7 Internes Messaging in R/3

Auch SAP R/3 verwendet ein sehr ausgefeiltes Messaging-Konzept, um die parallele Verarbeitung der Anwendung sicherzustellen. Grundsätzlich kennt SAP R/3 folgende Message-Pipes: Verbucher, Nachrichtensteuerung, Workflow und IDocs.

7.0.1 NAST Message Control

Die Nachrichtensteuerung in SAP R/3 ist um die Tabelle NAST herumgebaut, die auch ihren Namen davon ableitet (*NA*chrichten*ST*euerung). Wie bei jeder Message-Queue legt die anfragende Applikation ihre Nachrichten in NAST ab, wo sie dann von einem interessierten Verbraucher irgendwann abgegriffen und verarbeitet wird.

Der Verbraucher läuft asynchron

Ein Programm, das eine Nachricht abarbeitet, nennt man *Verbraucher* (englisch: *consumer*) oder *Handler*. Der Verbraucher läuft asynchron zum Requester ab, ist also vollkommen von diesem entkoppelt. Der Zeitpunkt der Abarbeitung der Nachricht ist nicht vorher bestimmt und die Verarbeitung der Nachricht läuft als separate Transaktion ab. Falls die rufende Transaktion das Ergebnis benötigt, muss sie die eigene Transa ktion erst selbst beenden und dann das Ergebnis pollen.

Entkopplung der Transaktion nur eingeschränkt

Die Entkopplung von NAST-Routinen aus einer Transaktion ist leider nicht vollständig. Tatsächlich wird im Falle eines Verarbeitungszeitpunkts »4 = sofort nach Sichern verarbeiten« die Verarbeitung noch innerhalb des Verbuchens angestoßen und unterliegt somit den Restriktionen der Verbucherverarbeitung. Es besteht die Möglichkeit, dies zu umgehen, indem man aus der Verarbeitungsroutine heraus einen RFC absetzt, allerdings ist es dann schwierig, den Verarbeitungsstatus synchron zu halten:

- Der Requester speichert seine Nachricht in der Tabelle NAST.
- Ein Programm, gewöhnlich `RSNAST00`, pollt die Tabelle NAST nach unverarbeiteten Nachrichten und aktiviert das Programm, das für die Verarbeitung der Nachricht als Verbraucher vorgesehen ist.
- Die Verarbeitungsroutine schließt die Verarbeitung ab, indem sie die Nachricht als verarbeitet kennzeichnet und einen Status in NAST zurückschreibt.

- Wenn RSNAST00 zur Verarbeitung verwendet wird, werden alle Verwaltungsaufgaben von diesem übernommen und lediglich eine im Customizing vorzugebende Routine zur Verarbeitung aufgerufen.

7.0.2 Entwicklung eigener NAST-Handler

Standardverarbeitung in SAP R/3 via RSNAST00

Wenn Sie planen, selbst Nachrichten aus der Tabelle NAST zu verarbeiten, sollten die Verarbeitungsroutinen kompatibel mit der allgemeinen Nachrichtenverarbeitung in RSNAST00 sein. Die Standardverarbeitung von SAP R/3 legt Nachrichten in der Tabelle NAST über die Nachrichtenfindung mit dem Funktionsbaustein function MESSAGING an. Dieser benutzt zum einen die Konditionstabellen für die Nachrichtenfindung, sofern diese für die gewünschte Applikation definiert worden sind. Am Ende einer Transaktion, die die Standardverarbeitung verwendet, ruft SAP R/3 die Routine

```
PERFORM einzelnachricht IN PROGRAM RSNAST00
```

auf. Diese Routine geht davon aus, dass in der Tabelle TNAPR für die gewünschte Nachricht ein Eintrag vorhanden ist, der den Namen des Verarbeitungsprogramms enthält. Das folgende Listing zeigt ein Beispiel für eine solche Routine.

Listing 7.1 ZSNASTWF – Beispiel einer NAST-Routine zum Triggern eines Workflows

```
*********************************************************
* Collection of NAST processing routines                *
*  for media = 8 : special processing                   *
*  for media = 9 : Workflow Event                       *
*  for media = T : Workflow Task                        *
*                                                       *
*********************************************************
* <object> contains call declarations for object hand-
ling
INCLUDE <OBJECT>.
* Following are the declarations for standard NAST
INCLUDE RVADTABL .
DATA: RETCODE LIKE SY-SUBRC.
DATA: XSCREEN.
*---------------------------------------------------------*
*       FORM CREATE_EVENT                                 *
*---------------------------------------------------------*
```

```
*  Routine is used to process NAST media type "9" (Work-
flow Event)    *
*  It will actually raise the specified workflow Event
*  as defined in NAST-OBJECT and NAST-EVENT
*  Note: NAST-EVENT must exist for object NAST-OBJECT
*        or its delegation type (=derived subtype)
*---------------------------------------------------*
FORM ENTRY_CREATE_EVENT USING RETURN_CODE US_SCREEN.
  PERFORM CREATE_EVENT(RSWEMC01) USING RETURN_CODE US_
SCREEN.
ENDFORM.                                " create_Event.
```

Das folgende Listing zeigt eine Routine zum Rückschreiben eines Verarbeitungsprotokolls.

Listing 7.2 Update des Verarbeitungsprotokolls im NAST-Handler

```
FORM PROTOCOL_UPDATE USING
            MSG_ARBGB
            MSG_NR
            MSG_TY
            MSG_V1
            MSG_V2
            MSG_V3
            MSG_V4.
  CHECK XSCREEN = SPACE.
  SYST-MSGID  = MSG_ARBGB.
  SYST-MSGNO  = MSG_NR.
  SYST-MSGTY  = MSG_TY.
  SYST-MSGV1  = MSG_V1.
  SYST-MSGV2  = MSG_V2.
  SYST-MSGV3  = MSG_V3.
  SYST-MSGV4  = MSG_V4.

  CALL FUNCTION 'NAST_PROTOCOL_UPDATE'
       EXPORTING
            MSG_ARBGB = SYST-MSGID
            MSG_NR    = SYST-MSGNO
            MSG_TY    = SYST-MSGTY
            MSG_V1    = SYST-MSGV1
            MSG_V2    = SYST-MSGV2
```

```
            MSG_V3    = SYST-MSGV3
            MSG_V4    = SYST-MSGV4
       EXCEPTIONS
            OTHERS    = 1.
ENDFORM.
```

Im nächsten Beispiel erzeugen wir ein Workitem aus einer NAST-Nachricht. Ein Workitem ist eine E-Mail, die eine Entscheidung vom Empfänger erwartet. Sobald der Entscheider sein Workitem beantwortet hat, wird der Workflow unter Auswertung der Antwort fortgeführt.

Listing 7.3 Workitem erzeugen aus einem NAST-Eintrag

```
FORM ENTRY_WORKITEM USING RETURN_CODE US_SCREEN.
  CLEAR RETCODE.
  XSCREEN = US_SCREEN.
    PERFORM DO_WORKITEM USING US_SCREEN NAST-OBJKY.
    CASE RETCODE.
      WHEN 0.
        RETURN_CODE = 0.              "all well done,
        sets VSTAT = 1
      WHEN 3.
        RETURN_CODE = 3.  "means not processed, leaves
        VSTAT = 0
      WHEN OTHERS.
        RETURN_CODE = 1.              "Errors, sets
        VSTAT = 2
    ENDCASE.
ENDFORM.

FORM DO_WORKITEM USING RETURN_CODE US_SCREEN.
DATA: WI_CONTAINER LIKE SWCONT   OCCURS 0 WITH HEADER
LINE.
DATA: AGENTS       LIKE SWHACTOR OCCURS 0 WITH HEADER
LINE.
DATA: SWWWIHEAD    LIKE SWWWIHEAD.
DATA: BEGIN OF OBJECT_ID,
       LOGSYS  LIKE T000-LOGSYS,
       OBJTYPE LIKE SWOTENTRY-OBJTYPE,
       KEY  LIKE SWCONT-VALUE,
      END OF OBJECT_ID.
*
```

```
DATA: TASKNAME LIKE HRP1000-MC_SHORT.
DATA: OBJNAME  LIKE SWOTENTRY-OBJTYPE.
*ARAMETERS: taskname LIKE hrp1000-mc_short DEFAULT
'ZAXXVA02'.
*ARAMETERS: objname  LIKE swotentry-objtype DEFAULT
'BUS2032'.
*  *** ************ Object ID ************************
* The logical system is retrieved from table T000-logsys
 CALL FUNCTION 'OWN_LOGICAL_SYSTEM_GET'
      IMPORTING OWN_LOGICAL_SYSTEM = OBJECT_ID-LOGSYS.
* BUS2032 is the sales order BAPI
 OBJECT_ID-OBJTYPE = NAST-OBJTYPE.
 TASKNAME          = NAST-EVENT.
 OBJECT_ID-KEY     = NAST-OBJKY.
* *** Load Container
 REFRESH WI_CONTAINER.
* The object_id must be passed to the macro as an
unstructured type
* If the type is structured, one single entry for every
sub-field
* of the parameter is created. This is not what we want.
 WI_CONTAINER-VALUE = OBJECT_ID.
 SWC_SET_ELEMENT WI_CONTAINER '_WI_OBJECT_ID'
 WI_CONTAINER-VALUE.
*swc_set_element wi_container 'SALESDOCUMENT' object_id-
key.
 SWC_SET_ELEMENT WI_CONTAINER 'VBELN' OBJECT_ID-KEY.
* *** Identify yourself
SWWWIHEAD-WI_CREATOR = SY-UNAME.
* *** Set the action that continues the Workitem
SELECT OBJID INTO SWWWIHEAD-WI_RH_TASK "need internal
object id
        FROM HRP1000 UP TO 1 ROWS
        WHERE MC_SHORT EQ TASKNAME.
ENDSELECT.
IF SY-SUBRC NE 0. MESSAGE A000(38) WITH 'Task not found'
TASKNAME.ENDIF.
* *** Define the users, which are meant to check the
workitem
REFRESH AGENTS.
```

```
AGENTS-OTYPE = 'US'.
AGENTS-OBJID = 'ANGELIAX1'.
APPEND AGENTS.
CALL FUNCTION 'SWW_WI_START_SIMPLE'
     EXPORTING
          CREATOR                       = SWWWIHEAD-
                                            WI_CREATOR
          TASK                          = SWWWIHEAD-
                                            WI_RH_TASK
     TABLES
          AGENTS                        = AGENTS
          WI_CONTAINER                  = WI_CONTAINER
     EXCEPTIONS
            ID_NOT_CREATED                = 1
            READ_FAILED                   = 2
            IMMEDIATE_START_NOT_POSSIBLE = 3
            EXECUTION_FAILED              = 4
            INVALID_STATUS                = 5
            OTHERS                        = 6.
ENDFORM. " entry_workitemh USING return_code us_screen.
```

7.0.3 Workflow-Events

Mit Workflow-Ereignissen kann man nachfolgende Verarbeitungen unmittelbar aufrufen. Die Idee dahinter ist, dass das rufende Programm der Öffentlichkeit, de facto der Workflow-Engine, seinen eigenen Verarbeitungsstatus anzeigt. Man sagt, es *feuert einen Event*. Die Workflow-Engine analysiert den Event und reagiert entsprechend darauf. Dazu wird die Liste der interessierten Applikationen durchgegangen und diese gegebenenfalls aufgerufen. Falls keine Applikation für den Event vorgesehen ist, wird er ignoriert, man spricht davon, dass der *Event verbrannt* werde.

7.0.4 Pro und Kontra Workflow versus NAST

Pro und Kontra Workflow und NAST

Die Entscheidung zwischen NAST und Workflow hängt von verschiedenen Kriterien ab. Beide Messaging-Varianten leisten im Endergebnis das Gleiche:

- Die Protokollierung bei NAST ist einfacher, da bereits Mechanismen zum Wegschreiben der Status und von Langprotokollen vorgesehen sind. Dazu bedient sich der NAST-Handler des Funktionsbausteins `Function NAST_PROTOCOL_UPDATE`.

- Zur Protokollierung von Langtexten bei Workflow müssen Sie entweder eigene Mechanismen verwenden oder Sie greifen auf den Applikationslog von R/3 zurück, der mit den Funktionsbausteinen der Funktionsgruppe SLG0, die mit `APPL_LOG_...` beginnen, geschrieben wird.
- Workflows werden grundsätzlich synchron gestartet. Versagt der Workflow, wird er nicht mehr neu gestartet.
- Nur einzelne Transaktionen unterstützen die Nachrichtenverarbeitung mit NAST. Ein nachträglicher Einbau innerhalb eines User-Exits ist zwar oft möglich, aber in jedem Fall schwierig.
- NAST erlaubt in der Verarbeitungsroutine keine Befehle, die nicht für Verbuchungsroutinen zugelassen sind, solange die Verarbeitung durch die auslösende Transaktion auch gestartet werden soll.
- Workflow erlaubt jede beliebige Verarbeitung, da der Aufruf immer in einem eigenen Programmkontext in einer eigenen RFC-Destination ausgeführt wird. Die übliche RFC-Destination für einen Workflow lautet `WORKFLOW_LOCAL_`*nnn*, wobei *nnn* die Nummer des R/3-Mandanten ist, in dem der Workflow ausgeführt werden soll.

7.1 Messaging mit R/3-IDocs

IDoc ist die Abkürzung für *Intermediate Document*. Dabei handelt es sich um ein formales Dateiformat, das von SAP festgelegt wurde, um Daten mit externen Systemen auszutauschen. Der Aufbau eines IDocs ist immer gleich und besteht aus

- einem Kopfsatz
- einer beliebigen Anzahl von Datensätzen
  wobei jeder Datensatz aus einer führenden Sequenz als Segmentkennung und einem Datensatz mit fester Länge von 1.000 Zeichen besteht

IDoc versendet eine Nachricht mit Daten

SAPs IDocs wurden ursprünglich als asynchrone Message-Queue für eingehende und ausgehende EDI-Daten entworfen. Im Unterschied zu einfachen Nachrichten mit NAST überträgt ein IDoc mit der Nachricht auch die gesamten Daten.[1]

SAP erzeugt keine EDI-Formate

SAP erzeugt keine klassischen EDI-Datenformate wie EDIFACT oder ANSI X.12. Ab Release 4.6 können IDocs auch als XML-Datei ausgegeben werden.

1 Für weitere Informationen zu IDocs empfehlen wir Axel Angeli: *The R/3 Guide to EDI and Interfaces*. Wiesbaden 2001.

Die folgende Tabelle zeigt das Beispiel eines IDocs mit Segment-Info und festem 1000-Zeichen-Datenblock rechts.

Segment Info Segment Data-à	
...E1MARAM ....00000001234567...	Material base segment
...E1MARCM ....PL01...	Plant Segment
...E1MARDM SL01	Storage location data
...E1MARDM SL02	Another storage location
...E1MARCM PL02	Another plant

Tabelle 7.1 Beispiel-IDoc

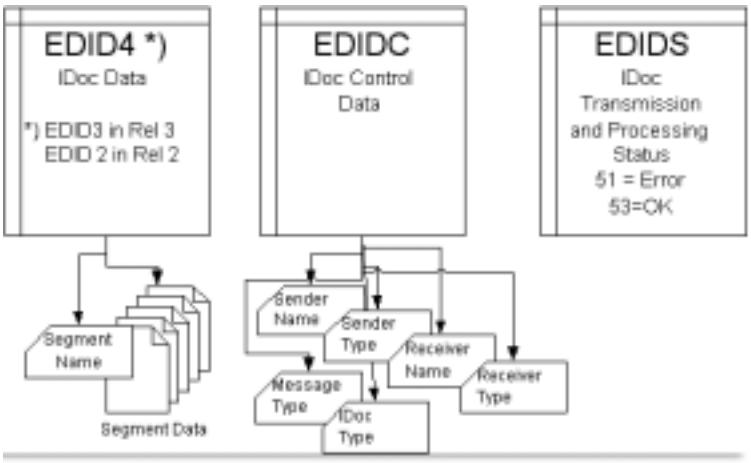

Abbildung 7.1 Tabellen, in denen IDocs in SAP R/3 gespeichert werden

Listing 7.4 Beispiel eines IDocs zur Übertragung von Materialstammdaten

```
EDI_DC40  0430000000000001234540B 3012  MATMAS03    MAT-
MAS   DEVCLNT100  PROCLNT100
E2MARAM001
0430000000000001234500000100000002005TESTMAT1
19980303ANGELI       19981027SAPOSS
E2MAKTM001
0430000000000001234500000300000103005EEnglish Name for
TEST Material 1        EN
E2MAKTM001
0430000000000001234500000400000103005FFrench Name for
TEST Material 1         FR
```

```
E2MARCM001
043000000000001234500000500000103005010ODEAVB
901       PD9010  0   0.00 EXX  0.000
E2MARDM001
04300000000000123450000060000050400510000
0.000         0.000
E2MARDM001
04300000000000123450000070000050400512000
0.000         0.000
E2MARMM
04300000000000123450000090000010300 5KGM1   1
0.000         0.000
```

Listing 7.5 Teil des Inhalts eines IDoc-Files für IDoc-Type MATMAS01

```
      0000000000012345 DEVCLNT100 PROCLNT100 19991103
210102
         E1MARAM                    005 TESTMAT1
19980303 ANGELI      19981027SAPOSS       KDEAVCB
               E1MAKTM                     005 D Ger-
man  Name for TEST Material 1       DE
               E1MAKTM                     005 E Eng-
lish Name for TEST Material 1       EN
               E1MAKTM                     005 F French
Name for TEST Material 1        FR
               E1MARCM                     005 0100
DEAVB               901
```

IDocs harmonisch in Applikationen eingefügt

Abbildung 7.2 zeigt das Funktionsprinzip der SAP-IDoc-Engine. Darin wird deutlich, dass IDocs harmonisch in das gesamte Geflecht der Applikationen von SAP eingefügt werden. IDocs werden gewöhnlich durch einen Eintrag in die NAST-Datenbank ausgelöst, können aber auch direkt aus der Applikation erzeugt werden.

Die folgenden Listings zeigen Auszüge aus einem IDoc. Jedes IDoc wird durch einen standardisierten Header-Satz eingeleitet. Die Datensätze haben eine Segmentkennung und einen Datenblock von fester Länge.

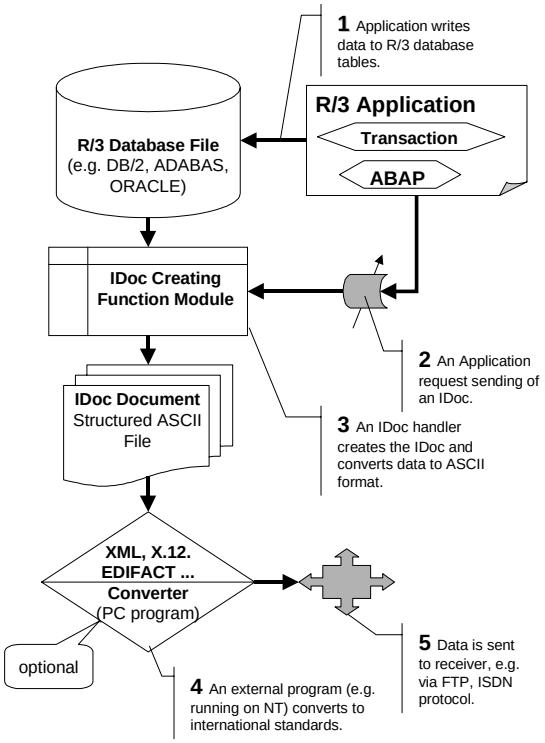

Abbildung 7.2 Ein typisches IDoc/EDI-Ausgangsszenario aus Sicht von R/3

Listing 7.6 Einfaches Beispiel eines IDoc Control Record für Kundenaufträge

```
IDOC Number Sender  Receiver   Port Message Type IDoc
Type
0000123456  R3PARIS R3MUENCHEN FILE ORDERS
ORDERS01
```

Listing 7.7 Einfaches Beispiel eines IDoc-Datensatzes für Kundenaufträge

```
SegmentType Sold-To Ship-To Value      Deldate  User
ORDERHEADER 1088    1089    12500,50   24121998 Micky
Maus
```

7.2 SAP R/3 Workflow programmieren

Automatische Abfolgen von Programmen

Ein Workflow dient dem automatisierten Verketten von einzelnen Transaktionen auf der Basis eines ausgelösten Events. R/3-Workflows erlauben auch konditionale Handlungsabläufe und die Weitergabe von Daten zwischen den einzelnen Schritten der Verarbeitung. Workflows können angehalten werden, um Benutzeraktionen zu erfragen, etwa das Senden einer E-Mail, das eine Benutzerentscheidung verlangt.

Kopplungstabelle

Ein Workflow ist eine Kette von Workflow-Schritten, die nacheinander ausgeführt werden. Ein solcher Workflow ist in der Tabelle SWETYPECOU festgelegt, beziehungsweise ab R/3 Enterprise in den Tabellen SWFDEVENA und SWFDEVTYP. Bei diesen Schritten handelt es sich entweder um einen Funktionsbaustein oder um einen mit der Transaktion PFTC angelegten Workflow-Task.

Bedingungen festlegen

Zu jedem Schritt kann ein Funktionsbaustein festgelegt werden, der bestimmt, ob der Schritt ausgeführt wird oder nicht.

Abbildung 7.3 Workflow-Kopplung zwischen Ereignis und Verbraucher

Field name	Example	Description
OBJCATEG	BO	Object Type Category
OBJTYPE	IDOCAPPL	Type of Objects in Persistent Object References
EVENT	INPUTERROROCCURRED	Event
RECTYPE	TS20000051	Name of Receiver Type
RECMODE		Indicator for Event Handler

Tabelle 7.2 Einzeleintrag in der Kopplungstabelle SWFDEVTYP mit Beispielwerten

Field name	Example	Description
RECCLASS		Class Name of Handler
RECINTERF		Interface Name of Handler
RECMETH		Name of Handler Method
RECMETYP		Method Type
RECFB	SWE_TEMPLATE_REC_FB	Name of Receiver Function Module
RECFBTYP		Type of Receiver Function Module
CHECKFB	SWE_TEMPLATE_CHECK_FB	Name of Check Function Module
CHKFBTYP		Type of Function Module
RECGETFB	SWE_TEMPLATE_RECTYPE_FB	Name of Receiver Type Function Module
GETFBTYP		Type of Function Module
RFCDEST	WORKFLOW_LOCAL_018	Name of logical RFC destination
DESTTYP		Type for Call Destination
TYPELINK		Flag: Global event linkage activated
QUEUE	X	Event Linkage: Enable Use of Queue
REC_ERROR		Event Linkage: Behavior Upon Error in Receiver
RECCATEG		Workflow: Object Category
RECTYPEID		Type of Objects in Persistent Object References

Tabelle 7.2 Einzeleintrag in der Kopplungstabelle SWFDEVTYP mit Beispielwerten

Listing 7.8 SAP R/3 Standard-Routine zum Auslösen eines Workflow Events

```
REPORT RSWEMC01.
INCLUDE <CNTAIN>.
TABLES: NAST, T681Z.
*---------------------------------------------------------*
*       FORM CREATE_EVENT                                 *
*---------------------------------------------------------*
FORM CREATE_EVENT USING RETURNCODE US_SCREEN.
  INCLUDE RSWUINCL.
  DATA: L_EVENT_CREATOR LIKE SWHACTOR .
```

```
  DATA: L_OBJKEY LIKE SWEINSTCOU-OBJKEY.
  DATA: L_EVENTID LIKE SWEDUMEVID-EVTID.
*
  IF NAST-EVENT EQ SPACE OR NAST-OBJTYPE EQ SPACE.
      RETURNCODE = 4.
  ELSE.
    SWC_CONTAINER L_CONT.
    L_EVENT_CREATOR-OTYPE = ORG_OBJTYPE_USER .
    "global in RSWUINCL
    L_EVENT_CREATOR-OBJID = SY-UNAME.
    "hopefully sy-uname does exist!
* Ereignis absetzen
    L_OBJKEY = NAST-OBJKY.
    CALL FUNCTION 'SWE_EVENT_CREATE'
         EXPORTING
              OBJTYPE           = NAST-OBJTYPE
              OBJKEY            = L_OBJKEY
              EVENT             = NAST-EVENT
              CREATOR           = L_EVENT_CREATOR
         IMPORTING
              EVENT_ID          = L_EVENTID
         TABLES
              EVENT_CONTAINER   = L_CONT
         EXCEPTIONS
              OBJTYPE_NOT_FOUND = 01.

    IF SY-SUBRC <> 0.
      RETURNCODE = 4.
    ELSEIF NOT L_EVENTID IS INITIAL.
      RETURNCODE = 0.
    ELSE.
      RETURNCODE = 4.
    ENDIF.
  ENDIF.
ENDFORM.
```

7.3 Workflow-Handler

Workflow braucht Eventhandler

Nachdem ein Workflow einen Event ausgelöst hat, wird noch ein Programm benötigt, das auf diesen Event reagiert und eine Handlung ausführt. Ein solcher Eventhandler kann in SAP R/3 entweder eine Methode von einem Business-Objekt sein oder ein Funktionsbaustein, der einer bestimmten Parameterkonvention folgt.

Vorlage für Workflow-Handler

Im Falle eines Funktionsbausteins haben wir die gleiche Abarbeitungslogik wie bei einem IDoc, nur dass diesmal keine Nachricht erstellt und abgespeichert, sondern eine Aktion direkt ausgeführt wird. Als Vorlage für das Interface des Funktionsbausteins kann man `SWW_WI_CREATE_VIA_EVENT` nehmen. Der neu angelegte Funktionsbaustein kann dann in die Kopplungstabellen für den Workflow (SWETYPECOU beziehungsweise SWFDEVTYP) eingetragen werden und wird nach Auslösen eines entsprechenden Events vom Funktionsbaustein `SWE_EVENT_CREATE` dynamisch aufgerufen.

Das folgende Listing kann als Vorlage für ein Interface eines Workflow-Handlers dienen.

Listing 7.9 Interface eines typischen Workflow-Handlers

```
FUNCTION SWW_WI_CREATE_VIA_EVENT.
*"----------------------------------------------------
*"*"Lokale Schnittstelle:
*"       IMPORTING
*"             VALUE(EVENT) LIKE  SWETYPECOU-EVENT
*"             VALUE(RECTYPE) LIKE  SWETYPECOU-RECTYPE
*"             VALUE(OBJTYPE) LIKE  SWETYPECOU-OBJTYPE
*"             VALUE(OBJKEY) LIKE  SWEINSTCOU-OBJKEY
*"       TABLES
*"              EVENT_CONTAINER STRUCTURE  SWCONT
*"----------------------------------------------------
```