Sabine Maisel

Practical Guide to IDoc Development for SAP®

Galileo Press
Bonn • Boston
Contents at a Glance

1 Introduction ................................................................. 15
2 Generating IDocs .......................................................... 29
3 Test Tools ........................................................................ 51
4 Changes to IDocs ............................................................ 67
5 Confirmations ................................................................. 187
6 Serializing IDocs ............................................................. 193
7 Administration ............................................................... 211
8 IDocs in Conjunction with SAP NetWeaver
   Process Integration ....................................................... 243
# Contents

Preface .............................................................................................................. 11

## 1 Introduction .......................................................................................... 15

1.1 What Is ALE? ................................................................. 15
1.2 BAPIs and IDocs — An Introduction ....................................... 18
  1.2.1 BAPIs ...................................................................... 18
  1.2.2 IDocs ...................................................................... 19
1.3 Differentiation of ALE and EDI ............................................. 25
1.4 Summary ...................................................................... 27

## 2 Generating IDocs .............................................................................. 29

2.1 Standard Methods for the IDoc Generation ......................... 29
  2.1.1 Shared Master Data Tool .................................. 29
  2.1.2 Message Control .............................................. 43
  2.1.3 Special Functions ............................................. 46
2.2 Use of Logical Systems in the Message Control ............. 49
2.3 Summary ...................................................................... 50

## 3 Test Tools ..................................................................................... 51

3.1 Individual IDocs ............................................................ 52
3.2 Testing Processing of Multiple IDocs .................................. 57
  3.2.1 Message Control .............................................. 57
  3.2.2 Sending Ready-for-Dispatch IDocs .................... 58
  3.2.3 Files ...................................................................... 59
3.3 Processing Status Files .................................................. 61
3.4 Summary ...................................................................... 66

## 4 Changes to IDocs ............................................................................. 67

4.1 Customizing ................................................................. 67
  4.1.1 Filtering Using Filter Objects .................. 67

7
# 7 Administration ................................................................. 211

7.1 IDoc Links ....................................................................... 211
7.2 Regular Jobs .................................................................. 216
7.3 Transaction Code Overview ........................................... 217
7.4 Archiving ...................................................................... 224
7.5 Status Conversion ......................................................... 230
7.6 Summary ...................................................................... 242

# 8 IDocs in Conjunction with SAP NetWeaver Process Integration ...................................................... 243

8.1 Conversion of Logical Systems to Business Systems ...... 243
8.2 Conversion of IDoc Partner Roles ................................. 244
8.3 Header Mapping ........................................................... 245
8.4 Handling the Control Record in SAP NetWeaver PI ...... 246
8.5 Updating IDocs Directly in SAP NetWeaver PI .......... 248
8.6 Summary ...................................................................... 248

The Author ........................................................................... 249
Index .................................................................................... 251
In the sending system, it’s necessary to generate IDocs so that they can be sent to the receiving system. How this is done depends on the type of data and the application. This chapter describes the different generation options and their use.

2 Generating IDocs

SAP has provided tools for the generation of IDocs in all locations where they are used in ALE scenarios or classic EDI, and they can usually be activated using the Customizing settings. However, there are different methods of IDoc generation depending on the type of data and the location where the IDoc will be generated. This chapter presents the most common methods of IDoc generation. Usually the IDoc administrator — not the developer — implements the settings required for creating the partner profiles, for instance, so they are outlined only briefly, and more importance is attached to the functional process flow.

2.1 Standard Methods for the IDoc Generation

Initially, you must distinguish between the generation of master data and the generation of transaction data because there are different requirements on the generation process or the generation frequency depending on the type of data. A special tool is available to generate master data, called the Shared Master Data Tool (SMD). Transaction data IDocs are generated via the already-existing message control. There are also some special functions for IDocs that are directly generated in a process.

2.1.1 Shared Master Data Tool

The SMD is a special tool for sending master data via IDoc. Master data is characterized by a relatively long retention period in the system during
which the data is changed rarely. Master data usually consists of multiple views that are not used all the time. You can omit views, including those that contain mandatory fields, because the check of whether all mandatory fields are populated is only carried out if the view is actually used. This enables you to select from the wealth of information that is offered for a specific object and use exactly the data that is actually required within your enterprise.

To distribute data using IDocs, an automated process is desirable that responds to the creation and modification of master data without requiring further user interventions. Also, empty views are not supposed to be transferred.

The SMD takes these requirements into account. The technical implementation of automation and control via views entails that already-existing procedures can be used for both functions. For automation, you revert to recording changes, which is implemented by default; for control via views, you use the option (which originated from batch input processing) to control irrelevant fields using a `NO_DATA` character. Then, the IDocs are regularly generated via background jobs.

Additionally, for almost all objects, you also have the option to explicitly generate IDocs or to request IDocs. You can use this option if waiting for periodically scheduled jobs is impossible.

### Recording Changes

For consistency reasons, changes to the master data are updated independently of the use of ALE in SAP systems. For each individual data element of the tables concerned, SAP has stipulated whether a change should be logged or not. Figure 2.1 shows the `CHANGE DOCUMENT FLAG`, which is activated in this case as an example of BISMT (`Old Material Number`) from the `MARA TABLE`.

For updating the changes, you always call the `CHANGEDOCUMENT_OPEN` function module, which prepares the writing of the change history. Then, all changes to be written are collected, and the process is concluded using the `CHANGEDOCUMENT_CLOSE` function module. Because IDocs are
supposed to be generated wherever changes are updated by default, the CHANGEDOCUMENT_CLOSE function module has an ALE share in addition to its standard function. This enables you to generate change pointers for ALE for the desired message types. In all Unicode-enabled releases, this is done using the CHANGE_POINTERS_CREATE_LONG function module; in old releases, this is done using the CHANGE_POINTERS_CREATE function module.

Characteristics of Data Elements

You also have the CHANGE_POINTERS_CREATE DIRECT function module as a second option to generate change pointers. This function module is called by applications that are not connected to the previously described change management for documents.

In both cases, change pointers are only written if you use the SMD for performance reasons. In the ALE Customizing, you can specify whether this is the case and for which master data you require change pointers. There is a separate Transaction code SALE for the ALE Customizing that takes you directly to the correct position in the menu tree. Figure 2.2 shows the menu path in Customizing in which you make the necessary settings.
Initially, you activate the generation of change pointers generally. As a result, the share of the CHANGEDOCUMENT_CLOSE function module that has not been used up to now and that is responsible for the SMD is run through. This must be set once only for all master data. Figure 2.3 shows the appropriate functionality.

If the change pointers are activated generally, you can specify in a second step for which message type you require the generation of change pointers. This is done in the second menu subitem (Activate Change Pointers for Message Types) and has been implemented for the message types, MATMAS and MATMAS_WMS (see Figure 2.4). No change pointers are written for MATCOR and MATMAS_GDS, which have not been activated.

In this context, it's important that a change pointer isn't written for each changed field because there are fields whose values are not significant for the downstream system. By means of Transaction BD52, SAP provides fields for each message type connected to the SMD, which are relevant for changes from SAP's point of view. For the material master, the DMAKT-SPRAS field is important, for example. In the transaction, you enter the fields that are used within your enterprise. If you've made
changes to the corresponding master data tables via the SAP enhancement concept and use customer-specific fields, you can also set the customer-specific fields here. Refer to Chapter 4, Changes to IDocs, to learn how to provide these fields.

Figure 2.4  Activate Change Pointers for Message Type

Figure 2.5  Change-Relevant Fields in Transaction BD52

Figure 2.5 shows a section of the fields for the MATMAS message type, which are provided by SAP as change-relevant. The message type is directly referenced to fields and tables of the material master except for the KEY field. This field isn’t part of the respective table but assumes a very important, additional control role: It ensures that the creation of a
Generating IDocs

table entry can be sent via IDoc. If the KEY field is specified in Transaction BD52, a change pointer is written during the creation of the corresponding object, for example, during the initial creation of the material for the MARA-KEY dummy field or during the creation of a text in a new language for the MAKT-KEY dummy field. Imagine that the key value of the table concerned is changed from “empty” to the new value. As a result, all fields of this table are transferred.

Additionally, for each of the change-relevant fields, you must specify to which field in which segment of the IDoc type it belongs. This is done in Transaction BD66, which is shown in Figure 2.6. The DMAKT-SPRAS sample field from Transaction BD52 belongs to the E1MAKTM IDoc segment and to the field that is also called “SPRAS.” For your own fields, you must specify this using the New Entries button.

![Figure 2.6 Assignment of IDoc Fields to Change-Relevant Fields](image)

The change pointers are then evaluated. It just depends on the object concerned which function module is used here. When you call Transaction BD60, you can view these function modules and replace them with your own function modules if you want to make so many changes to the
standard functionality that you don't want to enhance or modify the SAP original. Figure 2.7 shows another example of the material master data. There, the function module that uses change pointers to generate IDocs is called MASTERICOLOR_CREATE_SMD_MATMAS.

<table>
<thead>
<tr>
<th>Change View “Additional Data for Message Type”: Overview</th>
</tr>
</thead>
<tbody>
<tr>
<td>Message Type</td>
</tr>
<tr>
<td>----------------</td>
</tr>
<tr>
<td>MATMAS</td>
</tr>
<tr>
<td>MATMAS_GOS</td>
</tr>
<tr>
<td>MATMAS_UNS</td>
</tr>
</tbody>
</table>

**Figure 2.7** Function Modules to Evaluate Change Pointers

The RBDMIDOC report, which must be scheduled at regular intervals, must then use these function modules to generate IDocs from the change pointers and update which change pointers have been processed. As a transfer value, you can specify for which message type you want to implement the evaluation. For this purpose, you specify the appropriate type (here: MATMAS) in the Message Type field in the initial screen of the RBDMIDOC report. This is illustrated in Figure 2.8.

<table>
<thead>
<tr>
<th>Creating IDoc Type from Change Pointers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Message type</td>
</tr>
</tbody>
</table>

**Figure 2.8** Initial Screen of the “RBDMIDOC” Report

For sending IDocs using the SMD and for change pointers, the application and the ALE communication layer interact closely. Figure 2.9 schematically shows the process flow of IDoc generation. The entire process of writing change pointers takes place in the application; the evaluation of the change pointer and the generation of IDocs are carried out by the ALE communication layer.
Note that when you send IDocs via change pointers, you only send those views in which changes have actually been made. If a new view has been created, all its fields are transferred; if a view has been changed, only the changed fields are transferred to increase performance. Chapter 4, Changes to IDocs, uses the example of the material master to describe how you can change this default behavior using a minor modification.

You can find the change pointers in the BDCP table and the corresponding status records in the BDCPS table. As of SAP NetWeaver Application Server 6.20 (SAP NetWeaver AS), processing with higher performance is possible via a shared table called BDCP2. However, this new procedure isn’t supported for all message types. Whether it’s applicable for your message type is indicated in the detail view of Transaction BD60. As of Release 7.1, only the new processing using Table BDCP2 is available for all message types.

For the distribution via the SMD, you must consider some specifics for some master data. You can set a distribution lock for the material master to generally prevent the sending of a material. For this purpose, you
must make a short detour. In the design data of the material in the MARA table, there is a cross-plant material status (MARA-MSTAE field). This status refers to an existing entry in the T141 table. Here, you can assign additional properties for each status value. If the DLOCK field provided here is selected, the distribution lock is set.

For contracts (BLAORD message type), only released contracts are transferred using the SMD.

**Reducing Messages**

The second requirement on the distribution of master data arises from the distribution of all master data to individual views and the option to define, in Customizing, which fields of a view are actually supposed to be used. This is scalable for the IDoc transfer by using the reduced message types.

A reduced message type always relates to an existing message type but transfers less data. The reduction isn’t possible for all message types; so the developer of the message type must explicitly define it as reducible. All views and fields that must be transferred as a minimum are predefined here; all other views and fields can be selected additionally if required. Transaction BD60 in the detail view for a message type is the transaction for defining a message type as reducible. By calling Transaction BD65, you define the mandatory fields.

For each delivered message type, SAP specifies whether it's reducible. Because this involves functions in the generation and update module for the corresponding IDoc, customers can't simply set the message type to reducible retroactively. The mandatory fields that SAP provides with Transaction BD65 correspond to the Customizing that SAP provides for transactions that are used to maintain master data, for instance, Transaction MAT1 for maintaining the material master. If you make changes to the mandatory fields in the application’s Customizing, you also adapt them in Transaction BD65 so that custom-developed mandatory fields are also mandatory for reducing in the IDoc. Figure 2.10 shows a section of the data for the MATMAS message type.
SAP defines message type as reducible

**BD60 – Detail**

You then create your own reduced message types in Customizing of Transaction SALE. Under the Create Reduced Message Type menu item or via Transaction BD53 you can find the initial screen as shown in Figure 2.11.

**BD65**

Define mandatory fields (both SAP and you)

Figure 2.10 Basic Maintenance for Reducible Message Types

Figure 2.11 Create Reduced Message Type
When you specify the name of the new reduced message type (see Figure 2.12), you must consider the naming rules for your own objects (the name must start with Y or Z or your own namespace).

![Diagram of IDoc Reduction Maintenance: Initial Screen]

**Figure 2.12** Creating a Reduced Message Type

Segments and fields that are green in the SAP system and displayed with an * behind the name are mandatory and can’t be reduced. Fields and segments that are red or marked with – are optional and not selected; segments or fields that are white or marked with + are optional and selected in the respective reduced message type. You now specify which
Generating IDocs

segments you want to have in addition to the mandatory segments by selecting the segment and clicking Select. As soon as you’ve activated a segment, you can select the fields within the segment you want to have in addition to the mandatory fields, and then click Select again.

If you want to generate change pointers for custom reduced message types, you can activate the generation in Transaction BD53 using the Activate Change Pointers button. Of course, you can also activate the change pointers for the reduced message type in Customizing of Transaction SALE. Here, you must make sure that you not only set the flags for generating change pointers but also copy all standard field assignments and mandatory field assignments, which is automatically done in Transaction BD53. In Transaction BD53, you can also disable the writing of change pointers using the Deactivate Change Pointers button (see Figure 2.13).

![IDoc Reduction Maintenance: Initial Screen](image)

**Figure 2.13** Activate the Change Pointer for Reduced Message Types

Directly Generating or Requesting Master Data

If you want to publish the creation or change of material master data without waiting for IDocs that are generated from change pointers, you can use Transaction BD10. In Table 7.5 of Chapter 7, Section 7.3, Transaction Code Overview, you can also find the transactions that belong to other master data.

Because master data usually offers the option of reduction, these transactions “expect” you to enter the message type you want to use for sending and the logical target systems you want to send to. Additionally, you can select the objects you want to generate IDocs for. However, this is only possible via the material numbers or the class membership of the object (see Figure 2.14).
Targeted Sending of Material Master IDocs

If you activated the Send Material in Full flag and also set the distribution of classification IDocs to the same partner, the system generates the classification IDoc that belongs to the material additionally to the material IDoc itself. The specifications for the parallel processing help you increase the performance if you send a high quantity of data, for instance, for initial data load. If you leave the Logical System field empty, the data is sent to all partners that are available in Transaction BD64. If a selection is made, Transaction BD64 checks whether the selected logical system is permitted as a receiver of the material master IDocs. If there is a positive result, the system sends the IDoc.

If you are the receiver of master data IDocs and know that the sending system has changed or has newly created data, you can request a corresponding master data IDoc. The name of the appropriate message type starts like the master data IDoc but uses another abbreviation at the end, that is, FET (for “fetch”) instead of MAS. For example, the name is MATMAS for the message type of the material master IDocs and MATFET for the fetch IDoc.

These fetch IDocs must be maintained as usual in the distribution model (see Figure 1.4 in Chapter 1, Section 1.3, Differentiation of ALE and EDI) — just in the other direction; here, the partner that receives the master data IDoc sends the fetch IDoc. Fetch IDocs always transfer the
object key for which master data IDocs are requested and the message type that is supposed to be used to send the data. You can use Transaction BD11 to “get” material masters (see Figure 2.15).

![Get Material](image)

**Figure 2.15** Requesting the Material Master IDoc

“ALEREQ01” IDoc type

The same ALEREQ01 IDoc type is assigned to all fetch message types. It contains the segments shown in Figure 2.16.

![ALEREQ01](image)

**Figure 2.16** “ALEREQ01” IDoc Type
The actually sent fetch message contains information on the message type that is expected as the response, the short and the long name (before or after Release 4.0), and the keys of elements that are supposed to be sent as a response. Figure 2.17 shows a MATFET IDoc that request the ZSM1 material. For the MATMAS IDoc, both the long and the short name is “MATMAS” because this is a very old message type.

<table>
<thead>
<tr>
<th>MESTYP</th>
<th>Logical message type</th>
<th>MATMAS</th>
</tr>
</thead>
<tbody>
<tr>
<td>MESTYP40</td>
<td>Message Type</td>
<td>MATMAS</td>
</tr>
<tr>
<td>SEGNUM</td>
<td>Segment Number</td>
<td>C00002</td>
</tr>
<tr>
<td>SEGNAME</td>
<td>Segment Name</td>
<td>E1PLE01</td>
</tr>
<tr>
<td>OBJVALUE</td>
<td>Object value (with old length)</td>
<td>MATNR</td>
</tr>
<tr>
<td>SICN</td>
<td>ASAP: ID IE (include/exclude)</td>
<td>1</td>
</tr>
<tr>
<td>OPTION</td>
<td>ASAP: Selection option (EQUIFY)</td>
<td>EQ</td>
</tr>
<tr>
<td>LOWY</td>
<td>Character field of length 40</td>
<td>ZSM1</td>
</tr>
</tbody>
</table>

Figure 2.17 MATFET IDoc

Because master data is usually exchanged between different systems of the same enterprise, logical systems are used as partners like they are used in ALE.

### 2.1.2 Message Control

The message control is a standard function of SAP, which triggers a data transfer for all transaction data that is supposed to be received by other enterprises, too. This can be done using a printer, fax, or an IDoc. For processing using IDocs, you can use transmission medium 6 for processing using partner functions, and you can use transmission medium A for processing using logical systems. All the required settings for the message control are in Transaction NACE.

You use the condition technique to specify when you generate which messages in which way. The key for the conditions is composed of the application you’re in (e.g., “EF” for purchasing), the output type you want to generate (e.g., “NEU” for a purchase order), and the partner role everything will be sent to (e.g., partner “vendor” in its role as goods vendor).
In *message determination*, the actual message is generated using the `RSNAST00` report. The `EDI_PROCESSING` function that is used here transfers the message as an EDI message using the IDoc; it belongs to transmission medium 6. Depending on the system setting, you can call the `RSNAST00` report directly upon saving the document or at regular intervals as a batch job.

In *message control*, an EDI communication is usually assumed so that you work with partners and not with logical systems.

The information on how an IDoc is sent to the receiver (e.g., via RFC or file) and whether an EDI subsystem is supposed to be used if necessary, is set in the *outbound partner profile* in Transaction WE20 both for the communication with partners and for the communication with logical systems.

Here, you also specify which IDoc type is supposed to be used. If you work with the message control, in the outbound partner profile under the Message Control tab, you specify which process code (and which underlying function module) will be used to populate the IDoc data. You can find the valid process codes for the respective output type in Transaction WE41 (Figure 2.18).

You can also specify the optional *message codes* and *message functions*, which you know from the partner profiles, to be able to use different process codes for updating the IDocs. Message codes and functions are
freely selectable, and you don’t need to adhere to naming rules. However, this also means that no input help is provided in Transaction WE20, so you must ensure the correct spelling of the names yourself. In the details of the sample process code for the ORDERS generation, which is shown in Figure 2.19, you can view the link to the associated function module.

![Figure 2.19 Assignment of the Process Code to the Function Module](image)

Under Option ALE-Service/Inb. Procg, you can select whether the ALE services are supposed to be used or not. The ALE services represent options of IDoc manipulation using filters and rules. Chapter 4, Section 4.1, Customizing, discusses the ALE services topic in more detail.

Because you often require a lot of partner profiles — especially if you work with partners — you have the option to create templates to save some time. The partner profile from Transaction WE20 can be created from this template, so you don’t need to create it manually. Transaction WE24 is the transaction for the template in the outbound processing (see Figure 2.20). Transaction WE27 is the corresponding transaction for the inbound processing.
In some cases, a specific business process can be implemented completely locally on an SAP system or distributed across multiple SAP or non-SAP systems. IDocs are only generated when processes are distributed across multiple systems, and the generation can be activated via the Customizing functions of the respective application. Warehouse management is an example of this direct IDoc generation. The default setting of warehouse management assumes that your warehouse is managed by your SAP system. If this isn’t the case, you can activate the connection of your warehouse system via ALE in Customizing. This connection causes a WMTORD IDoc to be generated directly when you create a warehouse transport request to notify the external warehouse of what is supposed
to be transported. However, these special cases can’t be described in
general but instead must be named and set up in cooperation with the
individual end-user. Because all transactions that are required for such a
special case are module-specific, they are not discussed further here, but
just be aware that such special cases exist.

Within the scope of ALE scenarios, there can also be cases in which
sample postings are implemented synchronously via BAPI, and the actual
postings are implemented asynchronously via IDoc. Then, the BAPI is
created on the development side, and the appropriate IDoc is generated
using Transaction BDBG. You can also use this transaction if SAP doesn’t
provide an IDoc for a BAPI and you still require it for a distribution sce-
nario that isn’t provided for by SAP. Again in this case, you must observe
the naming rules for customer objects.

Figure 2.21 shows an IDoc generated by SAP in the SAP namespace. In
the IDoc Interface tab, you can view the names for the message type
and the IDoc type; in the ALE Outbound Processing tab, you can view
the function group in which the IDoc modules are located as well as the
name of the module that generates the IDoc; and in the ALE Inbound
Processing tab, you can view the name of the module that extracts the
IDoc in the receiving system and triggers the update.

The process of generating an IDoc from a BAPI is as follows:

1. The sending system wants to call the BAPI and checks whether this is
supposed to be carried out locally or remotely.

2. If the call is remote and is supposed to be carried out transactionally,
the function module that is generated in Transaction BDBG is called
in the sending system. This function module transfers the transfer
parameters of the BAPI to the IDoc format.

3. After you’ve made the settings in the customer distribution model
and in Transaction WE20, this generated IDoc is transferred to the
receiving system.

4. In the receiving system, the BAPI_IDOC_INPUT1 function module is
called using the BAPI process code or BAPI_IDOC_INPUTP using the
BAPP process code; this depends on whether one or more data records
are received simultaneously. These function modules call the inbound
function module that is generated in Transaction BDBG, which extracts
the IDoc and uses the transferred data to call the original BAPI that carries out the actual update.

Asynchronous BAPIs

Because both cases represent an ALE scenario, you maintain the customer distribution model for BAPIs and IDocs. For the BAPIs, you enter the methods that must be processed both synchronously and asynchronously via IDocs. Additionally, you require partner profiles for the transactional case as usual. If you have these partner profiles generated from the distribution model, the system automatically knows for which BAPIs there are message types and thus for which it requires a partner profile.

Use Transaction BD97 to maintain the destination for the synchronous BAPI call in a remote system. This can be done both generally for all method calls and only for special BAPIs and for dialog calls. You need the dialog calls if you want to work with the IDoc tracing in Transaction BD87. The distinction with regard to dialogs is made for security reasons.
Because a dialog user must be used in the RFC destination, this user should have only a few authorizations.

Figure 2.22 shows an example for each of the three cases.

![Assign RFC Destinations for Synchronous Method Calls](image)

**Figure 2.22** Customizing for Synchronous BAPI Calls in ALE Scenarios

The standard destination for BAPI calls for the APOCLNT800 logical system is APOCLNT800, the destination for the dialog calls is called ALEMANU, and the BACKEND destination is used for the AcctngEmployeePaybles.Check method only.

## 2.2 Use of Logical Systems in the Message Control

Transaction data may be exchanged within ALE scenarios. For example, this could be the scenario of central sales/decentralized shipping. Here, purchase orders, deliveries, and invoices are exchanged between plants of the same enterprise. For this case, in message control, you simply use the customer or vendor as the partner. Instead of “6” for the EDI pro-
cessing, however, you enter “A” for the ALE processing as the transmission medium for the found message. As a result, the system no longer uses the EDI_PROCESSING subroutine and instead uses the ALE_PROCESSING subroutine.

You then maintain a customer distribution model as usual and use a logical system as the sender and a logical system as the receiver of the message. After the message control within the application, the transmission medium A reads the customer distribution model and replaces the partner found in the message control with the logical system found in the model. For this purpose, you need the assignment of output types to message types. They are available in the process codes in the outbound processing in Transaction WE41 and in the settings that you made for the message control in the outbound partner profile in Transaction WE20.

The data from Transaction WE20 is evaluated in the ALE_PROCESSING subroutine. Because the subsequent receiver isn’t yet determined in this first step, the system searches using the “Logical System” partner type and the output type only. If more than one entry is found in Transaction WE20, for example, NEU – ORDERS, which would be the standard, and NEU – ZSMORD for a self-programmed message type, ALE_PROCESSING cancels with an error. You establish the required uniqueness by using your own output types for your own message types.

The search via the “Logical System” partner type and the partner type results in the benefit of a reduced number of partner profiles. Then you can just have a partner profile for the logical system instead of a partner profile for each vendor or customer.

2.3 Summary

In this chapter, we've taken a closer look at the creation of IDocs. You've learned which methods for creation exist and how they are related to the different needs of different processes or data types. The decision as to which one should be used is made by SAP.

Now let's take it a step further regarding our work with IDocs. The next chapter will teach you how to test the creation and the posting of IDocs.
Index

A

ABAP List Viewer → ALV, 134
Acknowledgment, 187
Agency, 244
Agent assignment, 184
ALE, 15
  business process, 15, 18
  communication layer, 35
  Customizing → SALE, 31
  scenario, 15, 18
  service, 45, 73, 84, 134
ALE00001, 119
ALEAUD, 187, 230
ALE interface
  for BAPI, 173
  for function modules, 174
ALE_MODEL_DETERMINE_IF_TO_SEND, 155
ALE_PROCESSING, 50
ALEREQ01, 42
ALE_SERIAL_KEY2CHANNEL, 200
ALV, 231
ANSI ASC X12, 27
API (Application Programming Interface), 18
Application, 43
Application acknowledgment, 187
Application Link Enabling → ALE, 15
Application record, 21
Archiving object, 225
Asynchronous interface, 17
a_t_complex_matnr, 135

B

BAdl, 91
  classic, 103
  implementing class, 113
  interface, 91, 113
  new, 110
BAPI, 18, 47
  release, 19
BAPI_IDOC_INPUT1, 47
BAPI_IDOC_INPUTP, 47
BAPIRETM, 174
BAPIRETS, 174
BAPI_TRANSACTION_COMMIT, 19
BAPI_TRANSACTION_ROLLBACK, 19
BAPP, 47
Basic type, 126
BD10, 40
BD40, 195
BD41, 196
BD42, 197
BD43, 197
BD44, 194
BD51, 168
BD52, 32
BD53, 38, 40
BD54, 25
BD55, 84
BD56, 77
BD57, 198, 199
BD58, 89
BD59, 74
BD60, 34, 36, 69, 70, 135, 151
BD64, 26, 41, 67
BD65, 37
BD66, 34
BD68, 72
BD79, 81
BD87 → IDoc-Monitor, 24
BD95, 73, 198
BD97, 48
BD100;BD97a, 202, 204
BD101;BD97b, 202
BD102;BD97c, 204
BD103;BD97d, 205
BD104;BD97e, 200
BD105;BD97f, 200
BDA4, 176
BDBG, 24, 47, 173, 174
INDEX

BDCP, 36
BDCP2, 36
BDCP_BEFORE_WRITE, 121
BDCP2, 36
BDCPS, 36
BDFG, 174
BDSEN, 198
BF01, 100
BF05, 100
Binding, 181
BOMMAT, 200
BOR, 18, 168
Breakpoint, 110
BTE, 90
BUS1001006, 168
Business Add-In → BAdI, 91
Business Application Programming Interface → BAPI, 18
Business Object Repository → BOR, 18
Business object type, 18
Business-related error, 20
Business system, 243
Business transaction event → BTE, 90
Business workflow, 18

CALL CUSTOMER-FUNCTION, 93
C file, 23
Change document, 30
CHANGEDOCUMENT_CLOSE, 30
CHANGEDOCUMENT_OPEN, 30
Change pointer, 31
  activate generally, 32
  for message type, 32
  for reduced message types, 40
CHANGE_POINTERS_CREATE, 31
CHANGE_POINTERS_CREATE_LONG, 31
Channel number, 200
CIMTYP, 93
CL02, 72
Class, 71
Classic BAdI, 103
Classification, 69, 72
CL_EXITHANDLER, 103
CMOD, 94
COMMith, 92
Communication channel, 245
Company code, cross-system, 85
CONDENSE, 153, 157
Condition technique, 43
Conversion exit, 83
CONVERSION_EXIT_ALPHA_OUTPUT, 83
Conversion routine, 83
Cross-system business area, 85
Cross-system company code, 85
Customer
distribution model, 26, 67
management object, 101
project, 95
segment, 122
Customer exit, 90, 93

Data channel registry, 204
Data Dictionary → DDIC, 124
Data record, 21
Data Source, 228
Date field, 153
DDIC, 124
Debugging, 55
Decimal number, 153
Description
  semantic, 16
  technical, 16
Dialog call, 48
Distribution class type, 70
Document number, 198
DTD (Document Type Description), 23

EDI, 26
EDIFACT, 22, 27, 144
EDI_PROCESSING, 44, 50
EDIS, 65, 175
Electronic Data Interchange → EDI, 26

332 Book.indb 252
11/3/09 10:23:19 AM
Enhancement, 91, 95, 107
   check, 129
dynamic, 118
explicit, 107
implement, 108
implementation, 111
implicit, 115
partner profile, 131
point, 107
project, 90
release, 128
section, 107, 108
spot, 91, 107, 108, 110
static, 118
technology, 67
type, 125
EOIO (Equally Once In Order), 205
Error
   business-related, 20
   status, 167
technical, 20
Error workflow, 175, 176
   connection, 167
Exit, 90
EXIT_SAPLBD11_001, 119
Extensible Markup Language → XML, 23
Extension, 126

F
Fetch IDoc, 41
FIBF, 101
FIDCMT, 89
File, 23
FILE, 140
File port, 56, 138
Filter group, 68
Filter object, 67, 68, 73
Floating point number, 153
Function module
   evaluation of change pointers, 34
   file name generation, 138
   for queue names, 206
test, 55

G
GDPdU, 224
General method call, 48
General rule, 83
General task, 181, 184
Generate control record, 158
German Principles on Data Access and the Examination of Digital Documents, 224
GET*BADI, 110
GMP (Good Manufacturing Practice), 224

H
Header mapping, 245
Header record, 21
Header table, 141
HTML (Hypertext Markup Language), 23
HTTP (Hypertext Transfer Protocol), 23

I
IDoc, 18
   adapter, 243
   administrator, 184
   monitor, 24, 160, 168
   request, 41
   status file, 62
   suppress, 68, 76
   tracking, 24
type, 22
IDOCAPPL, 176
IDOCBOMMAT, 200
IDOC_DATE_TIME_GET, 24
IDOC_INBOUNDASYNCHRONOUS, 23
IDOC_INBOUND_IN_QUEUE, 205
IDOC_INPUT_MATMAS01, 133
IDOC_REDUCTION_FIELD_REDUCE, 156
IDOCREL, 211, 212
IDOC_SERIALIZATION_CHECK, 199
**Index**

<table>
<thead>
<tr>
<th>Key Word</th>
<th>Page Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDOC_SERIAL_POST</td>
<td>199</td>
</tr>
<tr>
<td>IDX_DELETE_IDOCTYP_WITHOUT_IS</td>
<td>248</td>
</tr>
<tr>
<td>IDXIDOCINB</td>
<td>248</td>
</tr>
<tr>
<td>IDX_SELECT_IDOCTYP_WITHOUT_IS</td>
<td>248</td>
</tr>
<tr>
<td>Importing parameter unique</td>
<td>165</td>
</tr>
<tr>
<td>Inbound interface</td>
<td>15</td>
</tr>
<tr>
<td>Info structure</td>
<td>227</td>
</tr>
<tr>
<td>Initial</td>
<td>153</td>
</tr>
<tr>
<td>INPUTERROROCCURRED</td>
<td>179</td>
</tr>
<tr>
<td>inputFinished</td>
<td>178, 181</td>
</tr>
<tr>
<td>INPUTFOREGROUND</td>
<td>179</td>
</tr>
<tr>
<td>Interface</td>
<td>17</td>
</tr>
<tr>
<td>asynchronous</td>
<td>17</td>
</tr>
<tr>
<td>synchronous</td>
<td>16</td>
</tr>
<tr>
<td>Intermediate Document → IDoc</td>
<td>18</td>
</tr>
<tr>
<td>ISO code</td>
<td>144, 153</td>
</tr>
<tr>
<td>Item table</td>
<td>141</td>
</tr>
<tr>
<td><strong>K</strong></td>
<td></td>
</tr>
<tr>
<td>KEY</td>
<td>33</td>
</tr>
<tr>
<td><strong>L</strong></td>
<td></td>
</tr>
<tr>
<td>Link</td>
<td>211</td>
</tr>
<tr>
<td>Logical message</td>
<td>151</td>
</tr>
<tr>
<td>Logical system (LS)</td>
<td>25</td>
</tr>
<tr>
<td><strong>M</strong></td>
<td></td>
</tr>
<tr>
<td>Mandatory field</td>
<td>37</td>
</tr>
<tr>
<td>Mandatory segment</td>
<td>127, 148</td>
</tr>
<tr>
<td>Master data</td>
<td>29</td>
</tr>
<tr>
<td>MASTERIDOC_CREATE_MATMAS</td>
<td>93, 107</td>
</tr>
<tr>
<td>MASTERIDOC_CREATE_SMD_MATMAS</td>
<td>135</td>
</tr>
<tr>
<td>MASTER_IDOC_DISTRIBUTE</td>
<td>159</td>
</tr>
<tr>
<td>MATFET</td>
<td>26, 41</td>
</tr>
<tr>
<td>MATMAS</td>
<td>22, 26</td>
</tr>
<tr>
<td>mbdconwf</td>
<td>162</td>
</tr>
<tr>
<td>Message</td>
<td>44</td>
</tr>
<tr>
<td>code</td>
<td>44</td>
</tr>
<tr>
<td>control</td>
<td>29, 43</td>
</tr>
<tr>
<td>determination</td>
<td>44</td>
</tr>
<tr>
<td>function</td>
<td>44</td>
</tr>
<tr>
<td>Message type</td>
<td>22, 151</td>
</tr>
<tr>
<td>reduced</td>
<td>37</td>
</tr>
<tr>
<td>Meta information</td>
<td>22</td>
</tr>
<tr>
<td>Method call</td>
<td>48</td>
</tr>
<tr>
<td>MGV00001</td>
<td>95</td>
</tr>
<tr>
<td>MGV00100</td>
<td>100</td>
</tr>
<tr>
<td>Modification</td>
<td>90</td>
</tr>
<tr>
<td>MOVE</td>
<td>81</td>
</tr>
</tbody>
</table>

**N**

<table>
<thead>
<tr>
<th>Key Word</th>
<th>Page Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>NACE</td>
<td>43</td>
</tr>
<tr>
<td>Naming rule</td>
<td>148</td>
</tr>
<tr>
<td>customer object</td>
<td>47</td>
</tr>
<tr>
<td>own object</td>
<td>39</td>
</tr>
<tr>
<td>New BAdI</td>
<td>110</td>
</tr>
<tr>
<td>NO_DATA character</td>
<td>30</td>
</tr>
</tbody>
</table>

**O**

<table>
<thead>
<tr>
<th>Key Word</th>
<th>Page Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>O1CL</td>
<td>70</td>
</tr>
<tr>
<td>Object channel serialization</td>
<td>200, 201</td>
</tr>
<tr>
<td>Object Key</td>
<td>214</td>
</tr>
<tr>
<td>Object services</td>
<td>214</td>
</tr>
<tr>
<td>OBJTYPE</td>
<td>212</td>
</tr>
<tr>
<td>OPEN_FI_PERFORM_XXXXXXX_E;OPEN_FI_PERFORM_XXXXXXX_E</td>
<td>99</td>
</tr>
<tr>
<td>ORDCHG</td>
<td>22, 198</td>
</tr>
<tr>
<td>ORDERS</td>
<td>22</td>
</tr>
<tr>
<td>Organizational management</td>
<td>184</td>
</tr>
<tr>
<td>Outbound interface</td>
<td>15</td>
</tr>
<tr>
<td>Outbound partner profile</td>
<td>44</td>
</tr>
<tr>
<td>Output</td>
<td>43</td>
</tr>
<tr>
<td>type</td>
<td>43</td>
</tr>
<tr>
<td>OYEA</td>
<td>184</td>
</tr>
</tbody>
</table>
Index

P

Partner, 26
inbound profile, 24
outbound profile, 24
profile, 45, 79
role, 43, 145
Party, 244
PFTC_INS, 178
PI → SAP NetWeaver PI, 187
Port management, 138
Process code, 168
BAPI, 174
BAPP, 174
Processing layer, 63
Process Integration → SAP NetWeaver PI, 187
Process interface, 91
Proposal for rule, 81
Proxy interface, 16
P/S interface, 91
Publish&Subscribe → P/S interface, 91

Q

qRFC, 205
Qualifier, 144, 145
Quality of Service, 205
Queue, 209
Queued RFC → qRFC, 205

R

RBDMIDOC, 35
RBSDER01, 194, 195
RBSDER02, 196
RBSDER03, 197
RBSDER04, 197
RBDSCRLR, 199
RDBSTATE, 188
Receiver, 26
Receiver communication channel, 245, 247
Reduced message type, 37
Release security, 15, 19
Remote Function Call → RFC, 16
Request, 16
Request IDocs, 41
Response, 16
RFC, 16
queued → qRFC, 205
synchronous → sRFC, 19
transactional → tRFC, 19
RFM (RFC-enabled function module), 18
ROLEID, 212
ROLLBACK, 92
Root segment, 78, 79
RSEOUT00, 59, 196
RSNAST00, 44, 58
RSRLDREL, 215
Rule, 67, 80
general, 83

S

SALE, 25, 31
Sample file, 56
SAP_ALE_EDIQUEUE, 206
SAP_IDOC_001, 227
SAP NetWeaver PI, 205, 243
agency, 244
header mapping, 245
partner, 26
party, 244
SARA, 225
SARI, 228
SBWP, 183
SE11, 124
SE18, 103
SE19, 103, 110
SE80, 111
SE84, 91
Segment
definition, 124
filter, 76
qualifying, 145
release, 125
type, 23, 124
version, 84
Sender, 26
Index

Sender field
  convert, 83
  copy, 82
SERDAT, 197
SERIAL, 198, 203
Serialization, 193
  business objects, 200
  flag, 202
  group, 193, 194
  timestamp, 198
Shared Master Data Tool → SMD, 29
Signature, 154
  inbound function module, 162
SLD, 244
SM59, 102
SMD, 29
SNDPOR, 243
SNDPRN, 244
SOAP, 16
Special method call, 48
sRFC, 19
SRRELROLES, 211
Standard inbound processing, 54
Standard interface, 15
Standard outbound processing, 57
Standard task, 178
Status 42<normal>, 57
Status 53<normal>, 188
Status 66, 204
Status 68<normal>, 188
Status 74<normal>, 61
STATUS IDoc, 62, 190
Status record, 21
Success status, 167
SWO1, 176
Synchronous interface, 16
Synchronous RFC → sRFC, 19
SYSTAT, 62, 230
System Landscape Directory → SLD, 244
System name, logical, 25
System, technical, 244

Technical system, 244
Templates, 45
Test
  flag, 54
  function module, 55
  transaction, 51
Time, 153
Transactional RFC → tRFC, 19
Transaction data, 29
Transaction security, 17
TRANSID, 212
Transmission medium 6, 43, 44
Transmission medium A, 43, 50
Transport IDoc, 23
tRFC, 19

UN/EDIFACT → EDIFACT, 22
Unicode, 56
UNIT_OF_MEASURE_ISO_TO_SAP, 165
UNIT_OF_MEASURE_SAP_TO_ISO, 157
Update technique, 19

View, 77

Version conversion, 84, 119

Task, general, 181, 184
Technical error, 20
<table>
<thead>
<tr>
<th>Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>WE41, 44</td>
</tr>
<tr>
<td>WE42, 169</td>
</tr>
<tr>
<td>WE47, 63, 224</td>
</tr>
<tr>
<td>WE55, 139</td>
</tr>
<tr>
<td>WE57, 133, 168</td>
</tr>
<tr>
<td>WE60, 150</td>
</tr>
<tr>
<td>WE62, 81</td>
</tr>
<tr>
<td>WE81, 151</td>
</tr>
<tr>
<td>WE82, 136, 151</td>
</tr>
<tr>
<td>WE85, 205</td>
</tr>
<tr>
<td>WEDI, 51</td>
</tr>
<tr>
<td>WEINBQUEUE, 208</td>
</tr>
<tr>
<td>WENOLINKS, 214</td>
</tr>
<tr>
<td>WEOUTQUEUE, 206</td>
</tr>
<tr>
<td>WMTORD, 46</td>
</tr>
<tr>
<td>XML, 23</td>
</tr>
<tr>
<td>xPath, 246</td>
</tr>
</tbody>
</table>