFA1M](image)

When you develop a brand new segment, the final point of the development is the act of releasing the segment. At the moment of release, the system generates a dictionary structure with the same name and all of the segment's fields, which means that the segment can be used officially. All standard segments also have a dictionary structure of the same name. So if an IDoc type defined in your system contains a segment E1LFA1M, you can declare a variable in your program of the type E1LFA1M.

IDoc has a single primary key field—its 16-digit number. We recommend accessing an individual IDoc by the standard function module IDOC_READ_COMPLETELY. Besides the control data (which are outside of our current discussion), the function returns all of the IDoc segments in the form of an internal table of structure, EDIDD.
Each record contains exactly one segment; the segment's name is stored in field `SEGNAM`, while segment data are located in an unstructured field, `SDATA`. An example of a code snippet for IDoc segment processing is provided in Listing 5.2.

```
DATA: lt_edidd TYPE TABLE OF edidd,
      IDoc_number TYPE edidc-docnum,
      ls_ellfalm_segment TYPE e1lfalm,
      ls_ellfb1m_segment TYPE e1lfb1m.

FIELD-SYMBOLS: <edidd> TYPE edidd.

CALL FUNCTION 'IDOC_READ_COMPLETELY'
  EXPORTING
      document_number = IDoc_number
  TABLES
      int_edidd = lt_edidd
  EXCEPTIONS
     OTHERS = 3.

IF sy-subrc <> 0.
  MESSAGE ID sy-msgid TYPE sy-msgty NUMBER sy-msgno
    WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4.
ENDIF.

LOOP AT lt_edidd ASSIGNING <edidd>.
  CASE <edidd>-segnam.
    WHEN 'E1LFA1M'.
      ls_ellfalm_segment = <edidd>-sdata.
      * Processing...
    WHEN 'E1LFB1M'.
      ls_ellfb1m_segment = <edidd>-sdata.
      * Processing...
    ......
    WHEN OTHERS.
      * Processing non-standard segments...
  END_CASE.
ENDLOOP.
```

Listing 5.2 IDoc Segment Processing Code
Note that you can freely use direct assignment between unstructured field `EDIDD-SDATA` and the structured field of the segment despite the Unicode. This is possible because IDoc segment structure contains only character fields; `EDIDD-SDATA` is character typed as well.

**Available BAdIs in Customer Data IDoc Processing**

SAP standard batch-input program RFBIDE00 is called from within the function module `ERP_IDOC_INPUT_DEBITOR`. First, it calls method `CHECK_ADD_ON_ACTIVE` of BAdI definition `CUSTOMER_ADD_DATA`. All other methods of the BAdI definition `CUSTOMER_ADD_DATA_BI` are only called if there is at least one active add-on.

During the IDoc processing, function `ERP_IDOC_INPUT_DEBITOR` invokes the following methods of the BAdI definition `CUSTOMER_ADD_DATA_BI`:

- **PASS_NON_STANDARD_SEGMENT**
  This method is called when the system encounters an unknown segment during the main loop of IDoc segments processing. This call allows you to convert a nonstandard segment into an internal structure for later processing. The segment name and segment data are passed to the method as import parameters.

- **MODIFY_BI_STRUCT_FROM_STD_SEG**
  This method is called after fulfilling all standard processing for each standard segment. The method uses the segment name and segment data as import parameters, and one changing parameter with an already known structure, `BDIFIBIWA`. By the moment of the call, structure `BDIFIBIWA` is filled with standard values, and you can change it according to your requirements.

- **FILL_BI_TABLE_WITH_OWN_SEGMENT**
  This method is called when all standard batch-input data are saved into the internal table of structure `BDIFIBIWA`. This method has a changing table parameter with this structure and an import parameter of dictionary structure `CUSTOMER_ORG_DATA`. When this method is called, you should process the data that were prepared earlier and saved by the `PASS_NON_STANDARD_SEGMENT` method.

- **CHECK_DATA_ROW**
  When all segments are processed and all of the data gathered into the batch-input table of structure `BDIFIBIWA`, the system checks the data before starting the batch input. This method is called for each line of batch-input data if it contains the name of a nonstandard structure. The method has import parameter of structure `BDIFIBIWA` and a flag parameter for passing the data check status ("X")
for success and a blank space for failure). If some of the data have not passed
the check, the method can return an error message through the corresponding
export parameters.

- **FILL_FT_TABLE_USING_DATA_ROWS**
  This method is called just before calling the transaction in batch-input mode.
  It allows the user to make final alterations into the batch-input screen and field
  value sequence. Note that this method is only called if the transaction to be
called is either XD01 or XD02. This method has a changing table parameter
typed with structure **BDCDATA**.

**Available BAdIs in Vendor Data IDoc Processing**

Function module **ERP_IDOC_INPUT_CREDITOR** works with BAdIs in a slightly differ-
ent way: It calls BAdI **VENDOR_ADD_DATA** and method **CHECK_ADD_ON_ACTIVE** after
gathering information into an intermediary internal table of structure **BDIFIBIWA**, instead of at the beginning of IDoc processing.

The following methods of BAdI definition **VENDOR_ADD_DATA_BI** are called during
the processing of the vendor master IDoc:

- **PASS_NON_STANDARD_SEGMENT**
- **MODIFY_BI_STRUCT_FROM_STD_SEG**
- **FILL_BI_TABLE_WITH_OWN_SEGMENT**
- **CHECK_DATA_ROW**
- **FILL_FT_TABLE_USING_DATA_ROWS**

**Enhancement Spots**

Function group **VV02** has two entries of enhancement spot **ES_SAPLVV02CORE**. One
source code plug-in entry of this spot is located in the top include of the function
group and allows you to use your own includes here. Another spot entry can be
found at the beginning of the function code **ERP_IDOC_INPUT_DEBITOR**. On the vendor
side, there is an enhancement spot—**ES_SAPLKD02**—with the same functionality.

**Function Module Exits**

There are also components of old-styled function module exit **VSV00001**, which
you can examine in Transaction SMOD. Customer function **EXIT_SAPLKD02_001** is
called after the vendor data IDoc is completely processed and allows you to save
additional data in the database. Customer function EXIT_SAPLVV02_001 has the same purpose; it is called after processing the customer data IDoc.

5.2 Postings Inbound Scenarios

Now let's examine how accounting document data can come from the external world and what we can do with it.

5.2.1 Batch-Input or Direct Input

As with master data, an initial stage of an SAP ERP implementation project virtually always requires loading initial accounting transaction data. A traditional tool for this activity is the standard SAP report RFBIBL00. Input data for the report are provided in the form of a flat file located on the application server. The report is suitable for use with SAP LSMW, which effectively hides all the file preparation issues.

Internally, the report uses function modules of group FIPI, which are listed in Table 5.2.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>POSTING_INTERFACE_CLEARING</td>
<td>Post with clearing (FB05) using internal posting interface.</td>
</tr>
<tr>
<td>POSTING_INTERFACE_DOCUMENT</td>
<td>Post document using the internal posting interface.</td>
</tr>
<tr>
<td>POSTING_INTERFACE_END</td>
<td>The ending function of the group. Should be called in the end of the process.</td>
</tr>
<tr>
<td>POSTING_INTERFACE_RESET_CLEAR</td>
<td>Reset clearing via posting interface.</td>
</tr>
<tr>
<td>POSTING_INTERFACE_REVERSE_DOC</td>
<td>Cancel document via posting interface.</td>
</tr>
<tr>
<td>POSTING_INTERFACE_START</td>
<td>Initial information for internal accounting interface.</td>
</tr>
</tbody>
</table>

Table 5.2 FIPI Function Group Modules

These functions actually make postings through batch input by generating sessions or calling a transaction directly. The function modules also have detailed system documentation. Unfortunately, report RFBIBL00 does not contain a user-exit call, although you can rely on the user exits available inside the transactions that are called during processing.
5.2.2 Payroll Results

Note that the payroll result posting interface is fully equipped with specific user exits. However, it's worth seeing the overall process outline so that you can understand where and when the process should (or should not) be intercepted, depending on your business requirements.

If the company has SAP HR Payroll implemented, then in every payroll period (weekly or monthly), there must be an interface running that posts payroll results to the Financials department. SAP recommends implementing HR as a separate system to improve data security because payroll data are among the most sensitive corporate data.

If you are implementing a payroll results posting from SAP HR into SAP FI, then in the end, the posting will be performed with the same tools.

The whole process of HR payroll posting looks like this:

1. The responsible person in HR creates a payroll posting run with report RPCIPE00. The report creates a preliminary posting document stored in Tables PPDHD, PPDIT, and others.

2. Someone then checks and approves all of the resulting posting documents (they are not accounting documents) by editing particular payroll runs with Transaction PCP0.

3. Finally, someone runs report RPCIPPO0 to transfer values into accounting.

The last step can be performed either via ALE/IDoc interfaces (if HR Payroll works as a separate system), or locally—by direct call of an accounting BAPI. By default, all of HR Payroll IDocs are processed in the receiving system by the same BAPI. Let's trace the chain.

The HR system generates three types of postings:

- **Employee expenses**
  For example, travel and accommodation when on a business trip.

- **Employee vendor items**
  For example, an employee can be treated as a corporate vendor or service provider to justify salary payment; thus the document is generated as an Account Payables item.
Employee customer items

If an employee has debts that are not settled, he might appear in the role of a corporate customer; the document is generated as an Account Receivables item.

If an HR Payroll component is implemented as a separate system, then it generates three types of IDocs: ACC_EMPLOYEE_PAY02, ACC_EMPLOYEE_REC02, and ACC_EMPLOYEE_EXP02. In the receiving system, these IDocs are linked by default via the ALE/BAPI-generated interface to the following function modules:

- IDOC_INPUT_ACC_EMPLOYEE_EXP for employee expenses
- IDOC_INPUT_ACC_EMPLOYEE_PAY for employee payments
- IDOC_INPUT_ACC_EMPLOYEE_REC for employee debts

The accounting documents are generated with BAPI calls:

- BAPI_ACC_EMPLOYEE_EXP_POST for employee expenses
- BAPI_ACC_EMPLOYEE_PAY_POST for employee payments
- BAPI_ACC_EMPLOYEE_REC_POST for employee debts

Finally, each of the BAPIs call function modules AC_DOCUMENT_CREATE and AC_DOCUMENT_POST as a low-level accounting interface utility. Thus, you can employ any user exit (BAdI or BTE) appearing in the AC_DOCUMENT_CREATE function module (see Chapter 3, Posting to Accounting), including substitutions and validations.

At the call point of a user exit during the document generation, you can distinguish SAP standard HR Payroll postings from any others by the contents of the field BKPF-GLVOR:

- HRP1 for employee expenses
- HRP3 for employee payments (Account Payables)
- HRP2 for employee debts (Accounts Receivable)

5.2.3 Postings via IDoc

The SAP system delivers dozens of IDoc types to be used for posting different flavors of accounting documents: direct posting to a general ledger account, posting of incoming vendor invoice, and so on. You can find corresponding IDoc types in Transaction WE30 (Executing the Search Help with Mask ACC*). However, if you look into the processing function modules, you’ll notice that they aren’t equipped with user exits. If you thoroughly trace the chain of calls, you’ll see that this chain
is ended at the same function modules mentioned in the previous section: AC_DOCUMENT_CREATE and AC_DOCUMENT_POST. Thus, you should rely on already-known user exits discussed in Chapter 3.

### 5.2.4 Electronic Bank Statement

The process of loading a bank statement file consists of two phases: importing the bank statement file in Transaction FF_5, and posting the bank statement through Transaction FEBP.

**Importing the Bank Statement File**

Loading program RFEBKA00, which is linked to Transaction FF_5, parses incoming bank files according to a selected format, such as Multicash or SWIFT MT940, which are widely used in bank communication. Each individual file format is parsed in an external program, although the code in report RFEBKA00 that is responsible for choosing the format parsing program is quite static; there is just a `CASE` statement with no configuration.

However, if you look into the source code of format SWIFT MT940 parsing routine program RFEKA400, you can discover an old-fashioned user exit, `EXIT_RFEKA400_001`, belonging to function module exit `FEB00004`. The enhancement can be used for preprocessing raw file data, which is passed to the user exit in the form of a table parameter with a length of 512 unstructured lines. Listing 5.3 shows the interface of the user exit.

```plaintext
FUNCTION EXIT_RFEKA400_001.

*"--------------------------------------------------------------------
*"| Lokale Schnittstelle:
*"| TABLES
*"|   T_RAW_DATA STRUCTURE RAW_DATA
*"|   EXCEPTIONS
*"|   ERROR_OCCURED
*"--------------------------------------------------------------------

INCLUDE ZXF01U06.

ENDFUNCTION.

Listing 5.3 EXIT_RFEKA400_001 Interface
```
You can also see that `EXIT_RFEEKA400_001` has one exception, which signals the host program to stop processing the file any further.

Report RFEBKA00 gathers parsed data into the following bank statement database tables:

- FEBKO (electronic bank statement header records)
- FEBEP (electronic bank statement line items)
- FEBRE (reference record for electronic bank statement line item)

**Posting the Bank Statement**

When you link report RFEBKA30 to Transaction FEBP, it interprets data in bank statement tables and makes an accounting posting. A bank statement is a list of operations of what the bank did with your money on your behalf, such as company payments to vendors, bank charges for its services, interest payments, payments from your customers, and so on. All of these operations should be correctly reflected in the company's financial accounting to make sure that the money flow is consistent and correct.

At the same time, the bank’s statement can use different identification for the same objects presented in your system; also, it’s possible that some valuable data in the context of your SAP ERP system may be omitted in the statement for one reason or another. During the interpretation phase, report RFEBKA30 is trying to fill these gaps automatically, for example, to determine the business partner number for the bank transaction or even more important to determine the clearing reference (e.g., payment against invoice) document numbers.

Report RFEBKA30 actually is only a wrapper for another report, RFEBBU10, which performs the interpretation. The algorithm runs through header-item relation of two tables, FEBKO and FEBEP. For each FEBEP internal loop run, the report calls different user exits that can help discover missing statement data.

Now let’s walk through the available BTEs you can employ during the processing of a bank statement.

**BTE 00002810 and Process 00002820**

First, the system calls BTE 00002810 (you can see its interface in Listing 5.4). The event has a pair of parameters for the header record and for the line item of the bank statement that is being processed. The parameter with suffix `EXT` contains
fields with external data (records that were sent by the bank), whereas suffix INT signifies that this data is internal. As a result of its run, each function module that is subscribed to the 00002810 event must return a registration flag in one of two export parameters: E_REGISTER_AREA_1 or E_REGISTER_AREA_2.

*"--------------------------------------------------------------------
*"  IMPORTING
*"     VALUE(I_FEBKO_EXT) LIKE  FEBKOXT_BF STRUCTURE  FEBKOXT_BF
*"     VALUE(I_FEBEP_EXT) LIKE  FEBEPXT_BF STRUCTURE  FEBEPXT_BF
*"     VALUE(I_FEBKO_INT) LIKE  FEBKOIN_BF STRUCTURE  FEBKOIN_BF
*"     VALUE(I_FEBEP_INT) LIKE  FEBEPIN_BF STRUCTURE  FEBEPIN_BF
*"     VALUE(I_TESTRUN) TYPE  XFLAG OPTIONAL
*"  EXPORTING
*"     VALUE(E_REGISTER_AREA_1) LIKE  BOOLE-BOOLE
*"     VALUE(E_REGISTER_AREA_2) LIKE  BOOLE-BOOLE
*"     VALUE(E_SUPPR_STD_AREA_1) LIKE  BOOLE-BOOLE
*"     VALUE(E_SUPPR_STD_AREA_2) LIKE  BOOLE-BOOLE
*"  TABLES
*"     T_FEBRE STRUCTURE  FEBRE_BF
*"     T_FEBCL STRUCTURE  FEBCL_BF
*"--------------------------------------------------------------------

Listing 5.4 The Interface of BTE 00002810

Note that subscribers to event 00002810 are called from within function FEB_OPEN_ FI_CALL_1. This function allows only one application ID to be registered for each of the two areas. The application ID in the BTE framework is used to distinguish SAP internal and partner application areas. Customer-defined P&S modules and processes can have blank application IDs. Therefore, you should make sure that for this particular line item of the bank statement, your function is the only one registered, or an error will be reported. Another pair of event flag parameters, E_SUPPR_STD_AREA_1 and E_SUPPR_STD_AREA_2, will prevent execution of interpretation algorithm if they are assigned X.

Process 00002820 is called just after the event and only for registered application IDs. You can see the process interface in Listing 5.5. Note that there are export parameters to allow changing values in bank statement headers and items. Note that your changed data will be taken into account only if you assign X to the export parameter E_UPDATE_FEB.
Besides header and item data, you can also fill in clearing data in table parameter T_FEBCL.

Figure 5.12 The Signature of Method CHANGE_DATA of BAdI FIEB_CHANGE_BS_DATA

**BADI Definitions**

Progressing to business transaction events and processes, the system calls BADI definition FIEB_CHANGE_BS_DATA and method CHANGE_DATA. Figure 5.12 shows the
interface (or signature) of the method. Notice that the method has three changing parameters: \texttt{C\_FEBKO} and \texttt{C\_FEBEP} for the header and item of the bank statement, and table parameter \texttt{T\_FEBCL} for clearing data from the statement.

The method can also return error code and error message attributes to be reported in the log and prevent the statement from being processed further.

Another BAdI definition, \texttt{FIEB\_CHANGE\_STATEMENT}, is called after all of the interpretation is executed, and the system has done everything it can. You can see the interface of the BAdI method \texttt{CHANGE\_DATA} in Figure 5.13.

<table>
<thead>
<tr>
<th>Ty</th>
<th>Parameter</th>
<th>Type spec.</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>ID_TESTRUN</td>
<td>TYPE_XFELD</td>
<td>Check box</td>
</tr>
<tr>
<td>02</td>
<td>IT_FEBKO</td>
<td>TYPE_STANDARD_TABLE_OPTIONAL</td>
<td>Payment Notes</td>
</tr>
<tr>
<td>03</td>
<td>IT_FEBEP</td>
<td>TYPE_STANDARD_TABLE</td>
<td>Line Items</td>
</tr>
<tr>
<td>04</td>
<td>IT_FEBCL</td>
<td>TYPE_STANDARD_TABLE_OPTIONAL</td>
<td>Clearing Information</td>
</tr>
<tr>
<td>05</td>
<td>VALUE_FLT_VAL</td>
<td>TYPE_LANDT</td>
<td>Parameter FLT_VAL of Method CHANGE_DATA</td>
</tr>
<tr>
<td>06</td>
<td>ED_BUBRC</td>
<td>TYPE_SY_BUBRC</td>
<td>Return Value, Return Value After KA98 Statements</td>
</tr>
<tr>
<td>07</td>
<td>ED_MSG00</td>
<td>TYPE_SY_MSG00</td>
<td>Messages, Message Class</td>
</tr>
<tr>
<td>08</td>
<td>ED_MSG01</td>
<td>TYPE_SY_MSG01</td>
<td>Messages, Message Type</td>
</tr>
<tr>
<td>09</td>
<td>ED_MSG02</td>
<td>TYPE_SY_MSG02</td>
<td>Messages, Message Number</td>
</tr>
<tr>
<td>10</td>
<td>ED_MSG03</td>
<td>TYPE_SY_MSG03</td>
<td>Messages, Message Variable</td>
</tr>
<tr>
<td>11</td>
<td>ED_MSG04</td>
<td>TYPE_SY_MSG04</td>
<td>Messages, Message Variable</td>
</tr>
<tr>
<td>12</td>
<td>ET_FEBEP</td>
<td>TYPE_STANDARD_TABLE</td>
<td>Changed Line Items</td>
</tr>
<tr>
<td>13</td>
<td>ET_FEBCL</td>
<td>TYPE_STANDARD_TABLE</td>
<td>Changed Clearing Information</td>
</tr>
<tr>
<td>14</td>
<td>ET_DELETE_FEBCL</td>
<td>TYPE_STANDARD_TABLE</td>
<td>Deleted Clearing Information</td>
</tr>
<tr>
<td>15</td>
<td>GS_FEBKO</td>
<td>TYPE_FEBKO</td>
<td>Electronic Bank Statement Header Records</td>
</tr>
</tbody>
</table>

Figure 5.13 The Signature of Method CHANGE\_DATA of the BAdI FIEB\_CHANGE\_STATEMENT

\textit{Customer-Defined Interpretation Algorithm}

After calling BTEs and the first BAdI, the system runs the interpretation proper. Each bank statement item can have its own interpretation algorithm, which is defined by the field \texttt{FEBEP\_INTAG} value. Therefore, the individual item algorithm can be set during a user exit run: either BTE or BAdI.

A full list of interpretation algorithm numbers and descriptions can be found in the \texttt{INTAG\_EB} domain fixed values. \texttt{INTAG\_EB} is numeric 3. It is assumed that all SAP system algorithms belong to the range of \texttt{INTAG} values from 000 to 899, and everything above 900 is a customer-defined interpretation.
To implement a customer-defined interpretation, you have to create a function module with a predefined name structure—Z_FIEB_NNN_ALGORITHM—where NNN is the algorithm number.

This function module must have the interface shown in Listing 5.6.

```
FUNCTION Z_FIEB_901_ALGORITHM.
  "--------------------------------------------------------------------
  "Local Interface:
  "  IMPORTING
  "   REFERENCE(I_NOTE_TO_PAYEE) TYPE STRING
  "   VALUE(I_COUNTRY) TYPE T001-LAND1
  "  TABLES
  "   T_AVIP_IN STRUCTURE AVIP
  "   T_AVIP_OUT STRUCTURE AVIP
  "   T_FILTER1
  "   T_FILTER2
  "--------------------------------------------------------------------
ENDFUNCTION.
```

**Listing 5.6 Sample Interpretation Algorithm Function**

Based on the payment note passed to the function in parameter I_NOTE_TO_PAYEE and document references in T_AVIP_IN, the algorithm is expected to produce reasonable results in table structure T_AVIP_OUT, which has the structure of the payment advice line item. Table structure T_AVIP_OUT is then used to update the clearing reference data for the statement item.

**Function Module Exit**

After the interpretation algorithm and just before the second BAdI call, the system invokes a component (function module) EXIT_RFEBBU10_001 of the old-fashioned function module exit FEB00001. Its interface is shown in Listing 5.7.

```
FUNCTION EXIT_RFEBBU10_001.
  "--------------------------------------------------------------------
  "Lokale Schnittstelle:
  "  IMPORTING
  "   VALUE(I_FEBEP) LIKE FEBEP STRUCTURE FEBEP
  "   VALUE(I_FEBKO) LIKE FEBKO STRUCTURE FEBKO
  "   VALUE(I_TESTRUN) TYPE XFLAG
  "  EXPORTING
  "   VALUE(E_FEBEP) LIKE FEBEP STRUCTURE FEBEP
  "--------------------------------------------------------------------
```

5.3 Summary

In this chapter, we discussed several inbound interfaces of Financial Accounting, which cover some of the general corporate activities. Thanks to the SAP design in all of these scenarios, you can find ways to seamlessly tailor the standard process for specific corporate needs.

In the next chapter, you’ll see what user-exit techniques are available for development in outbound scenarios when the system sends accounting data to external systems.
# Index

## A

- ABAP Objects class
  - `CL_EXITHANDLER`, 32
- Account assignment, 135
- Accounts Payable, 68
- Accounts Receivable, 68
- Account type, 116
- ALV list, 153
- Append structure, 51, 69, 93
- Application code, 25

## B

- BADI, 31
  - `AC_DOCUMENT`, 149
  - `BADI_FDCB_SUBBAS01`, 131
  - `BADI_FDCB_SUBBAS05`, 131
  - `CALL BADI`, 34
  - Classic, 31
  - `CUSTOMER_ADD_DATA`, 72, 76, 85, 86, 90, 170
  - `CUSTOMER_ADD_DATA_BI`, 170, 200
  - `CUSTOMER_ADD_DATA_CS`, 72, 79, 81, 85, 86, 107
  - `FAGL_DERIVE_PSEGMENT`, 140
  - `FAGL_DERIVE_SEGMENT`, 139
  - `FAGL_ITEMS_MENUE01`, 164
  - `FAGL_ITEMS_MENUE02`, 164
  - `FAGL_PERIOD_CHECK`, 139
  - `FI_F110_SCHEDULE_JOB`, 217
  - `FI_HEADER_SUB_1300`, 129
  - `FI_LIMIT_ACCOUNT`, 66
  - `FI_TRANS_DATE_DERIVE`, 139
  - `GET BADI`, 34
  - `GET_INSTANCE`, 37
  - Kernel-based, 31
  - `TR_GET_ACCNT_ASSIGN`, 140
  - `VENDOR_ADD_DATA`, 72, 106, 108
  - `VENDOR_ADD_DATA_BI`, 202
  - `VENDOR_ADD_DATA_CS`, 72, 107, 108

- BADI definition, 84
  - `CUSTOMER_ADD_DATA`, 184
  - `CUSTOMER_ADD_DATA_BI`, 184
  - `FAGL_AUTHORITY_CHECK`, 164
  - `FAGL_ITEMS_MENUE01`, 165
  - `FI_ITEMS_MENUE01`, 157
  - `FI_ITEMS_MENUE02`, 157
  - `VENDOR_ADD_DATA`, 185
  - `VENDOR_ADD_DATA_BI`, 185

- Batch input, 167, 197
- Binding, 239
- Breakpoint, 22, 30
- BSEG
  - DMBTR, 115
- BTE, 24, 28, 124
  - 00001005, 138, 144
  - 00001011, 138
  - 00001020, 144
  - 00001025, 144
  - 00001030, 145
  - 00001050, 149
  - 00001070, 122, 124
  - 00001080, 122, 138

## 247
Index

00001610, 156
00001620, 156
00001630, 162, 165
00001640, 153, 164
00001650, 162, 163, 165
00001703, 208
00001705, 213
00001719, 214
00001751, 203
00001762, 209
00001763, 210
00001764, 212
0002105, 218
0002310, 65
0002810, 190
Sample function module, 30
BTE application code, 25, 29
BTE configuration, 28
BTE customer event, 29
BTE partner event, 28
BTE process, 24, 29
00001020, 213
00001030, 215
00001040, 215
00001050, 211
00001053, 209
00001060, 209
00001061, 210
00001068, 211
00001074, 211
00001076, 211
00001100, 139
00001110, 138
00001120, 145
00001130, 146
00001150, 146
00001170, 146
00001809, 223
00001810, 220, 223
00001811, 224
00001815, 223
00001819, 218
00001820, 219
00001821, 221
00001830, 220
00001831, 222
00002820, 190
Default function name, 24
Multiple subscription, 30
Sample function module, 30
BTE product code, 26
Business Add-In, 31
Business Transaction Event, 24

C
Change document, 69, 87
Chart of accounts, 42
Classic BAdI, 32
Classic ledger, 118
Cluster table, 113
Coding block, 135
BSEG, 135
CI_COBL, 135
Custom defined fields, 135
CUSTOMER_ADD_DATA
CHECK_ADD_ON_ACTIVE, 184
CUSTOMER_ADD_DATA_BI
CHECK_DATA_ROW, 184
FILL_BI_TABLE_WITH_OWN_SEGMENT, 184
MODIFY_BI_STRUCT_FROM_STD_SEG, 184
PASS_NON_STANDARD_SEGMENT, 184
Customer enhancements, 17, 18, 21
Customer function module component, 20
Customer product, 26

D
Data element, 42
Data enhancements, 68
Dialog processing, 137
Domain
List of values, 42
Value table, 42
Dunning, 203
Dunning printout phase, 212
Dunning run, 206, 207
Dynamic assign, 62, 68, 110
Index

E

Electronic bank statement, 189
Enhancement Framework, 34, 35
Enhancement project, 18
Enhancement spot, 35, 185, 200
Enhancement techniques, 17
Enjoy transactions, 121, 124
Event creation, 231
Event handling, 228
Event processing, 231
Export parameters, 163

F

FI Business Framework, 24
Filtered BAdI, 35
Flexible General Ledger, 117, 118, 119, 120, 140, 151
Flow logic, 53, 61, 93
Foreign key, 46
Function group, 103, 148
ATAB, 55, 59
FAGL ITEMS_SELECT, 163
FARI, 103
FI_Items, 153
FIPI, 186
GL_ACCOUNT_MASTER_MAINTAIN, 50, 64
RWCL, 121, 147, 148
Function module, 100
AC_DOCUMENT_CREATE, 148, 188
AC_DOCUMENT_GENERATE, 148
AC_DOCUMENT_POST, 148, 188
BAPI_ACC_EMPLOYEE_EXP_POST, 188
BAPI_ACC_EMPLOYEE_PAY_POST, 188
BAPI_ACC_EMPLOYEE_REC_POST, 188
BF_FUNCTIONS_READ, 30
DATE_TO_PERIOD_CONVERT, 119
ERP_IDOC_INPUT_CREDITOR, 180
ERP_IDOC_INPUT_DEBITOR, 180, 184
EXIT_RFEBBU10_001, 194
EXIT_RFeka400_001, 189
EXIT_SAPLKD02_001, 185
EXIT_SAPLVV01_001, 200
EXIT_SAPLVV02_001, 186
EXIT_SAPMF02D_001, 110
EXIT_SAPMF02H_001, 66
EXIT_SAPMF02K_001, 109
FI_ITEMS_DISPLAY, 163
FI_ITEMS_DISPLAY, 153, 163
FI_PRINT_DUNNING_NOTICE, 215
FI_PRINT_DUNNING_NOTICE_PDF, 215
FI_PRINT_DUNNING_NOTICE_SMARTF, 215
GET_DUNNING_CUSTOMIZING, 215
GL_ACCT_MASTER_MAINTAIN, 40
IDOC_INPUT_ACC_EMPLOYEE_EXP, 188
IDOC_INPUT_ACC_EMPLOYEE_PAY, 188
IDOC_INPUT_ACC_EMPLOYEE_REC, 188
IDOC_INPUT_CREDITOR, 179
IDOC_INPUT_DEBITOR, 179
IDOC_READ_COMPLETELY, 182
MASTERIDOC_CREATE_CREDOR, 202
MASTERIDOC_CREATE_CREDMAS, 202
MASTERIDOC_CREATE_DEBCOR, 200
MASTERIDOC_CREATE_DEBMAS, 200
MASTERIDOC_CREATE_GLCore, 199
MASTERIDOC_CREATE_GLMAST, 199
MASTER_IDOC_DISTRIBUTE, 199
MODX_FUNCTION_ACTIVE_CHECK, 22
OUTBOUND_CALL_00002310_E, 65
PC_FUNCTIONS_READ, 30
RWIN_CHECK_SUBSET, 147
TABSTRIP_INIT, 41
TABSTRIP_LAYOUT_READ, 41, 55, 72
Function module exit
FEB00004, 189
VSV00001, 185
Functional method, 236
Funds Management, 140

G

General ledger, 39
GET BAdI, 37
GUI status enhancement, 121
Index

I

IDoc processing, 184, 185
Implicit enhancement, 38
Import parameters, 163
Interface
  BI_OBJECT, 233
  BI_PERSISTENT, 233, 234
  IF_WORKFLOW, 233

K

Kernel-based BAdI, 34

L

Legacy System Migration Workbench (LSMW), 167
Line-item report, 151
Local Persistent Object Reference (LPOR), 234
Logical database, 117, 152
  DDF, 117
  KDF, 117, 152
  LDF, 152
  SDF, 117, 152
Logical messages, 179
LPOR, 234

M

Maintenance view, 46
  V_T80D, 141
  VWTTYGB01, 142
Maintenance view cluster
  VC_TAMLAY_00, 55, 58
  VC_TAMLAYA_00, 55
  V_T004_B, 60
Master data enhancements, 39
Menu enhancement, 157, 164
Menu exit, 19
Module pool, 107
Module pool SAPMF05A, 121
Multicash, 189

O

Open FI, 24
Output layout enhancement, 161

P

PAI modules, 64, 137
Partner functions, 97
Partner products, 26, 96
Payment method, 217
Payment proposal, 217
Payment run, 219
PBO modules, 63, 64, 82, 85, 137
Program
  RBDSECRE, 202
  RBDSEGLM, 198
  RFBIBL00, 186
  RFBIDE00, 169, 170, 184
  RFBIDE10, 198
  RFBIKR00, 167, 169, 170
  RFBIKR10, 198
  RFEBK00, 189
  RFEEKA00, 189
  RFBISA10, 198
  RGGBS00, 142
  RPCIPE00, 187
  SAMF02D, 68
  SAMF02K, 68
  SAPF110V, 217
  SAPF15052, 207
  SAPLFAGL_ITEMS_DISPLAY, 165
  SAPGL_ACCOUNT_MASTER_START, 40
  SAPLFDCEB, 133
  SAPMFO2C, 92, 93
  SAPMFO2D, 87
  SAPMFO2H, 65, 66
  SAPMFO5A, 121, 147
Publish and subscribe (P&S), 23, 24, 89, 137

R

Record type, 168
Relational Database Management System (RDBMS), 111
Index

Report development, 151
Repository Information System, 113
RW Interface (RWIN), 146, 147
TRWPR, 147

S

SAP Business Workflow, 227
SAP HR Payroll, 187
SAP LSMW, 170
SAP NetWeaver Master Data Management, 197
SAP Smart Form, 214
Secondary indices, 116
Segmental reporting, 140
SEPA, 224
Standard logic, 41
Standard task, 228, 236
Structure
BDIFIBIWA, 168, 181
BGR00, 168
BKN00, 168
BKNA1, 169
BLF00, 168
CUSTOMER_ORG_DATA, 184
GLACCOUNT_CCODE_DATA, 51
INVFO, 135
RF61B, 93
SIBFLPOR, 234
Subroutine, 141, 142
Substitution, 142
SWIFT MT940, 189

T

Table
BKPF, 112
BSAD, 117
BSAK, 117
BSAS, 117
BSEC, 114
BSED, 114
BSEG, 113, 114
BSES, 114
BSET, 114, 115
BSID, 117
BSIK, 117
BSIS, 117
EDIFCT, 179
FAGLPLEXT, 119
FEBEP, 190
FEBKO, 190
FEBRE, 190
GB01, 142
GLT0, 118
KONV, 113
KNA1, 69
KNB1, 69, 86
KNC1, 119
KNKK, 93, 94, 96
KNVV, 86
LFA1, 69
LFB1, 69
LFC1, 119
MHND, 207
MHNK, 207
PPDHD, 187
PPDIT, 187
SKA1, 41, 50
SKB1, 41, 50, 51, 64
SXDA2, 170
T004, 42
T020, 96
T061S, 97, 99, 103
T061V, 99, 103
T881, 119
TAMLAY1, 55
TAMLAY2, 55
TAMLAYA, 55
TAMLAYB, 55
TBE01, 24, 25
TBE11, 26
TBE12, 26
TBE22, 26
TBE23, 26
Technical settings, 44
TPS01, 24, 25
VBKPF, 116
VBSEC, 116
VBSEGA, 116
VBSEG, 116
VBSEGK, 116

251
Index

VBSEGS, 116
VBSET, 116
Tabstrip, 58
Total tables, 117
Transaction
BD12, 199
BD14, 201
BD18, 198
CMOD, 18
FB01, 129
F-02, 121, 142
F-42, 121
F110, 217
F150, 203, 217
FAGLL03, 151, 163
FB01, 142
FB50, 121, 129
FB50, 142
FB60, 121, 124, 130, 142
FB70, 124, 130
FBF1, 129
FBD5, 129
FBL1N, 151
FBL3N, 151, 160
FBL5N, 151
FD32, 92, 93, 98
FD33, 92, 93
FIBF, 25, 65, 96
FS00, 40, 66
FS01, 41
FS02, 41
FS03, 41
FSP0, 40
FSS0, 40, 61, 64
PCP0, 187
PFTC, 236
RBDSECRE, 201
RBDSEDEB, 199
SE11, 47, 50, 113
SE18, 31
SE24, 233

V
Validation, 142
VENDOR_ADD_DATA
CHECK_ADD_ON_ACTIVE, 185
VENDOR_ADD_DATA_BI
CHECK_DATA_ROW, 185
FILL_BI_TABLE_WITHOWN_SEGMENT, 185
FILL_FT_TABLE_USING_DATA_ROWS, 185
MODIFY_BI_STRUCT_FROM_STDSEG, 185
PASS_NON_STANDARD_SEGMENT, 185
Vendor control data, 105
Vendor master enhancements, 103

W
Where-used list, 35
Workflow event, 229
Workflow template, 228