Janus/IDOC

Tandem-SAP-Kopplung über ALE

Funktionsbeschreibung

Datum: 05.03.1999

Marcus von Cube
B&S GmbH
Borsigstraße 12
D-65205 Wiesbaden
Tel: 06122 / 77 05 0
Fax: 06122 / 77 05 25
marcus_von_cube@csi.com

Inhaltsverzeichnis

1 Einleitung

Der vorliegende Text beschreibt das Produkt Janus/IDOC zur Kopplung von Tandem-NonStop-Servern mit SAP-Systemen über das SAP-Verfahren ALE (Application Link Enabling.) Die bei diesem Verfahren ausgetauschten Dokumente heißen *intermediate Documents* (IDOC) und geben dieser Janus-Variante ihren Namen.

Janus/IDOC basiert auf dem Produkt Janus/TCP und benutzt dessen Transportprotokoll. Auf der Seite von SAP wird nicht CPI-C sondern Remote Function Call (RFC) eingesetzt, wie es durch die ALE-Technik vorgegeben ist.

Die SAP-Dokumente werden auf dem NonStop-Server in einer SQL-Datenbank gespeichert.

1.1 Aufgaben

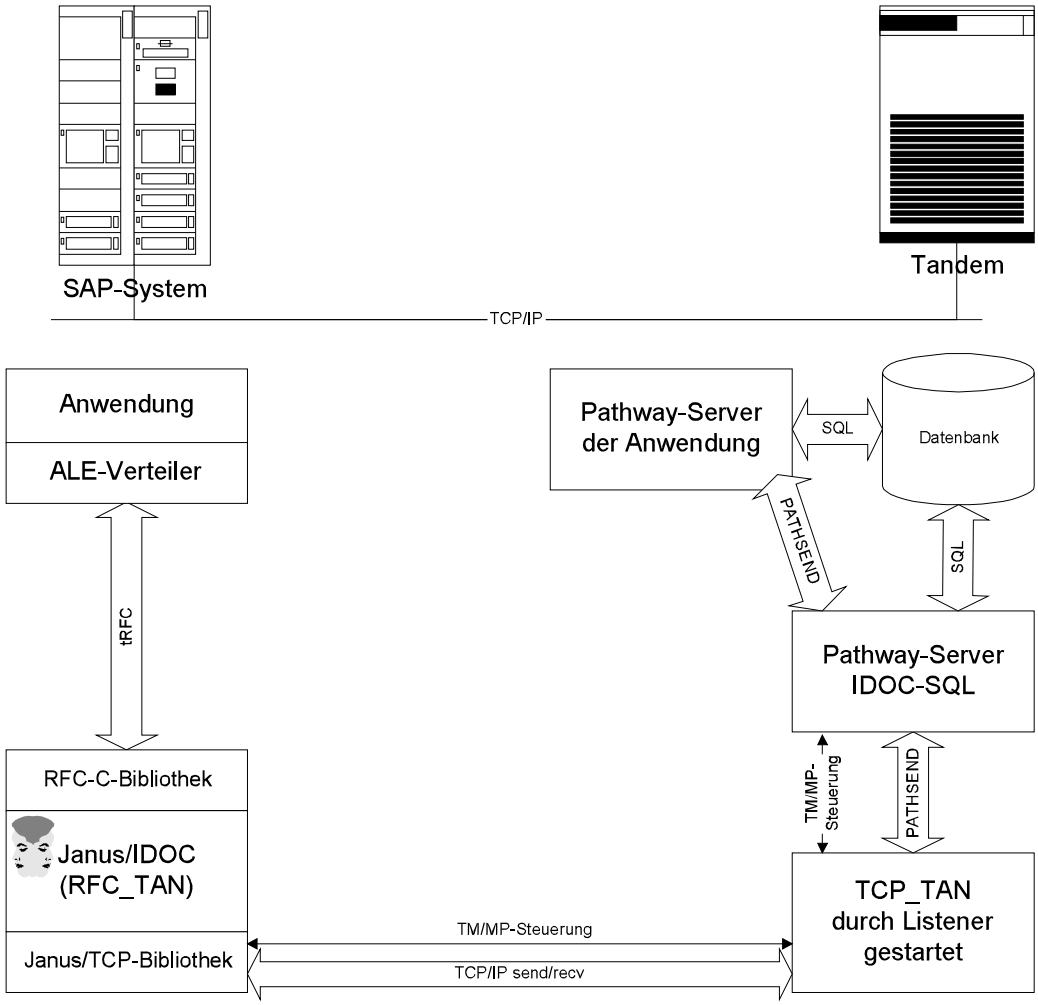
Janus/IDOC besteht aus mehreren Modulen, die im wesentlichen zwei Aufgaben erfüllen:

1. Entgegennahme von IDOCs aus dem SAP-System und Übertragung und Speicherung in einer Tandem-SQL-Datenbank. Durch den Einsatz der Transaktionskontrolle auf beiden Seiten wird die Eindeutigkeit und Sicherheit der Übertragung gewährleistet: SAP sorgt im Zweifel für eine Wiederholung der Übertragung, während TM/MP auf der Tandem-Datenbank garantiert, daß nur vollständig und richtig übertragene Dokumente gespeichert werden.
2. Weitergabe von auf dem Tandem-System in derselben Datenbank abgelegten IDOCs an das SAP-System. In diesem Fall wird durch Verwendung des *transactional RFC* und Speicherung der von SAP generierten *Transaction-ID* (TID) in der Tandem-Datenbank sichergestellt, daß keine unerkannten doppelten Übertragungen erfolgen können. Ein Tandem-Server kontrolliert regelmäßig, ob alle Transportaufträge korrekt durchgeführt wurden und startet solche, bei denen das nicht der Fall war, gegebenenfalls neu.

2 Übertragung von SAP zum Tandem-System

2.1 Übersicht

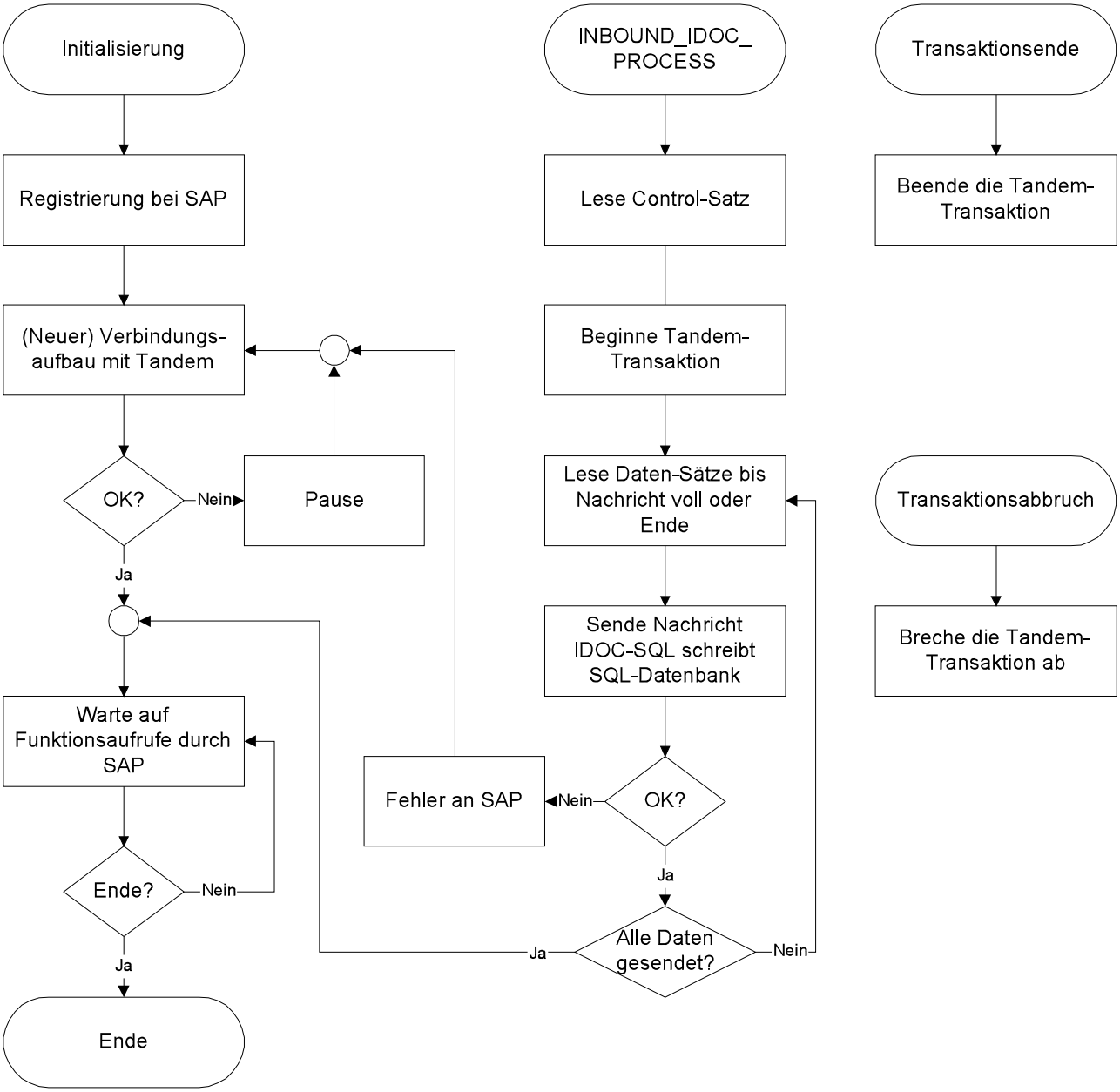
Die folgende Grafik veranschaulicht die Zusammenarbeit der beteiligten Komponenten:



Janus besteht aus dem RFC-Server *rfc_tan* auf einem SAP-Unix-Rechner, dem Tandem-TCP/IP-Server *TCPTAN*, der unverändert von Janus/TCP übernommen wurde und dem Server *IDOC-SQL*, der die übertragenen Dokumente in die SQL-Datenbank einträgt. Letzterer sollte in dem Pathway-System ihrer Tandem-Anwendung eingetragen sein, kann aber auch als separater Prozess gestartet werden.

2.2 Ablauf

Den Ablauf der Übertragung aus Sicht des RFC-Servers *rfc_tan* zeigt das folgende Flußdiagramm:



Der Prozess kann entweder statisch als *registrierter* RFC-Server laufen oder für jede Übertragung vom SAP-Gateway neu gestartet werden. Die dargestellten Einsprungpunkte *INBOUND_IDOC_PROCESS*, *Transaktionsende* und *Transaktionsabbruch* werden während der Initialisierung registriert und dann von der RFC-Bibliothek automatisch aufgerufen. Die Kontrolle der Ereignisse liegt bei SAP bzw. der RFC-Bibliothek.

Der Tandem-Server *IDOC-SQL* sendet beim Empfang eines IDOC bei Bedarf eine Nachricht mit der IDOC-Nummer an einen konfigurierten Server.

2.3 Installation

2.3.1 Installation des Tandem-Gateways TCPTAN

Der Gateway-Prozess TCP_TAN muß auf dem Tandem-System installiert werden. Dazu ist eine Zeile in der Datei PORTCONF der TCP/IP-Konfiguration für den LISTNER einzutragen:

```
3000     $system.janus.tcptan
```

Danach ist der LISTNER-Prozess neu zu starten.

Das ausführbare Programm TCPTAN muß im Binärformat auf den Tandem-Rechner, hier im Subvolume $SYSTEM.JANUS kopiert werden und den Filecode 100 erhalten. Die Portnummer, hier 3000, kann frei gewählt werden.

Eine vorhandene Janus/TCP-Konfiguration kann unverändert weiterbenutzt werden.

2.3.2 Einrichten der Datenbank

Legen Sie mit den SQLCI-Dateien SIDOCDC, SIDOCDD und SIDOCKEY die Tabellen der Datenbank an. Ändern Sie *vorher* die physikalischen Dateinamen der Tabellen und des Kataloges sowie die Extents-Parameter in den Dateien nach Ihren Vorstellungen ab. Die Tabelle IDOCKEY enthält nur einen Satz und braucht daher nicht vergrößert zu werden.

2.3.3 Installation des Servers IDOC-SQL

Legen Sie in einem vorhanden Pathway-System einen Server IDOC-SQL (der Name ist änderbar) mit dem Programm $SYSTEM.JANUS.IDOCSQL an.

Das ausführbare Programm IDOCSQL muß im Binärformat auf den Tandem-Rechner, hier im Subvolume $SYSTEM.JANUS kopiert werden und den Filecode 100 erhalten.

Setzen Sie die TACL-Defines =IDOCDC, =IDOCDD und =IDOCKEY auf Ihre Datenbank-Tabellen und führen Sie SQLCOMP mit dem Server durch.

In der Pathway-Umgebung können Sie noch einige PARAM-Werte für den Server setzen:

```
SET SERVER PARAM NOTIFY-PATHMON $TM01
SET SERVER PARAM NOTIFY-SERVER MYAPP-SERVER
```

An den so definierten Server sendet *IDOC-SQL* eine Nachricht, sobald Daten von SAP eingetroffen sind. Pathmon hat dabei das aktuelle System als Default. Einen Default für den Server gibt es nicht, ohne diese Angabe werden die Daten lediglich in die Tabellen eingearbeitet.

Das Programm legt im Subvolume JANUSLOG des Default-Volumes beim Programmstart die Log-Datei IDOCSQL an, in der der Programmlauf und eventuelle Fehler protokolliert werden. Benutzten Sie das Hilfsprogramm *MORE*, um die Logs zu betrachten. Die Datei sollte von Zeit zu Zeit gelöscht werden.

2.3.4 Installation im SAP unter Unix

Die Installation von *rfc_tan* erfolgt auf einem SAP-Rechner, bevorzugt auf dem Gateway-System. Kopieren Sie die ausführbare Datei *rfc_tan* und das Script *rfc_tan.sh* in das Homeverzeichnis eines Users *janus.* Alle Dateien sollten diesem User gehören.

2.3.4.1 Konfiguration im SAP-System

Tragen Sie das Script */home/janus/rfc_tan.sh* in die SAP-Umgebung als RFC-Server ein (Transaktion *sm59*.) Der Server läuft im sogenannten *Register Mode* als Dämon. Die Program-ID muß mit dem Eintrag PROGID in der Konfigurationsdatei *rfc.ini* übereinstimmen (s. u.)

2.3.4.2 Shell-Script *rfc_tan.sh*

Der Start von *rfc_tan* erfolgt über das folgende Script, in dem Sie die Umgebungsvariablen setzen können:

```
#!/bin/sh
#
PATH=$PATH:/home/janus
export PATH
cd /home/janus

export TRACE=1
export DEBUG=0
export RFC_TRACE=0
export TRACE_FILE=/home/janus/rfc_tan.trc

export RFC_INI=/home/janus/rfc.ini
export HOST=tandem
export PORT=3000
export PATHMON=\$TM01
export SERVER=SQL-IDOC

# Alte Traces loeschen
rm dev_rfc 2>/dev/null

while true;do
    /home/janus/rfc_tan -D SAPIN
    mv $TRACE_FILE $TRACE_FILE.old
    mv rfc_tan.log rfc_tan_log.old
    sleep 10
done
```

Der Hostname muß entweder über einen Eintrag in */etc/hosts* oder über einen Name-Server bekannt sein. Alternativ ist die Angabe der IP-Adresse möglich. Die Portnummer muß dem Eintrag in der PORTCONF-Datei des Tandemsystems für TCPTAN entsprechen. Pathmon und Server korrespondieren mit Ihrer Anwendungsumgebung. Läuft IDOC-SQL nicht innerhalb von Pathway, dann können Sie als Servernamen auch den Prozessnamen angeben, Pathmon wird dann ignoriert. Die Trace und Debug-Einstellungen dienem dem Test der Kommunikation. In der Datei *rfc_tan.log* wird der Status und eventuelle Fehler protokolliert.

Das Backslash-Zeichen vor den Dollarzeichen im Parameter PATHMON ist notwendig, damit die Shell den Tandem-Prozessnamen nicht als Shellvariable interpretiert.

2.3.4.3 Start als registrierter RFC-Server im *Register Mode*

Damit das Programm das SAP-System finden kann, bei dem es sich registriert, müssen Sie zuerst eine Datei *rfc.ini* mit folgendem Inhalt anlegen:

```
DEST=SAPIN
TYPE=R
PROGID=RFC_SAP
GWHOST=<Hostname des SAP-Gateways>
GWSERV=<Servicename des SAP-Gateways: sapgw00>
RFC_TRACE=<0 oder 1>
```

Starten Sie das Script manuell:

```
nohup rfc_tan.sh &
```

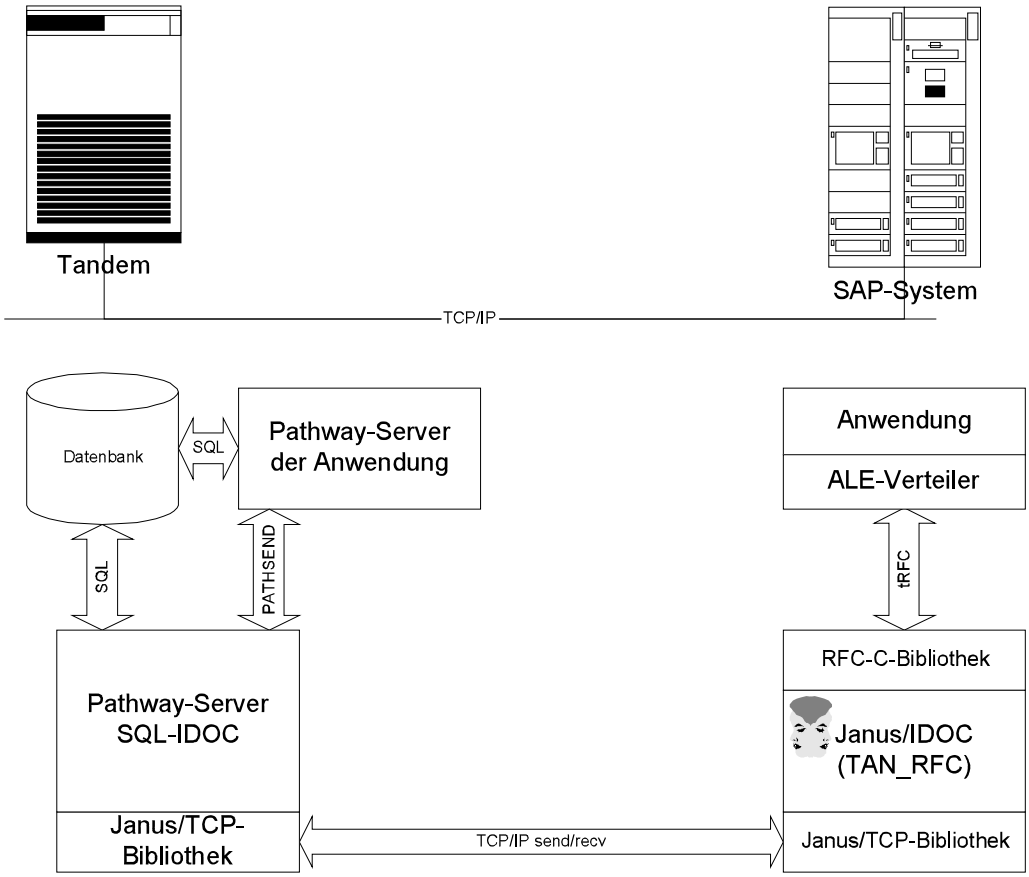
Habe Sie mehrere DEST-Einträge in der Datei, dann können Sie einen davon im Script als Parameter von *rfc_tan* angeben: *-D<dest>* angeben, "SAPIN" ist der Default. Legen Sie für mehrere Ziele mehrere Versionen des Scripts an.

Zum Test können Sie das Script auch direkt starten. Vorher sollten Sie den DEBUG-Parameter auf 1 setzen, dann sehen Sie auf dem Terminal die Aktionen des Servers.

3 Übertragung vom Tandem-System zu SAP

3.1 Übersicht

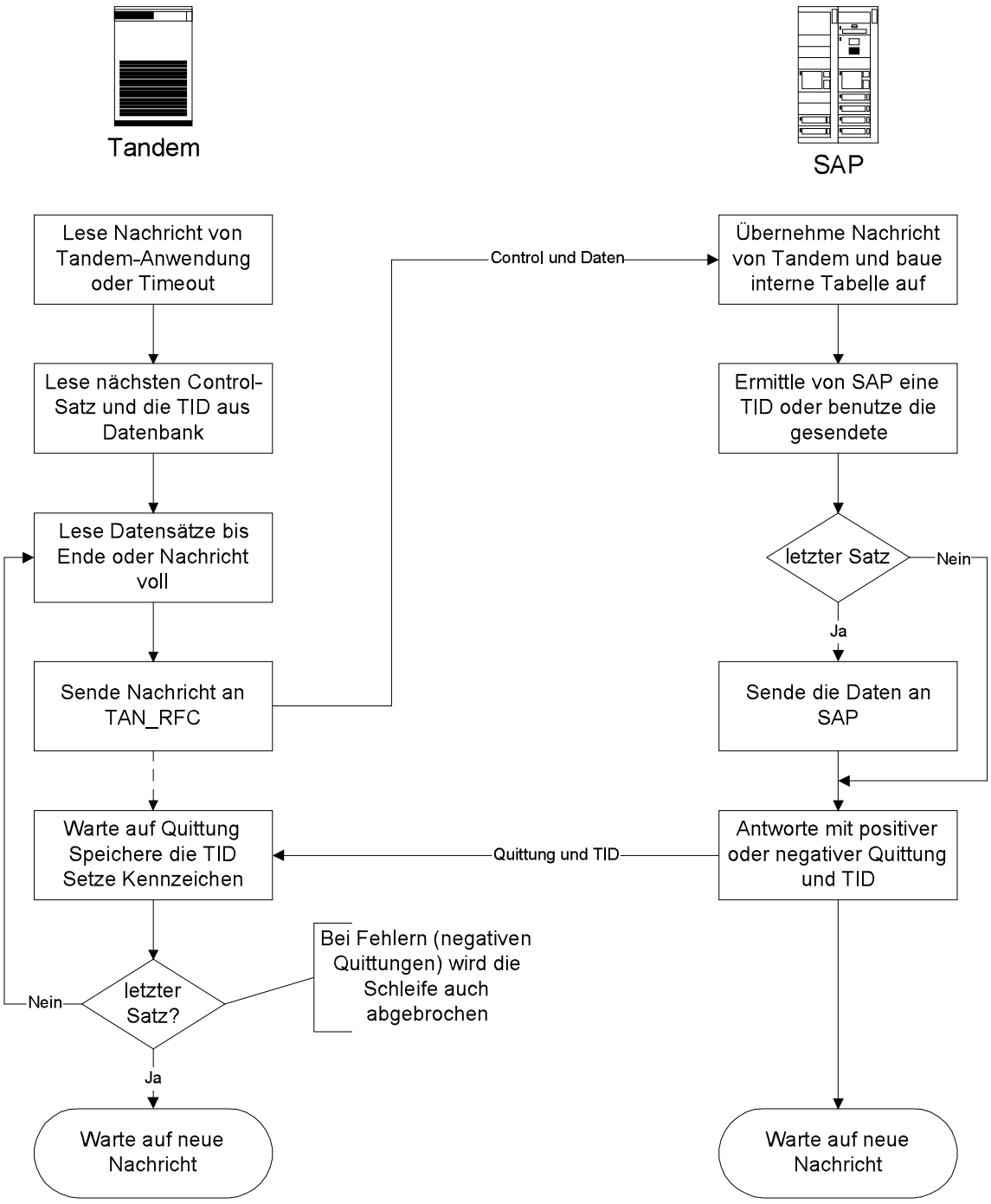
Die folgende Grafik veranschaulicht die Zusammenarbeit der beteiligten Komponenten:



Janus besteht aus dem Tandem-Pathway-Server SQL-IDOC und dem RFC-Client TAN_RFC auf einem SAP-Unix-Rechner. SQL-IDOC entnimmt die Dokumente aus der Datenbank, in die sie von Ihrer Anwendung vorher eingetragen werden müssen. SQL-IDOC sollte in dem Pathway-System ihrer Tandem-Anwendung eingetragen sein.

3.2 Ablauf

Den Ablauf der Übertragung aus Sicht beider Seiten zeigt das folgende Flußdiagramm:



Der Unix-Prozess wird beim Start des Tandem-Prozesses vom *inetd*-Server automatisch gestartet.

Der Tandemprozess wird entweder durch eine Nachricht von außen angestoßen oder wartet ein einstellbares Delay bevor er versucht, alle noch nicht gesendeten Dokumente zu übertragen. Die Kontrolle der Übertragung liegt beim Tandem-Prozess.

3.3 Installation

3.3.1 Einrichten der Datenbank

Die Datenbank ist dieselbe, wie sie von IDOC-SQL benutzt wird.

3.3.2 Installation des Servers SQL-IDOC

Legen Sie in einem vorhanden Pathway-System einen Server SQL-IDOC (der Name ist beliebig) mit dem Programm $SYSTEM.JANUS.SQLIDOC an.

Das ausführbare Programm SQLIDOC muß im Binärformat auf den Tandem-Rechner, hier im Subvolume $SYSTEM.JANUS kopiert werden und den Filecode 100 erhalten.

Setzen Sie die TACL-Defines =IDOCDC, =IDOCDD und =IDOCKEY auf Ihre Datenbank-Tabellen und führen Sie SQLCOMP mit dem Server durch.

In der Pathway-Umgebung müssen Sie noch einige PARAM-Werte für den Server setzen:

```
SET SERVER PARAM HOST saphost
SET SERVER PARAM PORT 11001
SET SERVER PARAM RECEIVER <Name>
SET SERVER PARAM MAX-RETRY <Anzahl>
SET SERVER PARAM DELAY <Sekunden>
```

Wird der Parameter RECEIVER eingegeben, dann sendet der Server nur IDOCs, deren Feld RCVPRN im Control-Satz (Spalte DC_RCVPRN in Tabelle IDOCDC) mit dem angegebenen Namen übereinstimmt. Ohne diese Angabe werden alle IDOCs übertragen.

Die Parameter MAX-RETRY und DELAY legen das Verhalten des Servers bei der Wiederholung oder dem automatischen Start von Übertragungen fest.

Das Delay legt die Zeit zwischen den Übertragungsversuchen fest. Innerhalb eines Versuchs werden alle anstehen Aufträge (bis zum ersten Fehler) in der Reihenfolge ihres Eintrags in die Datenbank ausgeführt. Eine beliebige Nachricht an den Server löst sofort eine Übertragung aus.

3.3.3 Installation im SAP unter Unix

Die Installation von TAN_RFC erfolgt auf einem SAP-Rechner, bevorzugt auf dem Gateway-System. Kopieren Sie die ausführbare Datei *tan_rfc* und das Script *tan_rfc.sh* in das Homeverzeichnis eines Users *janus*. Alle Dateien sollten diesem User gehören. Für den Start durch *inetd* legen Sie noch einen symbolischen Link des Scripts nach *tan_rfc_1.sh* an.

3.3.3.1 Shell-Script *tan_rfc.sh*

Der Start von *rfc_tan* erfolgt über das folgende Script, in dem Sie die Umgebungsvariablen setzen können:

```
#!/bin/sh
#
PATH=$PATH:/home/janus
export PATH
cd /home/janus

export TRACE=1
export DEBUG=0
export RFC_TRACE=0

export RFC_INI=/home/janus/rfc.ini
export DEST=SAPOUT
export USER=JANUS
export PASSWORD=bje
export MANDANT=00
export SPRACHE=D

# Alte RFC-C Traces loeschen
rm dev_rfc 2>/dev/null

/home/janus/tan_rfc $* 2>tan_rfc.out
```

3.3.3.2 Konfigurationsdatei *rfc.ini*

Damit das Programm das SAP-System finden kann, zu dem es die Dokumente senden soll, müssen Sie zuerst eine Datei *rfc.ini* mit folgendem Inhalt anlegen:

```
DEST=SAPOUT
TYPE=A
ASHOST=<Hostname des Application-Servers>
SYSNR=<Nummer des R3-Systems>
GWHOST=<Hostname des SAP-Gateways, wenn es nicht auf ASHOST liegt>
GWSERV=<Servicename des SAP-Gateways: sapgw00, siehe GWHOST>
RFC_TRACE=<0 oder 1>
```

Habe Sie mehrere DEST-Einträge in der Datei, dann können Sie einen davon beim Start des Scripts durch den Parameter DEST im Script angeben, "SAPOUT" ist der Default.

Sie können auch als TYPE "B" für Load Balancing angeben:

```
DEST=SAPOUT
TYPE=B
R3NAME=<Name des R/3-Systems>
MSHOST=<Name des Message-Servers>
GROUP=<Gruppenname des Application-Servers, Default: PUBLIC>
RFC_TRACE=<0 oder 1>
```

Der Servicename des Message-Servers, *sapms<R/3-Systemname>*, muß in */etc/services* eingetragen sein.

3.3.3.3 Dynamischer Start durch den INET-Dämon *inetd*

Folgende Eintragungen sollten in */etc/services* erfolgen:

```
janus1    11001/tcp
janus2    11002/tcp
janus3    11003/tcp
janus4    11004/tcp
janus5    11005/tcp
janus6    11006/tcp
janus7    11007/tcp
janus8    11008/tcp
```

Folgende Eintragungen sollten in */etc/inetd.conf* erfolgen:

```
janus1    stream    tcp  nowait    janus     /home/janus/tan_rfc_1.sh tan_rfc_1 -
janus2    stream    tcp  nowait    janus     /home/janus/tan_rfc_2.sh tan_rfc_2 -
janus3    stream    tcp  nowait    janus     /home/janus/tan_rfc_3.sh tan_rfc_3 -
janus4    stream    tcp  nowait    janus     /home/janus/tan_rfc_4.sh tan_rfc_4 -
janus5    stream    tcp  nowait    janus     /home/janus/tan_rfc_5.sh tan_rfc_5 -
janus6    stream    tcp  nowait    janus     /home/janus/tan_rfc_6.sh tan_rfc_6 -
janus7    stream    tcp  nowait    janus     /home/janus/tan_rfc_7.sh tan_rfc_7 -
janus8    stream    tcp  nowait    janus     /home/janus/tan_rfc_8.sh tan_rfc_8 -
janus9    stream    tcp  nowait    janus     /home/janus/tan_rfc_9.sh tan_rfc_9 -
```

Die Minuszeichen am Ende sind wichtig.

Sie haben jetzt 9 verschieden Scripte für verschiedene SAP-Konfigurationen unter den Portnummern 11001 bis 11009 zur Verfügung. Geben Sie den Scripten sprchende Namen und legen Sie symbolische Links nach *tan_rfc_*x.*sh* an.

3.3.3.4 Start auf der Konsole

Zum Test können Sie das Script auch direkt starten. Setzen Sie vorher den DEBUG-Parameter im Script auf 1:

```
(tan_rfc.sh 11000 &);sleep 2;tail -f tan_rfc.out
```

Der komplizierte Aufruf ist notwendig, da das Script für den Start durch *inetd* gemacht ist, der Ausgaben auf der Konsole in den TCP/IP-Datenstrom umlenken würde, wäre dort nicht die Umleitung *2>tan_rfc.log*.

Der Server ist jetzt unter der Portnummer 11000 erreichbar.

4 Datenstrukturen und Tabellen

Der folgende Abschnitt beschreibt die Organisation der Daten auf dem Tandem-System und die Nachrichten, die zwischen den Servern ausgetauscht werden.

4.1 Data-Dictionary, Quelltexte und Skripte

4.1.1 Allgemeines

Alle Datenstrukturen sind auf dem Tandemsystem in einem DDL-Dictionary gespeichert, zu dem auch eine HTML-Dokumentation existiert. Aus diesem Dictionary werden die Nachrichten und auch die Tabellendefinitionen direkt erzeugt.

4.1.2 Subvolumes

Die Installation belegt die folgenden Subvolumes auf einem installationsabhängigen Volume $<volume>:

Subvolume	*Datei*	*Bemerkung*
Default-Volume [Default-Volume]OBJ	MKSQL u. a.	TACL-Makros und Hilfsprogramme des Autors. Im OBJ-Subvolume liegen auch die neu übersetzten Server.
$<volume>.JANUSDDL	JANUS	DDL-Quelltext.
	CJANUS	COBOL85-COPY-Datei mit allen Datenstrukturen.
	FIDOC*	FUP-Dateien zur Tabellenerzeugung mit MKSQL.
$<volume>.JANUSDIC	DICT*	DDL-Dictionary.
$<volume>.JANUSSRC	JANUSH *C, *H, *O	Die von DDL generierte C-Headerdatei und die Quelltexte der Tandem-Server sowie übersetzte Unterprogramme.
$<volume>.JANUSSQL	SIDOC*	SQLCI-Skripte, die von MKSQL erzeugt wurden.
$<volume>.JANUSDAT	IDOC*	Die von MKSQL angelegten SQL-Tabellen.
$<volume>.JANUSCAT	TABLES, ...	Der SQL-Katalog für die Anwendung.

4.1.3 SQLCI-Skripte

Das TACL-Macro MKSQL erzeugt direkt die SQL-Tabellen aus RECORD-Definitionen des Dictionary und legt sie im Subvolume $<volume>.JANUSDAT an. Für einen produktiven Betrieb sollten aber die SQLCI-Dateien JANUSSQL.SIDOCKEY, JANUSSQL.SIDOCDC und JANUSSQL.SIDOCDD vom Datenbank-Administrator angepaßt werden.

4.1.4 Define-Namen

Den SQL-Tabellen sind die TACL-Define-Namen =IDOCKEY, =IDOCDC und =IDOCDD zugeordnet. Sie müssen vor dem SQLCOMP-Lauf gesetzt werden.

4.2 Struktur der Datenbank

Der folgende Abschnitt beschreibt die Tabellen IDOCKEY, IDOCDC und IDOCDD auf dem Tandem-System. Die Schlüsselfelder jeder Tabelle sind *kursiv* markiert.

4.2.1 Schlüsselvergabe IDOCKEY

Die Tabelle IDOCKEY dient der Vergabe von eindeutigen Schlüsseln für jedes Dokument:

Spalte	***Format***	***Bemerkung***
APPLICATION	CHAR(16)	Schlüsselfeld, hat immer den Wert "Janus/IDOC".
NUMERIC_KEY	PIC 9(8)	Nächster freier Schlüsselwert.

Der Zugriff erfolgt mit folgender Sequenz:

```
begin work;

select numeric_key from =IDOCKEY
 where application = "Janus/IDOC"
   for repeatable access;
```

Es folgen Zugriffe auf die anderen Tabellen mit dem gelesenen Schlüsselwert. Danach wird der Schlüssel weitergezählt und die Transaktion abgeschlossen:

```
update =IDOCKEY
   set numeric_key = case numeric_key
                         when 99999999 then 1
                         else numeric_key + 1
                     end
 where application = "Janus/IDOC";

commit work;
```

Durch die Transaktionskapsel und den Accessmode *repeatable access* bleiben die Tabellen solange gesperrt, bis alle Änderungen durchgeführt sind. Dadurch wird eine doppelte Schlüsselvergabe sicher vermieden. Der CASE-Ausdruck verhindert einen Überlauf beim hundertmillionsten IDOC.

Die Tabelle muß einmalig mit folgendem Befehl initialisiert werden:

```
insert into =IDOCKEY
       ( application,
         numeric_key )
values ( "Janus/IDOC",
         1 );
```

Dies wird von *IDOC-SQL* automatisch beim allerersten Zugriff gemacht.

4.2.2 Control-Sätze IDOCDC

Die Tabelle IDOCDC enthält die Übertragungs-Steuerinformationen für jedes Dokument und den Inhalt des Control-Satzes. Pro Dokument existiert ein Satz in IDOCDC:

Spalte	***Format***	***Bemerkung***
NUMERIC_KEY	PIC 9(8)	Eindeutiger Schlüssel aus IDOCKEY.
TRANSFER_STATUS	CHAR(1)	Status der Übertragung: " " ST-NEW Dokument ist neu. "T" ST-TRANSMITTING Dokument wird gerade übertragen. "R" ST-RETRY Übertragung muß wiederholt werden. "C" ST-COMPLETE Übertragung erfolgreich. "S" ST-FROM-SAP Das Dokument stammt von SAP. "E" ST-ERROR Übertragung nicht möglich. "P" ST_PROCESSING Dokument wird bearbeitet. "D" ST_DONE Bearbeitung ist erfolgt.
CREATED	TIMESTAMP	Zeitpunkt der Erzeugung dieses Dokuments.
TRANSMITTED	TIMESTAMP	Zeitpunkt der Übertragung oder des letzten Versuchs.
RETRY_COUNT	PIC 9(4)	Anzahl der erfolglosen Übertragungsversuche. *SQL-IDOC* vergleicht diesen Zähler mit seinem Parameter MAX-RETRY.
ERROR_TYPE	CHAR(2)	Fehlerkennzeichen: " " ET-NO-ERROR Kein Fehler "GN" ET-GUARDIAN Guardian-Filesystem-Error "SQ" ET-SQL SQL-Fehler "PW" ET-PATHWAY Pathway-Fehler "AP" ET-APPLICATION Anwendungsfehler
ERROR1	PIC S9(5)	Primärer Fehlercode, Bedeutung abhängig von ERROR_TYPE.
ERROR2	PIC S9(5)	Sekundärer Fehlercode, in der Regel ein Guardian-Filesystem-Error.
TRANSACTION_ID	CHAR(24)	Die vom *transactional RFC* vergebene Transaktions-ID.

Die folgenden Spalten sind unverändert aus der Datenstruktur *EDI-DC*, übernommen, die den Control-Satz eines Dokuments beschreibt:

Spalte	***Format***	***Bemerkung***
DC_TABNAM	CHAR(10)	
DC_MANDT	CHAR(3)	
DC_DOCNUM	CHAR(16)	
DC_DOCREL	CHAR(4)	
DC_STATUS	CHAR(2)	
DC_DOCTYP	CHAR(8)	
DC_DIRECT	CHAR(1)	
DC_RCVPOR	CHAR(10)	
DC_RCVPRT	CHAR(2)	
DC_RCVPRN	CHAR(10)	Receive-Port: Kann von *SQL-IDOC* überprüft werden.
DC_RCVSAD	CHAR(21)	
DC_RCVLAD	CHAR(70)	
DC_STD	CHAR(1)	

Spalte	*Format*	*Bemerkung*
DC_STDVRS	CHAR(6)	
DC_STDMES	CHAR(6)	
DC_MESCOD	CHAR(3)	
DC_MESFCT	CHAR(3)	
DC_OUTMOD	CHAR(1)	
DC_TEST	CHAR(1)	
DC_SNDPOR	CHAR(10)	
DC_SNDPRT	CHAR(2)	
DC_SNDPRN	CHAR(10)	
DC_SNDSAD	CHAR(21)	
DC_SNDLAD	CHAR(70)	
DC_REFINT	CHAR(14)	
DC_REFGRP	CHAR(14)	
DC_REFMES	CHAR(14)	
DC_ARCKEY	CHAR(70)	
DC_CREDAT	CHAR(8)	
DC_CRETIM	CHAR(6)	
DC_MESTYP	CHAR(6)	
DC_IDOCTYP	CHAR(8)	
DC_CIMTYP	CHAR(8)	
DC_RCVPFC	CHAR(2)	
DC_SNDPFC	CHAR(2)	
DC_SERIAL	CHAR(20)	

Die Spaltennamen tragen den Präfix *DC*, da einige von SAP vorgegebene Bezeichnungen COBOL-Schlüsselworte sind und deshalb die Generierung von COBOL-Strukturen verhindern würden.

Janus/IDOC wertet diese Felder bis auf DC_RCVPRN nicht direkt aus sondern übernimmt sie verbatim. Sie sind trotzdem als eigene Spalten definiert, um gezielte Abfragen der Tabelle mit SQL zu ermöglichen.

Die Spalte DC_RCVPRN wird von *SQL-IDOC* mit dem Parameter RECEIVER verglichen. Ist der Parameter gesetzt dann werden nur solche Dokumente übertragen, die in DC_RCVPRN den passenden Wert haben. dadurch lassen sich verschiedene SAP-Systeme unterscheiden.

4.2.3 Datensätze IDOCDD

Die Tabelle IDOCDD enthält zu jedem Dokument alle Datensätze oder Segmente:

Spalte	***Format***	***Bemerkung***
NUMERIC_KEY	PIC 9(8)	Eindeutiger Schlüssel aus IDOCDC, der die Beziehung zum Control-Satz herstellt.
RECORD_NUMBER	PIC 9(6)	Laufende Nummer des Satzes für eindeutigen Schlüssel.
TRANSFER_STATUS	CHAR(1)	Kopie des Feldes aus IDOCDD.

Die folgenden Spalten sind unverändert aus der Datenstruktur *EDI-DC*, übernommen, die den Control-Satz eines Dokuments beschreibt:

Spalte	***Format***	***Bemerkung***
DD_TABNAM	CHAR(10)	
DD_MANDT	CHAR(3)	
DD_DOCNUM	CHAR(16)	
DD_SEGNUM	CHAR(6)	
DD_SEGNAM	CHAR(10)	
DD_PSGNUM	CHAR(6)	
DD_HLEVEL	CHAR(2)	
DD_DTINT2	CHAR(2)	
DD_SDATA	CHAR(1000)	Die Segment-Daten, mit SPACE aufgefüllt.

Die Spaltennamen tragen den Präfix *DD* um der Einheitlichkeit mit der Tabelle IDOCDC willen. Namenskonflikte mit COBOL kommen hier nicht vor.

Janus/IDOC wertet diese Felder nicht direkt aus sondern übernimmt sie verbatim. Sie sind trotzdem als eigene Spalten definiert, um gezielte Abfragen der Tabelle mit SQL zu ermöglichen.

Die Spalte RECORD_NUMBER könnte auch aus einer der DD-Spalten übernommen werden, aber um in der Beziehung von SAP unabhängig zu sein, numeriert Janus die Datensätze selber durch.

4.3 Nachrichten

4.3.1 Allgemeines

Die Nachrichten, die zwischen den beteiligten Servern ausgetauscht werden, folgen alle einem gemeinsamen Schema, das durch die Strukturen MSG-COMMAND, MSG-REPLY und MSG-IDOC beschrieben wird.

4.3.1.1 COBOL-Definitionen

Die Struktur kann in COBOL-Programmen mit `COPY MSG-REPLY OF "JANUSDDL.CJANUS"` eingefügt werden. Die Symbolischen Konstanten (REPLY-OK) sind als Level-88-Konstanten deklariert und können mit den COBOL-Befehlen `SET REPLY-OK TO TRUE` oder `IF REPLY-OK THEN` angesprochen werden.

4.3.1.2 C-Definitionen

C-Programme sollten den ganzen Headerfile mit `#include "janus.h"` einfügen und können dann auf die *typedef*-Definition *msg_reply_def* zugreifen. Die Feldnamen sind klein geschrieben und "-" ist durch "_" ersetzt. Die symbolischen Konstanten (REPLY-OK) sind als Makros in Großbuchstaben mit "_" (`REPLY_OK`) definiert.

4.3.2 Die Struktur der Nachrichten

4.3.2.1 Gemeinsame Felder

Feld	*Format*	*Bemerkung*
REPLY-CODE	NATIVE-2	Binärer Reply-Code mit folgenden Bedeutungen: 0 REPLY-OK Alles in Ordnung. 2 REPLY-NOT-OK Fehler aufgetreten. Die Codes haben nur in Antworten eine Bedeutung. In Nachrichten wird REPLY-CODE ignoriert und sollte 0 sein.
MSG-TYPE	PIC XX	Art der Nachricht: OK MT-OK Positive Quittung ohne Daten. ER MT-ERROR Fehlermeldung: REPLY-CODE ist ungleich Null. DA MT-DATA Die Nachricht enthält Daten. DL MT-DATA-LAST Letzer Datenblock eines Dokuments. GK MT-GET-KEY Anfrage zur Erzeugung eines neuen Schluessels aus IDOCKEY. CT MT-CHECK_TID Prüfe die übergebene Transaktions-ID auf Vorhandensein in der Datenbank.
TIMESTAMP	PIC X(16)	Zeitstempel (JJJJMMTThhmmsshs).
TRANSACTION-ID	PIC X(14)	tRFC-Transaktions-ID von SAP.

(MSG-COMMAND endet hier.)

NUMERIC-KEY	PIC 9(8)	Datenbank-Schlüssel des betroffenen Dokuments.

Die Nachrichten MSG-REPLY und MSG-IDOC unterscheiden sich ab hier.

4.3.2.2 MSG-REPLY

Die folgenden Felder kommen nur in MSG-REPLY vor, sie schließen direkt an NUMERIC-KEY an:

Feld	*Format*	*Bemerkung*
ERROR-TYPE	PIC XX	Fehlerkennzeichen: " " ET-NO-ERROR Kein Fehler aufgetreten. GN ET-GUARDIAN Guardian-Filesystem-Error. SQ ET-SQL SQL-Fehler (negativ!) PW ET-PATHWAY PATHSEND-Fehler. AP ET-APPLICATION Anwendungsfehlercode.
ERROR1	PIC S9(5)	Hauptfehlercode, abhängig vom Fehlerkennzeichen.
ERROR2	PIC S9(5)	Nebenfehlercode, abhängig vom Fehlerkennzeichen, oft ein Guardian-Filesystem-Error.
ERROR-MESSAGE	PIC X(256)	Fehlermeldung im Klartext.

4.3.2.3 MSG-IDOC

Die folgenden Felder kommen nur in MSG-DATA vor, sie schließen direkt an NUMERIC-KEY an:

Feld	*Format*	*Bemerkung*
TRANSFER-STATUS	PIC X	Status des Dokuments, wie er in die Datenbank eingetragen werden soll. SAP überträgt hier immer "S".
FILLER	PIC X	Affüllung auf Wortgrenze.
RECORD-COUNT	PIC 9(4)	Anzahl der folgenden Datensätze.
MESSAGE-DATA	Variabel	Daten-Records, Anzahl ist abhängig von RECORD-COUNT. Die folgenden Felder kommen pro Record einmal vor.
DATA-TYPE	PIC XX	Datentyp: DC DT-EDI-DC Control-Satz. DD DT-EDI-DD Datensatz oder Segment.
DATA-LENGTH	PIC 9(4)	Länge dieses Records.
DATA-RECORD	Variabel	Die Daten des Records. Bei DD-Records werden Leerzeichen am Ende abgeschnitten.

Das Bearbeiten der Datensätze ist wegen ihrer variablen Länge besonders in COBOL nicht ganz einfach. Wird das Feld RECORD-COUNT auf 1 gesetzt, kann die Struktur MSG-DATA direkt benutzt werden, der Vorteil der Blockung geht dann allerdings verloren.

4.3.3 Verwendung der Nachrichten

4.3.3.1 MSG-REPLY

Die Antwort-Nachricht wird von allen Servern als positive oder negative Quittung auf eine Nachricht erzeugt. Bei negativen Quittungen vom Typ MT-ERROR ist REPLY-CODE ungleich Null und die Fehlerfelder sind ausgefüllt. Sendet ein Requester eine Nachricht mit WRITE statt WRITEREAD und es entsteht ein Fehler, dann erzeugt der Server statt der Antwort-Nachricht einen Guardian-Filesystem-Error durch entsprechenden Aufruf von REPLY.

4.3.3.2 MSG-COMMAND

Die Kommando-Nachrichten dienen dazu, den Servern Befehle zu senden. Bisher sind nur zwei Nachrichtentypen definiert.

Eine Nachricht MT-GET-KEY, die nur den REPLY-CODE und den TYP "GK" enthalten muß, wird vom Tandem-Server *IDOC-SQL* mit MT-OK und einem gültigen Feld NUMERIC-KEY beantwortet, das für die Neuanlage eines Dokuments in der Datenbank genutzt werden kann. Der Requester sollte vor dem Aufruf eine Transaktion beginnen und diese erst beenden, wenn er alle Daten in der Datenbank gespeichert hat.

Eine Nachricht vom Typ MT-CHECK-TID wird von *rfc_tan* an *IDOC-SQL* im Rahmen der Übertragung gesendet.

Der Server SQL-IDOC reagiert auf Messages aller Art nur mit einem "Aufwachen": Die Datenbank wird erneut nach nicht übertragenen Daten durchsucht.

4.3.3.3 MSG-IDOC

Die Daten-Nachrichten vom Typ MT-DATA bzw. MT-DATA-LAST dienen der Übertragung von Dokumenten:

- Der Tandem-Server *SQL-IDOC* sendet Dokumente aus der Datenbank in diesem Format and den RFC-Client *tan_rfc* für die Weitergabe an SAP.
- Das RFC-Programm *rfc_tan* sendet so die von SAP erzeugten Dokumente über das Netzwerk an den Tandem-Server *IDOC-SQL*.
- *IDOC-SQL* sendet eine solche Nachricht ohne Daten aber mit sonst gültigen Werten und mit Kennzeichen MT-DATA-LAST an den als NOTYFY-SERVER konfigurierten Prozess und erwartet eine Nachricht vom Typ MSG-REPLY mit Reply-Code 0 als Antwort. Andere Antworten, speziell Fehlermeldungen im Reply führen zum Abbruch der Transaktion!
- Die Applikation kann neue Dokumente mit dem Transferstatus ST-NEW an *IDOC-SQL* senden, der Sie dann in die Datenbank einarbeitet. Dies ist eine Alternative zu direkten SQL-Zugriffen in die Datenbank.

Der Datenbereich ist variabel lang. Er enthält die Anzahl der Records und entsprechend viele Datensätze, die selbst wieder verschieden lang sein können. Bei Datensätzen (DC) werden Leerzeichen am Ende abgeschnitten bzw. ergänzt, um Platz in der Nachricht zu sparen.