

# RFC-Programmierung in ABAP



**Release 4.6C**



## Copyright

© Copyright 2001 SAP AG. Alle Rechte vorbehalten.

Weitergabe und Vervielfältigung dieser Publikation oder von Teilen daraus sind, zu welchem Zweck und in welcher Form auch immer, ohne die ausdrückliche schriftliche Genehmigung durch SAP AG nicht gestattet. In dieser Publikation enthaltene Informationen können ohne vorherige Ankündigung geändert werden.

Die von SAP AG oder deren Vertriebsfirmen angebotenen Software-Produkte können Software-Komponenten auch anderer Software-Hersteller enthalten.

Microsoft<sup>®</sup>, WINDOWS<sup>®</sup>, NT<sup>®</sup>, EXCEL<sup>®</sup>, Word<sup>®</sup>, PowerPoint<sup>®</sup> und SQL Server<sup>®</sup> sind eingetragene Marken der Microsoft Corporation.

IBM<sup>®</sup>, DB2<sup>®</sup>, OS/2<sup>®</sup>, DB2/6000<sup>®</sup>, Parallel Sysplex<sup>®</sup>, MVS/ESA<sup>®</sup>, RS/6000<sup>®</sup>, AIX<sup>®</sup>, S/390<sup>®</sup>, AS/400<sup>®</sup>, OS/390<sup>®</sup> und OS/400<sup>®</sup> sind eingetragene Marken der IBM Corporation.

ORACLE<sup>®</sup> ist eine eingetragene Marke der ORACLE Corporation.

INFORMIX<sup>®</sup>-OnLine for SAP und Informix<sup>®</sup> Dynamic Server<sup>™</sup> sind eingetragene Marken der Informix Software Incorporated.

UNIX<sup>®</sup>, X/Open<sup>®</sup>, OSF/1<sup>®</sup> und Motif<sup>®</sup> sind eingetragene Marken der Open Group.

HTML, DHTML, XML, XHTML sind Marken oder eingetragene Marken des W3C<sup>®</sup>, World Wide Web Consortium, Massachusetts Institute of Technology.

JAVA<sup>®</sup> ist eine eingetragene Marke der Sun Microsystems, Inc.

JAVASCRIPT<sup>®</sup> ist eine eingetragene Marke der Sun Microsystems, Inc., verwendet unter der Lizenz der von Netscape entwickelten und implementierten Technologie.

SAP, SAP Logo, R/2, RIVA, R/3, ABAP, SAP ArchiveLink, SAP Business Workflow, WebFlow, SAP EarlyWatch, BAPI, SAPPHIRE, Management Cockpit, mySAP.com Logo und mySAP.com sind Marken oder eingetragene Marken der SAP AG in Deutschland und vielen anderen Ländern weltweit. Alle anderen Produkte sind Marken oder eingetragene Marken der jeweiligen Firmen.

## Symbole

Symbol	Bedeutung
	Achtung
	Beispiel
	Hinweis
	Empfehlung
	Syntax

## Inhalt

<b>RFC-Programmierung in ABAP</b> .....	<b>5</b>
RFC-Grundlagen.....	6
Die RFC-Schnittstelle .....	7
RFC in SAP-Systemen .....	9
Technische Voraussetzungen.....	12
RFC-Funktionsbausteine in ABAP aufrufen .....	14
Einführung .....	15
Parameter in RFC-Aufrufen .....	17
RFC-Funktionen lokal aufrufen.....	18
RFC-Funktionen zurückrufen.....	19
Transaktionale RFC-Aufrufe.....	20
Transaktionale Integrität von tRFCs .....	23
<b>qRFC mit Ausgangsqueue</b> .....	<b>26</b>
qRFC mit Ausgangsqueue im Überblick .....	28
Serialisierung programmieren.....	30
Transaktionsreihenfolge und Queue-Belegung .....	34
Hilfsmittel.....	37
<b>Asynchrone RFC-Aufrufe</b> .....	<b>40</b>
Aufrufanforderungen für asynchrone RFCs .....	42
Ergebnisse von einem asynchronen RFC erhalten .....	43
Entfernte Kontexte aufrechterhalten .....	46
Parallelverarbeitung mit asynchronem RFC .....	47
<b>Berechtigungen für RFC prüfen</b> .....	<b>51</b>
<b>Vordefinierte Ausnahmen für RFCs verwenden</b> .....	<b>52</b>
<b>RFC-Funktionsbausteine in ABAP schreiben</b> .....	<b>53</b>
<b>Schritte bei der Implementierung von RFC-Funktionsbausteinen</b> .....	<b>54</b>
<b>Programmierungshilfen</b> .....	<b>55</b>
<b>RFC-Funktionsbausteine im Debugger</b> .....	<b>57</b>
<b>Entfernte Destinationen pflegen</b> .....	<b>58</b>
<b>Destinationen anzeigen, pflegen und testen</b> .....	<b>59</b>
<b>Destinationsparameter eingeben</b> .....	<b>61</b>
<b>Destinationstypen</b> .....	<b>63</b>
<b>Gruppendestinationen pflegen</b> .....	<b>66</b>
<b>Trusted System: Vertrauensbeziehungen zwischen R/3-Systemen pflegen</b> .....	<b>67</b>

## RFC-Programmierung in ABAP

---

**RFC-Grundlagen**

## RFC-Grundlagen

Im folgenden erhalten Sie einen kurzen Überblick über den Remote Function Call (RFC) in einem SAP-System, d.h.

- wie die RFC-Schnittstelle funktioniert.
- welche Funktionalität RFC bietet.
- welche technischen Anforderungen für RFC unter R/2, R/3 und externen Systemen für alle gegenwärtig unterstützten Plattformen bestehen.

Sie finden diese Informationen in folgenden Abschnitten:

[Die RFC-Schnittstelle \[Seite 7\]](#)

[RFC in SAP-Systemen \[Seite 9\]](#)

[Technische Voraussetzungen \[Seite 12\]](#)

## Die RFC-Schnittstelle

Ein "entfernter Funktionsaufruf" (Remote Function Call) ist der Aufruf eines Funktionsbausteins, der in einem anderen System läuft als das aufrufende Programm. Sie können einen Funktionsbaustein zwar auch im selben System als RFC aufrufen; normalerweise werden RFCs jedoch verwendet, wenn Aufrufer und aufgerufener Funktionsbaustein in unterschiedlichen Systemen laufen.

Im SAP-System stellt das RFC-Schnittstellensystem diese Funktionalität bereit. Das RFC-Schnittstellensystem ermöglicht Funktionsaufrufe zwischen zwei SAP-Systemen (R/2 oder R/3) oder zwischen einem SAP-System und einem externen System (nicht SAP).

Das RFC-Schnittstellensystem besteht aus folgenden Schnittstellen:

- Aufrufchnittstelle für ABAP-Programme

Jedes ABAP-Programm kann einen entfernten Funktionsbaustein mithilfe der Anweisung CALL FUNCTION...DESTINATION aufrufen. Der Parameter DESTINATION teilt dem SAP-System mit, daß der gerufene Funktionsbaustein in einem anderen System läuft als der Aufrufer. Die RFC-Kommunikation mit dem entfernten System findet als Teil der CALL FUNCTION-Anweisung statt.

RFC-Funktionsbausteine in einem SAP-System müssen echte Funktionsbausteine und in dem SAP-System als "entfernt" (remote) registriert sein.

Wenn sowohl Aufrufer als auch aufgerufenes Programm ABAP-Programme sind, stellt die RFC-Schnittstelle beide Kommunikationspartner. Der Aufrufer kann jedes beliebige ABAP-Programm sein, während das *gerufene* Programm ein als entfernt registrierter Funktionsbaustein sein muß.

- Detaillierte Informationen zum Aufruf als entfernt registrierter Funktionsbaustein finden Sie unter [RFC-Funktionsbausteine in ABAP aufrufen \[Seite 14\]](#).
- Informationen zum Schreiben von Funktionsbausteinen, die Sie entfernt aufrufen wollen, finden Sie unter [RFC-Funktionsbausteine in ABAP schreiben \[Seite 53\]](#).

- Aufrufchnittstellen für Nicht-SAP-Programme

Wenn entweder der Aufrufer oder der aufgerufene Partner ein Nicht-SAP-Programm ist, muß dieses Programm so programmiert werden, daß es den anderen Partner in einer RFC-Kommunikation spielen kann.

Um Sie bei der Implementierung von RFC-Partnerprogrammen in Nicht-SAP-Systemen zu unterstützen, liefert SAP:

- [Externe Schnittstellen \[Extern\]](#)

RFC- und GUI-gestützte Schnittstellen können von externen Programmen benutzt werden, um Funktionsbausteine in SAP-R/2- oder SAP-R/3-Systemen aufzurufen und in diesen Systemen auszuführen. Umgekehrt können auch ABAP-Programme in R/2 oder R/3 die von externen Programmen bereitgestellten Funktionen über diese Schnittstellen nutzen.



Wenn Sie einige Szenarien anhand eines konkreten Programmbeispiels kennenlernen möchten, lesen Sie die entsprechende Lektion im [Tutorial zu Kommunikationsschnittstellen \[Extern\]](#).

Die RFC-Schnittstelle

## RFC in SAP-Systemen

In allen R/3-Systemen ist **CALL FUNCTION** ein integraler Bestandteil der ABAP-Sprache (in R/2 ab Release 5.0). Diese Anweisung führt eine Funktion (einen Funktionsbaustein) im selben System aus (R/2 oder R/3).

**REMOTE FUNCTION CALL (RFC)** ist eine Erweiterung von CALL FUNCTION in einer dezentralen Umgebung. Bestehende Funktionsbausteine können über einen RFC-Aufruf aus einem entfernten System (R/2 oder R/3) heraus ausgeführt werden. Dazu fügen Sie einfach einen DESTINATION-Parameter zu der CALL FUNCTION-Anweisung hinzu:



Der DESTINATION-Parameter zeigt einen Eintrag der Tabelle RFCDES an (die mit Transaktion SM59 definiert wird). Dieser Eintrag enthält alle Parameter, die Sie benötigen, um eine Verbindung zum Zielsystem aufzubauen und sich dort anzumelden.

## RFC in SAP-Systemen

Sie können **RFC** zwischen zwei SAP-Systemen verwenden (R/3- und R/2-Systeme). Bei einem R/2-System in einer IBM-Umgebung können Sie RFC gegenwärtig nur unter CICS als DC-System ab Release 5.0D verwenden. In IMS-Umgebung wird RFC nicht vor IMS >= 4.1 und nur bei kompletter Unterstützung des LU6.2-Protokolls (über MVS/APPC) verfügbar sein.

Mit Hilfe der **RFC-Bibliothek (RFC API)** unter OS/2, Windows, Windows NT und allen auf R/3 basierenden UNIX-Plattformen können Sie die RFC-Funktionalität auch zwischen einem SAP-System (R/3 ab Release 2.1 und R/2 ab Release 5.0D) und einem C-Programm auf einer der oben genannten Plattform verwenden. Dabei ist es für den Aufrufer unerheblich, ob der entfernte Funktionsbaustein in einem SAP-Programm oder einem C-Programm läuft.

RFC entbindet den ABAP-Programmierer davon, seine eigenen Kommunikationsroutinen schreiben zu müssen. Bei einem RFC-Aufruf, übernimmt die RFC-Schnittstelle

- die Konvertierung aller Parameterdaten in die im entfernten System benötigte Darstellung. Dazu gehören auch die Konvertierung von Zeichenketten und alle erforderlichen Hardware-abhängigen Konvertierungen (z.B. Integer, Fixpunkt). Alle ABAP-Datentypen werden unterstützt.



Dictionary-Strukturen werden nicht unterstützt.

- den Aufruf der Kommunikationsroutinen, die benötigt werden, um mit dem entfernten System zu kommunizieren.
- die Handhabung der bei der Kommunikation auftretenden Fehler und, falls erwünscht, die Benachrichtigung des Aufrufers. (Der Aufrufer kann über den Parameter EXCEPTIONS der CALL FUNCTION-Anweisung festlegen, ob er benachrichtigt werden möchte.)

Die RFC-Schnittstelle ist für den ABAP-Programmierer unsichtbar. Die Verarbeitungsschritte für den *Aufruf* entfernter Programme sind in die CALL FUNCTION-Anweisung integriert. Die Verarbeitungsschritte für das *Aufgerufenwerden* werden für jeden als entfernt registrierten Funktionsbaustein automatisch generiert (in Form eines RFC-Stub). Dieses Stub-Programm dient dann als Schnittstelle zwischen aufrufendem Programm und Funktionsbaustein.

SAP unterscheidet zwischen einem RFC-Client und einem RFC-Server. Der **RFC-Client** ist die Instanz, die die RFC-Schnittstelle aufruft, um den Funktionsbaustein auszuführen, der von einem **RFC-Server** bereitgestellt wird. Im folgenden werden die Funktionsbausteine, die entfernt ausgeführt werden können, als **RFC-Funktionen** bezeichnet, und die über die RFC-Bibliothek bereitgestellten Funktionen als **RFC-Aufrufe**.

Alle in einem entfernten RFC-Server-System verfügbaren RFC-Funktionen, die von einem RFC-Client aufgerufen werden, werden **transaktional** verarbeitet. D.h. nach Ausführung der ersten RFC-Funktion im RFC-Server-System steht der gesamte Kontext (alle global definierten Variablen im RFC-Server-Programm oder im Hauptprogramm eines Funktionsbausteins) für weitere RFC-Funktionen zur Verfügung. Die RFC-Verbindung wird nur in folgenden Fällen geschlossen:

- Wenn der Kontext des aufrufenden ABAP-Programms beendet ist.
- Wenn im externen Programm explizit *RfcAbort* oder *RfcClose* verwendet wird.

Bis Release 3.0 muß die Verbindung zu einem R/3-System über einen zuvor definierten Anwendungs-Server hergestellt werden. Ab Release 3.0 können Sie einen Anwendungs-Server auch im Rahmen eines Belastungsausgleichs über einen Message-Server zuordnen. Dies gilt

## RFC in SAP-Systemen

sowohl für RFCs zwischen zwei R/3-Systemen als auch für RFCs zwischen einem externen und einem R/3-System.

Um die Ausführung von RFC-Funktionen verlässlich, sicher und von der Verfügbarkeit des RFC-Servers oder RFC-Server-Systems unabhängig zu machen, wurde ab Release 3.0 für R/3-Systeme der **transaktionale RFC (tRFC)** eingeführt. Der tRFC stellt sicher, daß der gerufene Funktionsbaustein **nur einmal** im RFC-Server-System ausgeführt wird.

Bei transaktionalen RFC-Aufrufen müssen die zu einer RFC-Funktion gehörenden Daten zunächst auf einer SAP-Datenbank im RFC-Client-System zwischengespeichert werden. Nach erfolgter Verarbeitung müssen sie dem rufenden ABAP-Programm zurückgemeldet werden. Um alles andere kümmert sich die tRFC-Komponente im R/3-System.

Da bei externen Systemen eine Datenbank nicht immer verfügbar ist, ist die Verbindung zu den tRFC-Schnittstellen so implementiert, daß die auf RFC API basierenden Client- oder Server-Programme einige zusätzliche Verwaltungsfunktionen übernehmen müssen, um sicherzustellen, daß der entsprechende Funktionsbaustein nur einmal ausgeführt wird.



In einem R/3-System können andere R/3-Systeme als Trusted Systeme (Vertrauenssysteme) definiert werden. Trusted Systeme können ohne Paßwort auf das aufgerufene (das vertrauende) Trusting System zugreifen. Weitere Informationen finden Sie unter [Trusted System \[Seite 67\]](#).

## Technische Voraussetzungen

## Technische Voraussetzungen

### Externe Systeme

Die externen Systeme müssen TCP/IP unterstützen.

• OS/2:	TCP/IP für OS/2 von IBM.
• Windows 3.1/3.11:	Alle TCP/IP-Produkte, die die Basisschnittstelle unterstützen.
• Windows NT/95:	Microsoft-Standard
• UNIX-Plattformen:	Herstellerstandard

RFCSDK für die betreffenden Plattformen enthält die folgenden Bibliotheken und Include-Dateien:

• sa prf c. h	Diese Include-Datei enthält alle benötigten Datentypen und Strukturen sowie die Prototypen (Deklarationen) der RFC-Aufrufe.
• sa pit ab .h	Diese Include-Datei enthält alle zur Bearbeitung interner Tabellen erforderlichen RFC-Aufrufe.
• lib rfc	Je nach Plattform sind folgende Bibliotheken erforderlich:
	<b>OS/2:</b> <b>librfc.dll</b> und <b>librfc.lib</b> für Compile/Link
	<b>Windows 3.1/3.11:</b> <b>librfc16.dll</b> , <b>librfc2.dll</b> , <b>librfc3.dll</b> , <b>librfc4.dll</b> und <b>librfc5.dll</b> und <b>librfc16.lib</b> für Compile/Link
	<b>Windows NT/95:</b> <b>librfc32.dll</b> und <b>librfc32.lib</b> für Compile/Link
	<b>UNIX-Plattformen:</b> <b>librfc.a</b>

### SAP-R/3-Systeme

Für RFC-Aufrufe zwischen einem externen System und R/3 gibt es im R/3-System keine besonderen Anforderungen, außer daß das R/3-System das Release  $\geq 2.1$  haben muß.

Im Gegensatz dazu muß bei **RFC zwischen SAP-R/2 in einer IBM-Umgebung und SAP-R/3 oder externen Systemen** ein SAP-Gateway auf einer Maschine laufen, die das SNA LU6.2-Protokoll für den IBM-Host unterstützt. Das SNA-Produkt muß auch auf dieser Maschine installiert sein und der SAP-Gateway muß mit dem Produkt laufen. Dies ist wichtig, da manche SNA-Produkte auf derselben Maschine nicht kompatibel sind.

Folgende SNA-Produkte werden zur Zeit unterstützt:

**Technische Voraussetzungen**

- SNA Services oder SNA Server unter IBM-AIX-Systemen
- SNAplusLink unter HP-UX-Systemen
- Communication Manager unter OS/2
- SNA Server unter WindowsNT-Systemen
- SNAlink SNA peer-to-peer 8.0 unter SUN-Systemen
- TRANSIT-SERVER und TRANSIT-CPIC unter SNI-SINIX-Systemen.

**SAP-R/2-Systeme**

- **IBM-Host (CICS):** Release 5.0D mit folgenden Komponenten:
  - 082  
Kommunikation über Remote Function Call (RFC)
  - 153  
SAP Intersystem Communication
  - 080  
Host-Kommunikation mit DOS, OS/2
  - 081  
Host-Kommunikation mit anderen LU6.2-Systemen
- **IBM-Host (IMS):** Wahrscheinlich nicht vor IMS >= 4.1 mit MVS/APPC
- **SNI-Host:** Release 5.0D mit folgenden Komponenten:
  - 082  
Kommunikation über Remote Function Call (RFC)
  - 153  
SAP Intersystem Communication
  - 083  
Host-Kommunikation über TCP/IP (BS2000)

## RFC-Funktionsbausteine in ABAP aufrufen

Dieser Abschnitt enthält folgende Themen:

[Einführung \[Seite 15\]](#)

[Parameter in RFC-Aufrufen \[Seite 17\]](#)

[RFC-Funktionen lokal aufrufen \[Seite 18\]](#)

[RFC-Funktionen zurückrufen \[Seite 19\]](#)

[Transaktionale RFC-Aufrufe \[Seite 20\]](#)

[Asynchrone RFC-Aufrufe \[Seite 40\]](#)

[Berechtigungen für RFC prüfen \[Seite 51\]](#)

[Vordefinierte Ausnahmen für RFCs verwenden \[Seite 52\]](#)

## Einführung

Um entfernte Funktionsbausteine aufzurufen, verwenden Sie die Anweisung CALL FUNCTION genauso wie zum Aufruf lokaler Funktionsbausteine. Allerdings müssen Sie zusätzlich ein DESTINATION-Parameter in die Anweisung aufnehmen, in dem Sie definieren, wo die Funktion laufen soll:

```
CALL FUNCTION EntfernteFunktion
```

```
DESTINATION Dest
```

EXPORTING	F1 =...
	F2 =...
IMPORTING	f3 =...
TABLES	t1 =...

```
EXCEPTIONS.....
```

Das Feld *Dest* kann entweder ein Literal oder eine Variable sein; sein Wert ist die dem lokalen SAP-System bekannte logische Destination (z.B. "hw1071\_53"). Logische Destinationen definieren Sie in der Tabelle RFCDES (oder in R/2-Systemen in der Tabelle TRFCD) mit der Transaktion SM59 oder über folgenden Menüpfad: *Werkzeuge* → *Administration, Verwaltung* → *Netzwerk* → *RFC Destinationen*. Auf lokale Destinationen können Sie auch über den Einführungsleitfaden (IMG) zugreifen. Wählen Sie *Werkzeuge* → *Business Engineer* → *Customizing, Grundfunktionen* → *Unternehmens-IMG anzeigen*. Aus der Struktur wählen Sie *Anwendungsübergreifende Komponenten* → *Verteilung (ALE)* → *Kommunikation* → *RFC-Destination definieren*.

Mit RFC können Sie z.B. aus einem Program in einem R/3-System heraus einen Funktionsbaustein in einem R/2-System aufrufen. Wenn Sie einen Kundensatz aus Ihrer R/2-Datenbank lesen wollen, legen Sie dazu in der R/2-Umgebung einen entfernt aufrufbaren Funktionsbaustein an, der Kundensätze liest. Dann rufen Sie diesen Funktionsbaustein aus Ihrem R/3-System mit RFC auf und geben die Destination für das R/2-Zielsystem an:

R/3-System: Client

```
CALL FUNCTION 'RFC_CUSTOMER_GET'
```

DESTINATION 'K50'	
EXPORTING	KUNNR = CUSTNO
TABLES	CUSTOMER_T = ITAB
EXCEPTIONS	NO_RECORD_FOUND = 01.

R/2-System: Server

```
FUNCTION RFC_CUSTOMER_GET.
....      (Kundensatz lesen)
ENDFUNCTION.
```

Richtlinien zum Programmieren von RFC-Aufrufen finden Sie unter:

## Einführung

[Parameter in RFC-Aufrufen \[Seite 17\]](#)

[RFC-Funktionen lokal aufrufen \[Seite 18\]](#)

[RFC-Funktionen zurückrufen \[Seite 19\]](#)

[Transaktionale RFC-Aufrufe \[Seite 20\]](#)

[Asynchrone RFC-Aufrufe \[Seite 40\]](#)

[Vordefinierte Ausnahmen für RFCs verwenden \[Seite 52\]](#)

## Parameter in RFC-Aufrufen

Bei einem RFC-Aufruf handhabt das System Parameter anders als bei einem lokalen Aufruf.

### TABLES-Parameter

Die eigentliche Tabelle wird übergeben, nicht jedoch der Tabellenkopf. Geben Sie keinen Tabellenparameter an, so verwendet der gerufene Funktionsbaustein eine leere Tabelle.

RFC arbeitet mit einem Delta-Verwaltungsmechanismus, um die Netzwerkbelastung bei der Parameter- und Ergebnisübergabe so gering wie möglich zu halten. Als Parameter bei Funktionsbausteinaufrufen können Sie interne ABAP-Tabellen verwenden. Bei einem lokalen Funktionsbausteinaufruf wird eine Parametertabelle "by reference" übergeben. Daher muß keine neue lokale Kopie angelegt werden. RFC unterstützt die Übergabe "by reference" nicht. Daher muß die gesamte Tabelle zwischen dem RFC-Client und dem RFC-Server hin und her übergeben werden. Sobald der RFC-Server die Tabelleneinträge erhält, legt er eine lokale Kopie der internen Tabelle an. An den RFC-Client werden dann nur noch die Delta-Informationen zurückgegeben. Allerdings werden Delta-Daten nicht bei jeder Tabellenoperation an den RFC-Client zurückgegeben, sondern gesammelt und auf einmal übergeben, sobald die Funktionskontrolle zum RFC-Client zurückkehrt.

Bei der ersten Übergabe einer Tabelle erhält die Tabelle eine Objekt-ID und wird im aufrufenden System als "virtuelle globale Tabelle" registriert. Diese Registrierung bleibt erhalten, solange weitere Aufrufe zwischen dem rufenden und dem gerufenen System möglich sind. Wenn also noch mehrere Aufrufe erfolgen, wird zwar jeweils die Delta-Information übergeben, um die lokale Kopie zu aktualisieren, aber die Tabelle selbst muß nur einmal kopiert werden (beim ersten Aufruf).

## RFC-Funktionen lokal aufrufen

## RFC-Funktionen lokal aufrufen

Es kann vorkommen, daß Sie einen entfernten Funktionsbaustein (also einen im SAP-System als entfernt registrierten Funktionsbaustein) aus demselben SAP-System heraus aufrufen wollen. Dieser Funktionsbaustein kann entweder als entfernter oder als lokaler Aufruf laufen, je nach CALL FUNCTION-Anweisung. Die Handhabung der Parameter ist jedoch unterschiedlich (wie erläutert unter [Parameter in RFC-Aufrufen \[Seite 17\]](#)).

Sie habe diese beiden Möglichkeiten:

- **CALL FUNCTION...DESTINATION = 'NONE'**

Dies ist ein entfernter Aufruf, wobei **DESTINATION = 'NONE'** bedeutet, daß der entfernte Funktionsbaustein im selben System läuft wie der Aufrufer. Als entfernter Aufruf läuft der Funktionsbaustein in einem eigenen Rollbereich und die Parameterübergabe erfolgt wie bei anderen entfernten Aufrufen (siehe [Parameter in RFC-Aufrufen \[Seite 17\]](#)).

**CALL FUNCTION 'RFC\_CUSTOMER\_GET'**

<b>DESTINATION</b> 'NONE'	
<b>EXPORTING</b>	<b>KUNNR = CUSTNO</b>
<b>TABLES</b>	<b>CUSTOMER_T = ITAB</b>
<b>EXCEPTIONS</b>	<b>NO_RECORD_FOUND = 01.</b>

- **CALL FUNCTION... [ohne DESTINATION]**

Dies ist ein lokaler Aufruf, obwohl der Funktionsbaustein als entfernt registriert ist. Der Baustein hat keinen eigenen Rollbereich und funktioniert wie ein normaler Funktionsaufruf. Die Parameterübergabe erfolgt wie bei anderen lokalen Funktionsaufrufen. Das bedeutet unter anderem, daß das System abbricht, wenn der Aufruf nicht alle EXPORTING-Parameter mit Werten versorgt.

**CALL FUNCTION 'RFC\_CUSTOMER\_GET'**

<b>EXPORTING</b>	<b>KUNNR = CUSTNO</b>
<b>TABLES</b>	<b>CUSTOMER_T = ITAB</b>
<b>EXCEPTIONS</b>	<b>NO_RECORD_FOUND = 01.</b>

Sie können einen Funktionsbaustein auch für Parallelverarbeitung im selben System aufrufen. Näheres dazu finden Sie unter [Parallelverarbeitung mit asynchronen RFCs \[Seite 47\]](#).

## RFC-Funktionen zurückrufen

Zu Beginn eines RFC-Aufrufs legen Sie Client und Server fest. Während auf dem Server ein Funktionsbaustein abläuft, kann dieser Server wiederum einen Funktionsbaustein auf dem Client aufrufen. Der entfernte Funktionsbaustein kann also seinen eigenen Aufrufer rufen (sofern der Aufrufer selbst ein Funktionsbaustein ist), oder jeden anderen mit dem Aufrufer geladenen Funktionsbaustein. Der zurückgerufene Funktionsbaustein läuft dann im selben Programmkontext wie der ursprüngliche Aufrufer.

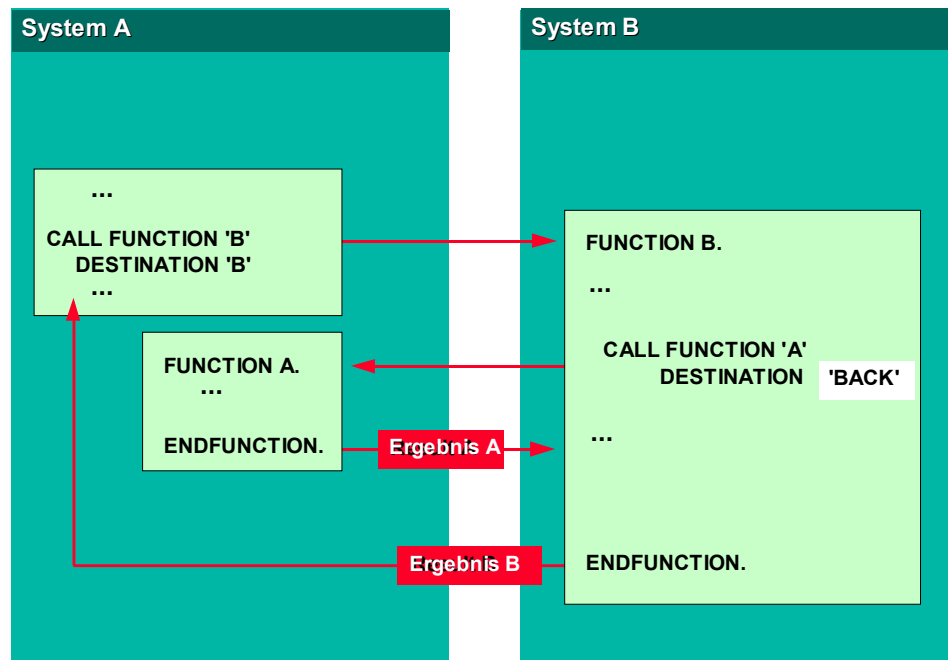
Diesen Rückrufmechanismus können Sie mit dem Destinationsnamen "BACK" aktivieren. Wenn Sie bei einem RFC-Aufruf diesen Namen im Server-System angeben, so verwendet das System genau die RFC-Verbindung, die beim ersten Aufruf des Servers aufgebaut wurde. Sobald eine RFC-Verbindung einmal aufgebaut ist, bleibt sie erhalten bis sie entweder explizit geschlossen wird oder bis das aufrufende Programm beendet ist. Während eines Rückrufs wird das System immer versuchen, bestehende RFC-Verbindungen zu verwenden, ehe es eine neue aufbaut.

Die Syntax für einen Rückruf lautet:

`CALL FUNCTION... DESTINATION 'BACK'`



In der Abbildung ruft der entfernte Funktionsbaustein B aus System B den entfernten Funktionsbaustein A im aufrufenden System A auf.



## Transaktionale RFC-Aufrufe

## Transaktionale RFC-Aufrufe

Ab Release 3.0 können Daten zwischen zwei R/3-Systemen **zuverlässig** und **sicher** über den **transaktionalen RFC (tRFC)** übergeben werden.



Dieser RFC-Typ hieß zuerst **asynchroner** RFC. Er wurde aber in **transaktionaler** RFC umbenannt, da asynchroner RFC im R/3 eine andere Bedeutung hat.

Der gerufene Funktionsbaustein wird im RFC-Server-System **genau einmal** ausgeführt. Das entfernte System muß in dem Moment, in dem das RFC-Client-Programm einen tRFC ausführt, nicht verfügbar sein. Die tRFC-Komponente speichert die gerufene RFC-Funktion zusammen mit den zugehörigen Daten in der R/3-Datenbank unter einer eindeutigen Transaktionskennung (TID).

Wird ein Aufruf gesendet während das Empfangssystem nicht verfügbar ist, bleibt der Aufruf in der lokalen Warteschlange stehen. Das rufende Dialogprogramm kann weiterlaufen ohne darauf zu warten, ob der Funktionsbaustein mit oder ohne Erfolg durchgeführt wurde. Wird das Empfangssystem nicht innerhalb eines bestimmten Zeitraums aktiviert, so wird der Aufruf als Hintergrund-Job eingeplant.

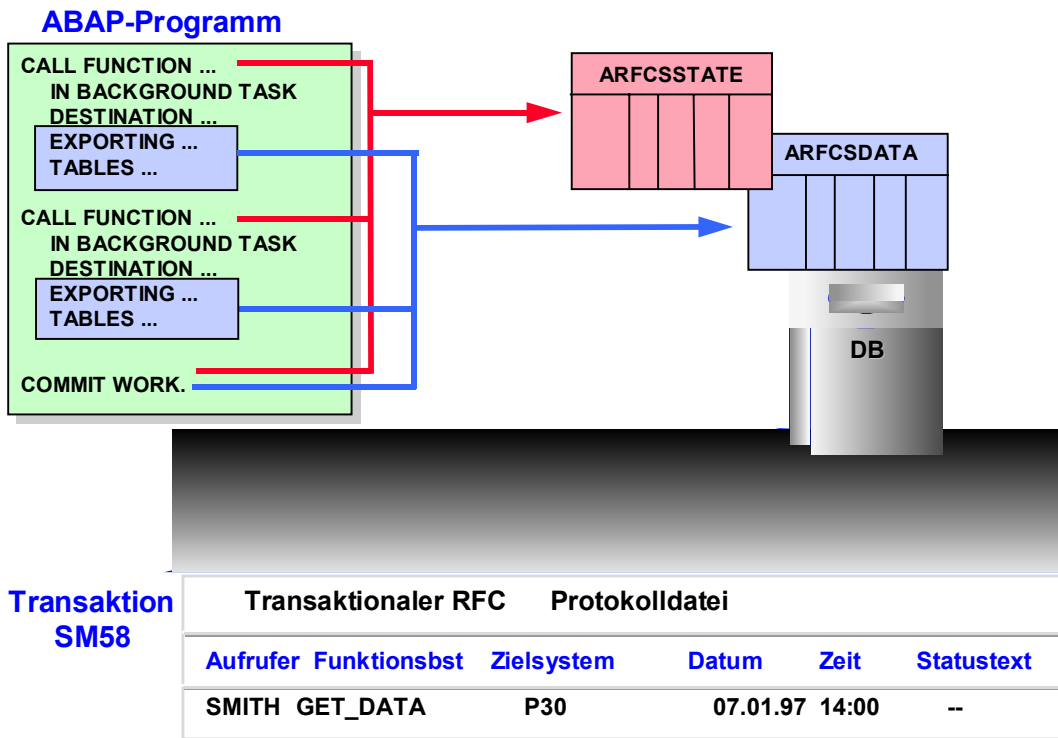
Transaktionale RFCs verwenden das Suffix **IN BACKGROUND TASK**.

Wie bei synchronen Aufrufen definiert der Parameter **DESTINATION** einen Programmkontext im entfernten System. Wenn Sie also mit derselben Destination einen Funktionsbaustein mehrmals oder mehrere Funktionsbausteine einmal aufrufen, können Sie im selben Kontext auf die globalen Daten des gerufenen Funktionsbausteins zugreifen.

Das System protokolliert die RFC-Anforderung in den Datenbanktabellen ARFCSSTATE und ARFCSDATA mit allen Parameterwerten. Sie können die Protokolldatei mit der Transaktion SM58 anzeigen. Wenn das rufende Programm ein COMMIT WORK erreicht, wird der RFC-Aufruf dem angeforderten System zur Ausführung übergeben.

Alle tRFCs mit einer einzigen Destination, die zwischen zwei COMMIT WORK vorkommen, gehören zu einer einzigen logischen Arbeitseinheit (LUW). Näheres zu LUWs, TIDs und der Statusabfrage von transaktionalen Aufrufen finden Sie unter [Transaktionale Integrität von tRFCs \[Seite 23\]](#).

Transaktionale RFC-Anforderungen werden über TCP/IP oder X400 übergeben mit den Parameterdaten als Byte-Strom.



Beis

pielsweise können Sie transaktionale RFCs für spezielle Aktualisierungsprozeduren verwenden. Einige komplexe Dialoge erfordern die Aktualisierung einer Reihe verwandter Datenbanktabellen während unterschiedlicher Phasen innerhalb einer Transaktion. Wenn die Funktionsbausteine zur Aktualisierung auf einer entfernten Maschine liegen und wenn die Änderungen an den Tabellen nicht unbedingt ausgeführt sein müssen, ehe der Dialog weiterläuft, dann können Sie transaktionale RFC-Aufrufe verwenden. Anstatt warten zu müssen bis jede einzelne Aktualisierungsprozedur abgeschlossen ist, kann der Benutzer ohne Verzögerung die Transaktion bis zum Ende durchführen. Die Verwendung transaktionaler RFCs stellt sicher, daß alle geplanten Aktualisierungen ausgeführt werden, sobald das Programm die Anweisung COMMIT WORK erreicht.



Wie bei synchronen entfernten Aufrufen müssen Sie auch bei transaktionalen RFC-Aufrufen die EXPORTING- und TABLES-Parameter nicht angeben.



Sie dürfen entfernte Funktionsbausteine dann **nicht** transaktional aufrufen, wenn Sie in der Funktionsschnittstelle EXPORT-Parameter angeben. Ein IMPORTING-Parameter in Ihrer CALL FUNCTION-Anweisung führt zu einem Compiler-Fehler.

Beachten Sie auch, daß Sie Funktionsbausteine, die Rückrufe durchführen, nicht transaktional aufrufen dürfen.

## Transaktionale RFC-Aufrufe

### Wenn das entfernte System nicht verfügbar ist

Falls das entfernte System nicht verfügbar ist, plant das SAP-System den Report RSARFCSE mit der entsprechenden Transaktionskennung (TID) als Variante für die Hintergrundverarbeitung ein. Dieser Report, der asynchrone Aufrufe zur Ausführung übergibt, wird solange immer wieder aufgerufen, bis er eine Verbindung mit dem gewünschten System hergestellt hat.

Bei einer Einplanung im Hintergrund läuft RSARFCSE automatisch in vorbestimmten Zeitintervallen (Standard: alle 15 Minuten, bis zu 30 Versuche). Dieses Intervall und die Anzahl der Versuche können Sie beliebig einstellen. Verwenden Sie dazu die Erweiterungsprogramme SABP0000 und SABP0003 (oder siehe SAP-Erweiterungskonzept und CALL CUSTOMER-FUNCTION).

In der Transaktion SM59 (Menüpfad *Werkzeuge* → *Administration, Verwaltung* → *Netzwerk* → *RFC Destinationen*) wählen Sie *Destination* → *TRFC Optionen*, um die einzelnen Destinationen zu konfigurieren. Dabei können Sie die Anzahl der Versuche zum Verbindungsaufbau und den Zeitraum zwischen den Versuchswiederholungen festlegen.

Kann das System während der vorgegebenen Zeit nicht erreicht werden, ruft das System RSARFCSE nicht länger auf, sondern schreibt den Status CPICERR in die Tabelle ARFCSDATA. Der entsprechende Eintrag in der Tabelle ARFCSSTATE wird, ebenfalls nach Verstreichen eines bestimmten Zeitraums (Standard: acht Tage), gelöscht (auch diesen Zeitraum können Sie selbst einstellen). Es ist jedoch weiterhin möglich, solche Einträge mit Transaktion SM59 von Hand zu starten.

## Transaktionale Integrität von tRFCs

Sie können Funktionsbausteine im Hintergrund in einem anderen SAP-System oder einem externen Programm ausführen. So aufgerufene Funktionsbausteine werden nicht sofort ausgeführt, sondern warten, bis ein COMMIT WORK ausgelöst wird.

Transaktionale RFCs verdanken ihren Namen der Tatsache, daß der zugeordnete Aufrufmechanismus die transaktionale Integrität für alle innerhalb des **IN BACKGROUND TASK**-Suffixes durchgeführten Aufrufe garantiert. So wie bei Datenbankaktualisierungen werden auch für Aufrufe, die zum Ablauf in Hintergrund-Aufgaben eingeplant sind, logische Arbeitseinheiten (LUWs) erzeugt. Alle tRFCs mit einer einzigen Destination, die zwischen zwei COMMIT WORK-Anweisungen vorkommen, gehören zu einer einzigen LUW. Innerhalb einer LUW werden alle Aufrufe

- in der Reihenfolge ausgeführt, in der sie aufgerufen werden
- im selben Programmkontext im Zielsystem aufgeführt
- in einer einzigen Transaktion ausgeführt: sie werden entweder komplett auf die Datenbank geschrieben (COMMIT) oder komplett zurückgesetzt (ROLLBACK).

LUWs werden anhand von Transaktionskennungen identifiziert, die weltweit eindeutig sind. Zur Festlegung der Transaktionskennung rufen Sie in einem ABAP-Programm den Funktionsbaustein **ID\_OF\_BACKGROUNDTASK** auf. (Sie müssen diesen Funktionsbaustein nach dem ersten asynchronen CALL und vor dem zugehörigen COMMIT WORK aufrufen.)

Da der RFC wie eine Transaktion behandelt wird, werden Datenbankoperationen entweder alle ausgeführt oder, falls ein Funktionsbaustein abbricht, alle zurückgesetzt. Wurde eine LUW erfolgreich abgeschlossen, können Sie sie nicht noch einmal ausführen. In manchen Fällen kann es erforderlich sein, eine LUW programmgesteuert zurückzusetzen, z.B. wenn eine Tabelle gesperrt ist. Dazu rufen Sie den Funktionsbaustein **RESTART\_OF\_BACKGROUNDTASK** auf, der ein ROLLBACK durchführt und sicherstellt, daß die LUW später noch einmal ausgeführt wird.

In der Regel wird eine LUW sofort nach einem COMMIT WORK im Zielsystem ausgeführt. Sie können jedoch auch eine bestimmte Zeit festlegen, zu der sie ausgeführt werden soll. Dazu rufen Sie innerhalb der betreffenden LUW den Funktionsbaustein **START\_OF\_BACKGROUNDTASK** auf, und zwar nach dem ersten CALL... **IN BACKGROUND TASK** und vor **COMMIT WORK**.



Nähere Informationen zum transaktionalen RFC finden Sie in der Online-Hilfe des ABAP-Editors.

## Status von transaktionalen Aufrufen prüfen

Alle transaktionalen RFC-Aufrufe werden in den Tabellen ARFCSSTATE und ARFCSDATA gespeichert. Dabei wird jede LUW durch eine eindeutige ID identifiziert. Bei einem COMMIT WORK werden die zu dieser ID gehörenden Aufrufe im Zielsystem ausgeführt. Der Systemfunktionsbaustein **ARFC\_DEST\_SHIP** transportiert die Daten ins Zielsystem und der Funktionsbaustein **ARFC\_EXECUTE** führt die gespeicherten Aufrufe aus. Tritt während eines Aufrufs ein Fehler oder eine Ausnahme ein, so setzt das System alle Datenbankoperationen, die von den vorhergehenden Aufrufen gestartet wurden, zurück und schreibt eine entsprechende Fehlernachricht in die Datei ARFCSSTATE.

Den Status einer Transaktionskennung können Sie auf zwei Arten prüfen:

## Transaktionale Integrität von tRFCs

- **Aus einem ABAP-Programm**

Der Funktionsbaustein `ID_OF_BACKGROUNDTASK` gibt die ID der LUW zurück. Rufen Sie diesen Funktionsbaustein nach dem ersten `CALL . . . IN BACKGROUND TASK` und vor `COMMIT WORK` auf.

```
CALL FUNCTION 'ID_OF_BACKGROUNDTASK' IMPORTING      TASK-ID =
TID.
```

Haben Sie die ID der LUW herausgefunden, können Sie mit dem Funktionsbaustein `STATUS_OF_BACKGROUNDTASK` den Status des transaktionalen RFC abfragen.

```
CALL FUNCTION 'STATUS_OF_BACKGROUNDTASK'
```

<code>EXPORTING TID</code>	<code>= TASK-ID</code>
<code>IMPORTING ERRORTAB</code>	<code>= ERTAB</code>
<code>EXCEPTIONS COMMUNICTATION</code>	<code>= 01</code>
<code>(Verbindung nicht verfügbar: Versuch wird wiederholt)</code>	
<code>RECORDED</code>	<code>= 02</code>
<code>(ARFC ist eingeplant)</code>	
<code>ROLLBACK</code>	<code>= 03</code>
<code>(Rollback im Zielsystem ausgelöst)</code>	

- **Online**

Rufen Sie die Transaktion SM58 auf (*Werkzeuge → Administration, Verwaltung → Monitor → Transaktionaler RFC*). Dieses Werkzeug zeigt eine Liste aller transaktionalen RFCs an, die nicht erfolgreich abgeschlossen werden konnten oder die als Hintergrund-Jobs eingeplant werden mußten. Die Liste enthält die ID der LUW und die entsprechende Fehlermeldung, die im Zielsystem auftrat. Um den Text der Nachricht zu sehen, klicken Sie zweimal darauf.

Mit der Transaktion SM58 können Sie außerdem Ihren transaktionalen RFC in unterschiedlichen Phasen steuern. Wenn der Aufruf während des Sendevorgangs abbricht, können Sie mit der Funktion *LUW-Rollback* die LUW von Hand zurücksetzen, ehe Sie einen neuen Sendeversuch starten. Wenn das Zielsystem nicht verfügbar ist, können Sie mit der Funktion *Eingeplante Jobs anzeigen* den Hintergrund-Job anzeigen, den das System für Sie erzeugt hat. Wenn ein temporärer Fehler aufgetreten ist (z.B. ein Syntaxfehler), können Sie mit der Funktion *Funktionsbaustein ausführen* den Aufruf erneut starten.

Wenn eine LUW im Zielsystem erfolgreich gelaufen ist, löst das System den Funktionsbaustein `ARFC_DEST_CONFIRM` aus und bestätigt die erfolgreiche Ausführung im Zielsystem.

Anschließend werden die Einträge in den Tabellen ARFCSTATE und ARFCSDATA gelöscht.

## RFC-API

Auch in Funktionsbausteinen, die in 'C' implementiert wurden, können Sie Programme asynchron ausführen (Verbindungstyp TCP/IP in Transaktion SM59, siehe [Destinationstypen \[Seite 63\]](#)). Die Funktionsbausteine werden wie üblich in Verbindung mit der RFC-Bibliothek implementiert. Die RFC-Bibliothek enthält die Funktionsbausteine `ARFC_DEST_SHIP` und `ARFC_DEST_CONFIRM`, die die entsprechenden Funktionen aufrufen.

Nähere Informationen hierzu finden Sie unter [RFC-API \[Extern\]](#).

## qRFC mit Ausgangsqueue

### qRFC mit Ausgangsqueue

Um eine von der Anwendung diktierte Reihenfolge der LUW's zu garantieren, wird eine Serialisierung des tRFC über Queues vorgenommen. Sie wird daher als **queued RFC (qRFC)** bezeichnet. Mit der Serialisierung wurde im R/3 eine Ausgangsqueue für tRFC geschaffen, weshalb die Bezeichnung **qRFC mit Ausgangsqueue** üblich ist.

Darüber hinaus gibt es auch Anwendungen, die den Zeitpunkt der LUW-Ausführung im Zielsystem selbst bestimmen wollen. Meistens handelt es sich hierbei um Anwendungen auf einem externen (Non-SAP-) System, wo eine Art Ausgangsqueue nicht vorhanden ist. **qRFC mit Eingangsqueue** inklusiv einer entsprechenden Erweiterung der RFC-Library befindet sich z.Z. in Entwicklung und wird wahrscheinlich zum kommenden Release verfügbar sein.

### Motivation

#### Was leistet bereits der tRFC ?

- Der tRFC gewährleistet, daß ein gerufener Funktionsbaustein **genau einmal** im Zielsystem ausgeführt wird.
- Alle tRFC-Aufrufe, die mit der Anweisung Commit Work abgeschlossen werden, gehören zu einer LUW (**Logical Unit of Work**). Jeder LUW wird automatisch eine Transaktions-ID zugeordnet.
- Innerhalb einer LUW werden alle zugehörigen Funktionsbausteine in derselben Reihenfolge im Zielsystem ausgeführt wie sie im Sendesystem aufgerufen werden.
- tRFC-Aufrufe mit dem Zusatz **AS SEPARATE UNIT** bilden im Zielsystem jeweils eine separate LUW ab. Eine solche **Sub-LUW** enthält genau einen tRFC-Aufruf und wird unabhängig von der eigentlichen (übergeordneten) LUW behandelt. Diese Sub-LUW erhält eine eigene Transaktions-ID.
- Werden innerhalb einer LUW mehrere Destinationen verwendet, so bilden alle zu einer Destination zugehörigen tRFC-Aufrufe ebenfalls eine eigene Sub-LUW. Aufgrund dieser internen Bündelung wird jedoch **keine** eigene Transaktions-ID für die zugehörige Sub-LUW vergeben.

#### Nachteile des tRFC

Mit dem tRFC werden alle LUW's unabhängig voneinander behandelt. Dies kann aufgrund der Vielzahl von aktivierten tRFC-Prozessen zu einer deutlichen Verschlechterung der Performance sowohl im Sendesystem als auch im Zielsystem führen.

Ferner kann die in der Anwendung definierte **Reihenfolge der LUW's** nicht eingehalten werden. Es besteht also keine Gewähr, daß die vom Anwendungsprogramm vorgegebene Reihenfolge der Transaktionen auch eingehalten wird. Es gibt lediglich die Garantie, daß alle LUW's früher oder später übertragen werden.

### Verwendung und Verfügbarkeit

Zu den typischen Anwendungen, für die die Serialisierung des tRFC implementiert wurde, gehören Technologien in einer verteilten Umgebung wie:

- **ALE** (Application Link Enabling)
- **APO** (Advanced Planner and Optimizer)

- **SFA** (Sales Force Automation)

Der qRFC mit Ausgangsqueue ist ab **Release 4.5B** freigegeben.



Speziell für APO wurde diese Serialisierung auch in einer Sonderfreigabe für APO-Systeme, Rel. 3.1H bzw. 4.5A zur Verfügung gestellt.

### Weitere Themen

[qRFC mit Ausgangsqueue im Überblick \[Seite 28\]](#)

[Serialisierung programmieren \[Seite 30\]](#)

["Mischbetrieb" einsetzen \[Seite 32\]](#)

[Transaktionsreihenfolge und Queue-Belegung \[Seite 34\]](#)

[Hilfsmittel \[Seite 37\]](#)

## qRFC mit Ausgangsqueue im Überblick

## qRFC mit Ausgangsqueue im Überblick

Im folgenden gehen wir von den Voraussetzungen aus, daß die Verarbeitungsreihenfolge der Transaktionen im Anwendungsprogramm definiert ist und ferner, daß an einer Transaktion mehrere Queues beteiligt sein können.

### Eigenschaften

Die Besonderheiten des qRFC mit Ausgangsqueue lassen sich wie folgt zusammenfassen:

- Der queued RFC mit Ausgangsqueue erzwingt eine Serialisierung vonseiten des Sendesystems. Das Zielsystem hat keine Kenntnis von der Serialisierung im Sendesystem. Somit ist eine Kommunikation mit jedem beliebigen R/3-Zielsystem ab Rel. 3.0 möglich.
- qRFC mit Ausgangsqueue stellt eine **Erweiterung von tRFC** dar. Dabei wird eine LUW (Transaktion) erst dann übertragen, wenn sie keine Vorgänger (bezogen auf die in verschiedenen Anwendungsprogrammen definierte Reihenfolge) in den beteiligten Queues hat. Ferner wird nach der Ausführung einer qRFC-Transaktion versucht, alle wartenden qRFC-Transaktionen der Reihenfolge nach automatisch zu starten.
- Für die Queue-Verwaltung werden ein Queue-Name und ein Queue-Zähler pro qRFC-Transaktion benötigt. Jeder zu serialisierende tRFC-Aufruf wird einem von der Anwendung selbst zu bestimmenden Queue-Namen zugeordnet. Der Queue-Name wird mit dem Aufruf des Funktionsbausteins **TRFC\_SET\_QUEUE\_NAME** übergeben. Dies erfolgt direkt vor jedem zu serialisierenden tRFC-Aufruf. Siehe auch [Serialisierung programmieren \[Seite 30\]](#). Ein Queue-Name ist ein frei wählbarer Text bis zu einer max. Länge von 24 B, der das Zeichen \* nicht enthalten darf.



Es empfiehlt sich, Queue-Namen so zu wählen, daß verschiedene Anwendungen nicht unabsichtlich den gleichen Namen verwenden (Beispiele: SFA\_... oder APO\_...).

- Eine Queue-Konfigurierung wird nicht vorgenommen, da es sich um eine **logische Queue** handelt. Es sind keine separaten Tabellen für die einzelnen Queues vorgesehen. Zu einem R/3-System gibt also es nur eine Tabelle für die Ausgangsqueues. Alle Angaben zu Queues von allen beteiligten LUW's werden in der Queue-Tabelle **TRFCQOUT** abgelegt.
- Falls die zu serialisierenden tRFC-Aufrufe innerhalb einer LUW auf **mehrere Destinationen** verteilt sind, werden sie nach Destinationen gebündelt und unabhängig voneinander behandelt. Eine Gruppierung in Sub-LUW's innerhalb einer LUW, wie beim bisherigen tRFC, ist die Folge. Es gibt auch hier keine Garantie über die Reihenfolge der Ausführung dieser Sub-LUW's, sondern nur die Gewährleistung, daß alle früher oder später übertragen werden. Anders als beim bisherigen tRFC erhält aber jede solche Sub-LUW eine eigene Transaktions-ID zur besseren und einfacheren Queue-Behandlung.
- Die Reihenfolge der zu serialisierenden tRFC-Aufrufe mit oder ohne die Option **AS SEPARATE UNIT** in einer LUW wird gewährleistet, auch dann wenn sie verschiedenen Queues zugeordnet sind.
- Mit dem Absetzen des **COMMIT WORK** wird ein Zähler ermittelt, der als Kriterium für die Reihenfolge der LUW-Ausführung in einer Queue herangezogen wird. Bei 2 aus Anwendungssicht abhängigen LUW's kann ein COMMIT WORK der 2. LUW erst dann erfolgen kann, nachdem der COMMIT WORK der ersten LUW durchgeführt worden ist.

**qRFC mit Ausgangsqueue im Überblick**

- Ein "**Mischbetrieb**" wird unterstützt: Innerhalb einer LUW können sowohl normale als auch zu serialisierende tRFC-Aufrufe, mit oder ohne den Zusatz AS SEPARATE UNIT vorkommen. Die Reihenfolge der tRFC-Aufrufe wird selbstverständlich gewährleistet.
  - Die "normalen" tRFC-Aufrufe innerhalb einer LUW mit "Mischbetrieb" und die zu serialisierenden tRFC-Aufrufe gleicher Destination bilden bezüglich ihrer Zuordnung eine Sub-LUW. Falls innerhalb einer LUW für eine bestimmte Destination kein qRFC-Aufruf definiert wurde, bilden die "normalen" tRFC-Aufrufe selbst eine eigene Sub-LUW.



Anders als beim bisherigen tRFC, wo alle tRFC-Aufrufe unterschiedlicher Destinationen nur einer gemeinsamen Transaktions-ID, aber intern verschiedenen Sub-LUW's zugeordnet werden, erhält jede Bündelung von tRFC-/qRFC-Aufrufen gleicher Destination zur besseren Queue-Bearbeitung eine eigene Transaktions-ID. Um jedoch eine Kompatibilität in der bisherigen tRFC-Behandlung zu erreichen, muß eine Anwendung, die "Mischbetrieb" in einer LUW einsetzt und der erste Aufruf ein tRFC-Aufruf ist, zuerst den Funktionsbaustein **TRFC\_QUEUE\_INITIALIZE** absetzen. Siehe auch "[Mischbetrieb](#)" einsetzen [Seite 32].

- Die "normalen separaten" tRFC-Aufrufe (mit Zusatz AS SEPARATE UNIT) in einer LUW mit "Mischbetrieb" werden wie bisher behandelt (eigene Transaktions-ID).
- Die zu serialisierenden "separaten" tRFC-Aufrufe werden zwar queue-abhängig, jedoch unabhängig von der eigentlichen LUW bearbeitet.
- Ein Nebeneffekt und zugleich ein Vorteil des qRFC mit Ausgangsqueue gegenüber dem bisherigen tRFC ist, daß sowohl das Sende- als auch das Zielsystem durch die Serialisierung nicht sehr stark belastet werden (aufgrund geringerer Parallelisierung). Eine ungeschickte Auswahl der Queue-Namen (zu viele Namen) kann diesen Vorteil aber wieder zunichte machen. Es handelt sich hierbei aber um einen klassischen Konflikt: Bessere Performance durch mehr Serialisierung gegenüber schnellerer Verarbeitung durch mehr Parallelisierung. Jede Anwendung muß daher genau abwägen, wie Queue-Namen sinnvoll zu definieren und den einzelnen Funktionsaufrufen zuzuordnen sind.

**Welche Probleme sind zu erwarten ?**

Folgende allgemeine Serialierungsprobleme sind zu beachten:

Wenn z.B. wegen der Netzwerk-/Kommunikationsprobleme die erste LUW in einer Queue nicht ausgeführt werden kann, bleibt nicht nur diese Queue hängen, sondern auch alle anderen LUW's, die in einer Abhängigkeit mit dieser Queue stehen. Der dadurch entstandene Stau könnte zum Datenbank-Problem führen.

Sobald jedoch das Problem behoben wurde, können alle Transaktionen und somit alle Queues nach und nach automatisch bearbeitet werden.

## Serialisierung programmieren

## Serialisierung programmieren

Die folgende Vorgehensweise beschreibt die Programmierung des queued RFC mit Ausgangsqueue in seiner Grundform (d.h. ohne den Zusatz AS SEPARATE UNIT und ohne "Mischbetrieb")

### Voraussetzungen

- Das R/3-Sendesystem hat den Releasestand 4.5B.
- Alle benötigten Destinationen wurden in der Tabelle RFCDEST des Sendesystem gepflegt.
- Um eine gute Performance der Anwendung zu erzielen, ist die Anzahl und die Zuordnung der Queue-Namen sorgfältig zu bestimmen.

### Vorgehensweise

Um den qRFC mit Ausgangsqueue zu implementieren, sind die folgenden Schritte auszuführen:

1. Ordnen Sie die Queue-Namen für den (die) nachfolgenden Funktionsaufruf(e) zu.
2. Setzen Sie für die Übergabe des Queue-Namens den Funktionsbaustein TRFC\_SET\_QUEUE\_NAME unmittelbar vor jedem Funktionsaufruf ab.
3. Rufen Sie den Funktionsbaustein mit CALL FUNCTION ... IN BACKGROUND TASK auf.
4. Wiederholen Sie die Schritte 2-3 (und eventuell 1) für die weiteren Funktionsaufrufe.
5. Schließen Sie die LUW mit COMMIT WORK ab.
6. Führen Sie die Schritte 1-5 für die weiteren LUW's der Anwendung aus.



```
REPORT ... .
...
DATA Q_NAME LIKE TRFCQOUT-QNAME.

Q_NAME = 'BASIS_TEST_Q1'.
CALL FUNCTION ' TRFC_SET_QUEUE_NAME '
      EXPORTING
          QNAME = Q_NAME.
CALL FUNCTION 'RFC_FUNCTION_1'
      IN BACKGROUND TASK
      DESTINATION 'DEST'
      EXPORTING ...
      TABLES ... .
...
Q_NAME = 'BASIS_TEST_QM'.
```

```
CALL FUNCTION 'TRFC_SET_QUEUE_NAME'
  EXPORTING
    QNAME = Q_NAME.
CALL FUNCTION 'RFC_FUNCTION_N'
  IN BACKGROUND TASK
  DESTINATION 'DEST'
  EXPORTING ...
  TABLES ... .
COMMIT WORK.
* NEXT LUW
...
```

## Ergebnis

Die Queue-Angaben werden in der Tabelle TRFCQOUT abgelegt.

Mit jedem COMMIT WORK wird ein Zähler ermittelt, der für die Reihenfolge der Transaktionsverarbeitung herangezogen wird.



Zum Einstieg können Sie auch das einfache Demo-Programm **RSTRFCT0** ausführen.

**"Mischbetrieb" einsetzen****"Mischbetrieb" einsetzen**

Ein Mischbetrieb zeichnet sich dadurch aus, daß qRFC-Aufrufe mit Ausgangsqueue mit normalen tRFC-Aufrufen gemeinsam in einer LUW behandelt werden. Zur Charakterisierung des "Mischbetriebs" siehe auch [qRFC mit Ausgangsqueue im Überblick \[Seite 28\]](#).

**Voraussetzungen**

Analog zu [Serialisierung programmieren \[Seite 30\]](#).

**Vorgehensweise**

Bei der Implementierung des "Mischbetriebs" in einer LUW, ist folgende Besonderheit zu beachten:

Ist in einer LUW mit "Mischbetrieb" der erste Aufruf ein normaler tRFC -Aufruf, so muß zuerst der Funktionsbaustein **TRFC\_QUEUE\_INITIALIZE** abgesetzt werden. Damit wird sichergestellt, daß die aktuelle LUW über qRFC mit Ausgangsqueue bearbeitet wird.

Ist dagegen der erste Aufruf ein qRFC-Aufruf, so wird diese Initialisierung bereits durch den Funktionsbaustein TRFC\_SET\_QUEUE\_NAME durchgeführt.



Die folgende Programmsequenz zeigt exeplarisches, wie eine LUW mit "Mischbetrieb" implementiert wird:

```
REPORT ... .
...
DATA Q_NAME LIKE TRFCQOUT-QNAME.

CALL FUNCTION 'TRFC_QUEUE_INITIALIZE'.
CALL FUNCTION 'RFC_FUNCTION_1'           " tRFC call -> NO QUEUE
      IN BACKGROUND TASK
      DESTINATION 'DEST'
      EXPORTING ...
      TABLES ... .
...
Q_NAME = 'BASIS_TEST_Q1'.
CALL FUNCTION 'TRFC_SET_QUEUE_NAME'
      EXPORTING
          QNAME = Q_NAME.
CALL FUNCTION 'RFC_FUNCTION_2'           " qRFC call -> Q1
      IN BACKGROUND TASK
      DESTINATION 'DEST'
```

**"Mischbetrieb" einsetzen**

```

EXPORTING ...
TABLES ... .
...

CALL FUNCTION 'RFC_FUNCTION_N-1'           "tRFC call -> NO QUEUE
      IN BACKGROUND TASK
      DESTINATION 'DEST'
      EXPORTING ...
      TABLES ... .
Q_NAME = 'BASIS_TEST_QM'.
CALL FUNCTION 'TRFC_SET_QUEUE_NAME'
      EXPORTING
          QNAME = Q_NAME.
CALL FUNCTION 'RFC_FUNCTION_N'           " qRFC call -> QM
      IN BACKGROUND TASK
      DESTINATION 'DEST'
      EXPORTING ...
      TABLES ... .
...
COMMIT WORK.

```

## Ergebnis

Aufgrund der Transaktionseigenschaft bilden die normalen tRFC-Aufrufe und die qRFC-Aufrufe gleicher Destination eine Einheit. Falls jedoch zu einer Destination keine qRFC-Aufrufe innerhalb der aktuellen LUW zu bearbeiten sind, werden die normalen tRFC-Aufrufe zu einer eigenen Sub-LUW gebündelt.

Die normalen tRFC-Aufrufe mit dem Zusatz AS SEPARATE UNIT bilden, wie üblich, eine separate LUW im Zielsystem ab.

qRFC-Aufrufe mit dem Zusatz AS SEPARATE UNIT werden zwar queue-abhängig, jedoch auch separat von der eigentlichen LUW bearbeitet.



Zum Einstieg in die Programmierung des "Mischbetriebs" steht Ihnen das Demo-Programm **RSTRFCT1** zur Verfügung.

## Transaktionsreihenfolge und Queue-Belegung

## Transaktionsreihenfolge und Queue-Belegung

An einem Beispielszenario soll verdeutlicht werden, welche Abhängigkeit der Sendereihenfolge der LUW's von der Queue-Zuordnung der einzelnen Aufrufe besteht. An einem Beispiel wird demonstriert, welche Queue-Belegung schließlich vorliegt, nachdem eine bestimmte Transaktion versendet wurde. Testen Sie auch das Demo-Programm **RSTRFCT3**.

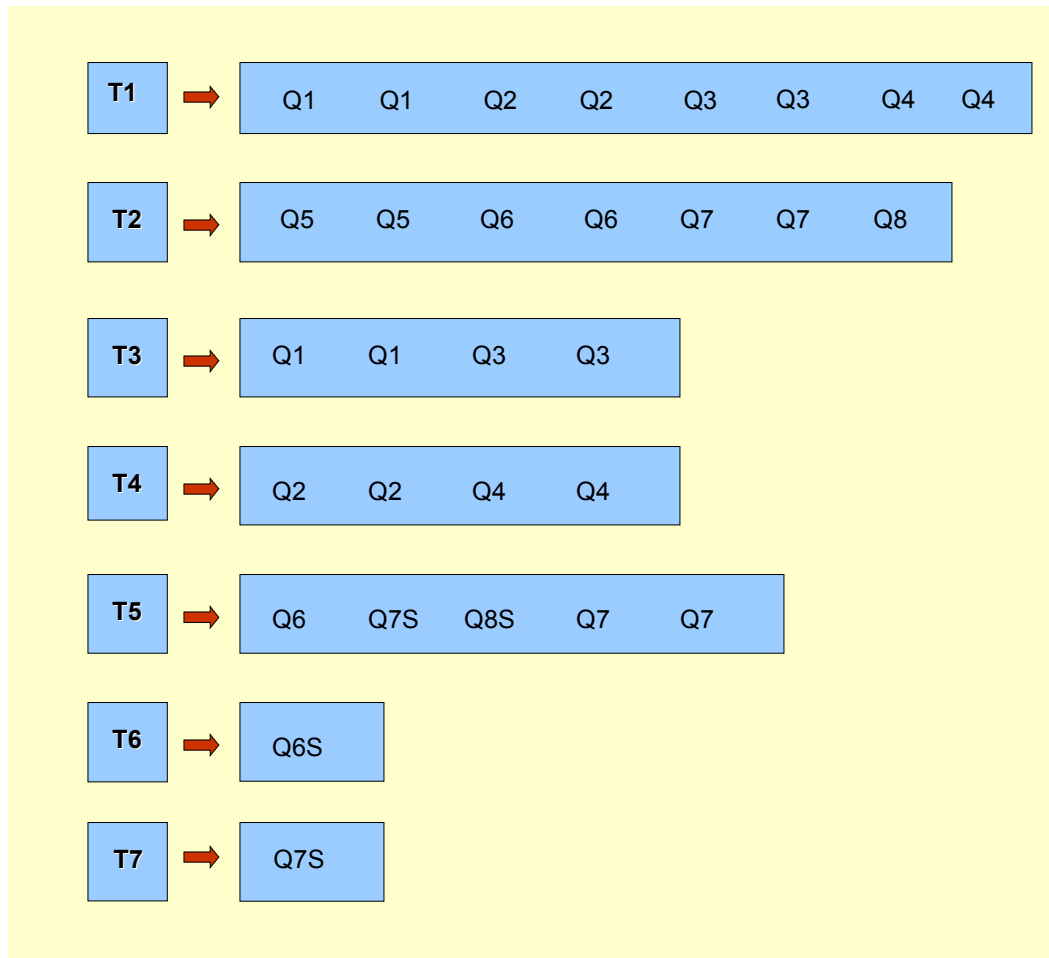
### Beispielszenario

#### Queue-Beteiligung

Die folgende Skizze gibt zum einen die Transaktionsreihenfolge wieder, so wie sie von einer Anwendung diktiert wird und zum anderen, die zugehörige Queue-Beteiligung der einzelnen Aufrufe wieder :

So enthält zum Beispiel die Transaktion **T3** 2 qRFC-Aufrufe mit der Zuordnung zur Queue **Q1** und 2 Aufrufe mit der Zuordnung zur Queue **Q3**.

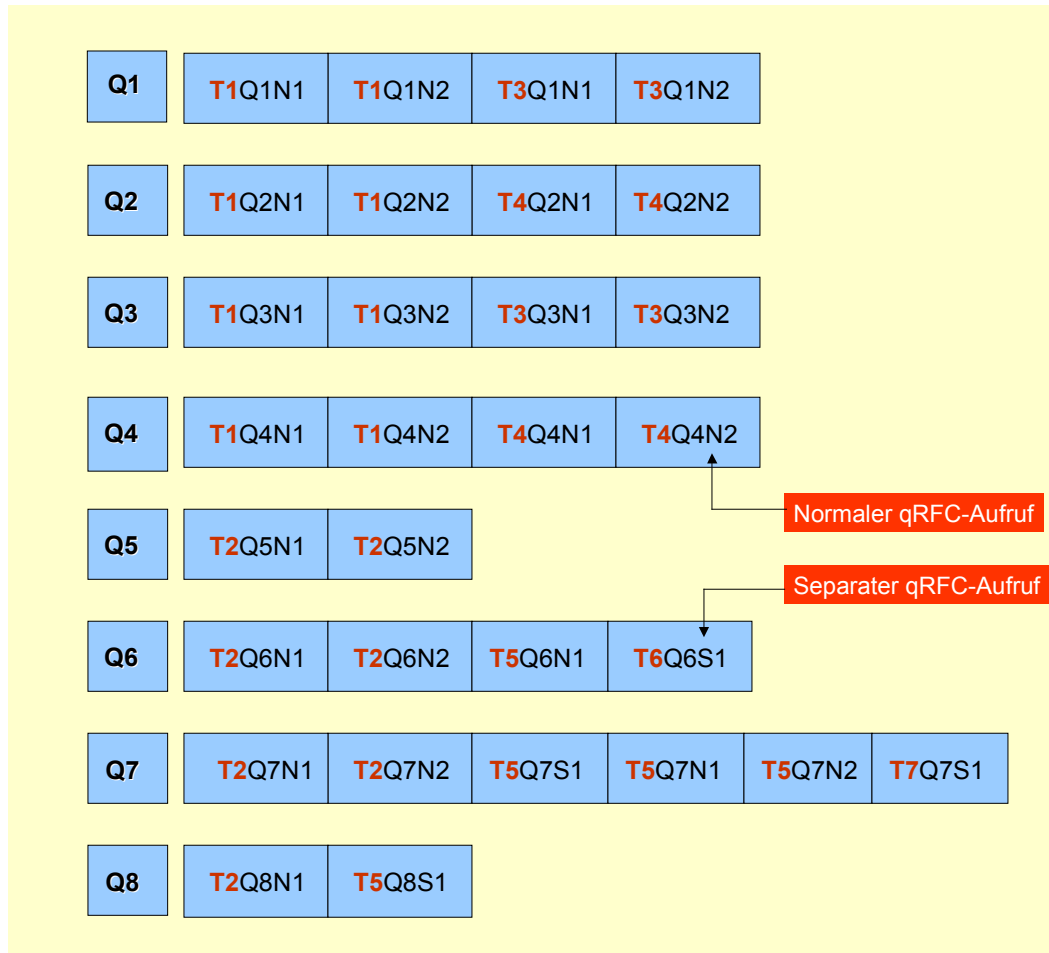
Demgegenüber enthält die Transaktion **T6** nur den qRFC-Aufruf mit dem Zusatz AS SEPARATE UNIT mit der Zuordnung zur Queue **Q6**.



## Belegung der Queues

### Vor dem Versenden

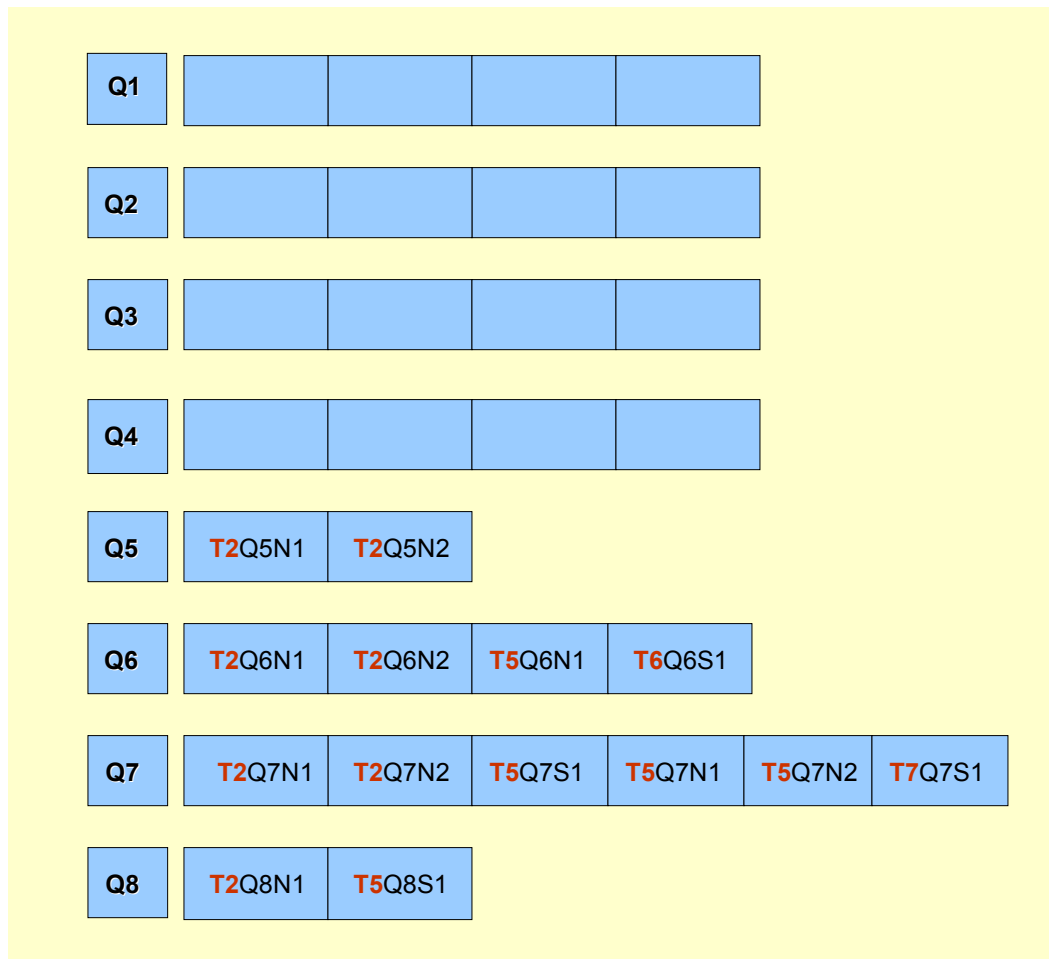
Aus der in unserem Beispiel vorliegenden Queue-Zuordnung resultiert die folgende Initial-Belegung für die einzelnen Queues:



### Nach dem Versenden von T1

Nach einem erfolgreichem Versenden der Transaktion T1 werden ebenfalls die Transaktionen T3 und T4 zum Versenden vorgesehen. Daraus resultiert die folgende Queue-Belegung:

## Transaktionsreihenfolge und Queue-Belegung



Zunächst werden die Aufrufe von T1 in allen beteiligten Queues ( Q1, Q2, Q3 und Q4) versendet. Alle weiteren unabhängigen Transaktionen, die mit T1 gemeinsame Queue-Teilnehmer haben, werden im Anschluß an T1 zum Versenden vorgesehen. Dies betrifft die Transaktionen T3 und T4.

Die Transaktionen T2, T5, T6 und T7 verbleiben im Wartezustand, bis T3 und T4 erfolgreich versendet wurden.

Erst hiernach wird versucht, die wartenden Transaktionen automatisch der Reihe nach zu starten.

## Hilfsmittel

### Tools

#### Queue-Namen übergeben

##### TRFC\_SET\_QUEUE\_NAME

Dieser Funktionsbaustein dient zur Übergabe eines Queue-Namens für jeweils nur einen Funktionsaufruf unmittelbar vor einem "call function ... in background task" (tRFC-Aufruf) und signalisiert, daß die aktuelle LUW über qRFC mit Ausgangsqueue bearbeitet werden muß. Der übergebene Queue-Name gilt nur für den nachfolgenden tRFC-Aufruf.

#### Queue-Einträge lesen

##### TRFC\_GET\_QUEUE\_INFO und TRFC\_GET\_QUEUE\_INFO\_DETAILS

Der aktuelle Inhalt einer oder aller Queues aus der Tabelle TRFCQUOT kann mit diesen Funktionsbausteinen in mehr oder weniger detaillierter Form gelesen werden.

##### RSTRFCQR

Dieses ABAP-Programm setzt die beiden obigen Funktionsbausteine ein, um den aktuellen Inhalt der Ausgangsqueue-Tabelle TRFCQUOT anzuzeigen.

##### RSTRFCQD

Mit diesem ABAP-Programm können alle Einträge einzelner oder aller Transaktionen gelöscht werden. Dieses Programm sollte entweder für Testzwecke und nur im äußersten Notfall zu verwendet werden. Das Löschen der Einträge einer Transaktion führt dazu, daß diese Transaktion nicht mehr serialisiert wird und evtl. manuell nachgestartet werden muß.

#### Initialisieren im "Mischbetrieb"

##### TRFC\_QUEUE\_INITIALIZE

Der Aufruf dieses parameterlosen Funktionsbausteins am Anfang einer qRFC-LUW mit "Mischbetrieb" (sowohl tRFC als auch qRFC) signalisiert, daß die aktuelle LUW über qRFC mit Ausgangsqueue bearbeitet werden muß. Dies ist jedoch nur notwendig, falls der erste Aufruf ein normaler tRFC-Aufruf ist. Ist dagegen der erste Aufruf ein qRFC-Aufruf, wird diese Initialisierung bereits durch den Funktionsbaustein TRFC\_SET\_QUEUE\_NAME durchgeführt. Siehe auch Demo-Programm **RSTRFCT1**.

#### SM58

Die bisherige Transaktion **sm58** als tRFC-Monitoring zum Anzeigen und Bearbeiten der tRFC-Transaktionen kann ebenfalls für qRFC verwendet werden. Das Löschen eines tRFC-Eintrags in dieser Transaktion führt automatisch zum Löschen entsprechender Einträge in der Ausgangsqueue-Tabelle, sofern sie dort vorhanden sind. Das Starten einer LUW führt nicht zum

## Hilfsmittel

sofortigen Übertragen dieser LUW, da zuerst geprüft werden muß, ob diese LUW serialisiert werden soll und ob sie wegen der vorhandenen Queue-Vorgänger warten muß.

## Stoppen, Erneutes Starten und Statusabfrage von Queues

Die Bearbeitung einer oder mehrerer Queues (mit generischer Angabe) kann angehalten und wieder fortgesetzt werden. Ferner kann der Status einer Queue auch abgefragt werden. Folgende Funktionsbausteine stehen hierfür zur Verfügung:

### TRFC\_QOUT\_STOP

Mit Angabe der Queue-Namen (einzeln oder generische Angabe wie BASIS\_TEST\_\* möglich) und einer Destination kann die Bearbeitung einer oder mehrerer Queues angehalten werden. Optional können diese Queues nach Möglichkeit auch sofort angehalten werden (Parameter FORCE = 'X') oder es werden zuerst alle bis zum STOP-Zeitpunkt vorhandenen Aufträge in dieser Queue abgearbeitet und danach die Bearbeitung angehalten (FORCE = ' '). Das Anhalten einer leeren Queue ist ebenfalls möglich und führt dazu, daß alle Transaktionen, die dieser Queue zugeordnet sind, nicht sofort versendet werden. Siehe auch Demo-Programm **RSTRFCQ1**.

### TRFC\_QOUT\_STATE

Mit der Angabe eines Queue-Namens und einer Destination kann der Status einer Queue abgefragt werden. Siehe auch Demo-Programm **RSTRFCQ2**. Folgende Status sind möglich:

- **RUNNING**
- **CPICERR**
- **SYSFAIL**
- **STOPPING**
- **STOP**
- **WAITSTOP**

Der Status **SYSFAIL** geht auf einen schwerwiegenden Fehler zurück und wird durch eine Exception im R/3-Kernel des Zielsystems selbst oder im aufgerufenen Funktionsbaustein hervorgerufen. In diesem Status bleibt die Queue praktisch stehen. In diesem Fall findet auch keine automatische Wiederholung statt. Über **SM58** kann diese LUW nochmals versendet oder auch gelöscht werden. Im zweiten Fall werden automatisch die entsprechenden Einträge in der Queue-Tabelle TRFCQOUT gelöscht.

Im Status **CPICERR** ist eine automatische Wiederholung abhängig von der Konfiguration der Destination in sm59 (defaultmäßig mit "Ja" vorbelegt).

### TRFC\_QOUT\_RESTART

Mit Angabe des Queue-Namens (einzeln oder generische Angabe wie BASIS\_TEST\_\* möglich) und einer Destination kann die Bearbeitung einer oder mehrerer Queues fortgesetzt werden unabhängig davon, ob diese Queues vorher durch Queue-Stop (Parameter FORCE) oder durch CPICERR oder SYSSFAIL angehalten wurden. Siehe auch Demo-Programm **RSTRFCQ3**.

### TRFC\_QOUT\_RESTART\_COND

Mit Angaben des Queue-Namens (einzeln oder generische Angabe wie BASIS\_TEST\_\* möglich) und einer Destination kann die Fortsetzung der Bearbeitung einer oder mehrerer Queues vom Ende der Bearbeitung einer oder von mehreren anderen Queues abhängig gemacht werden. Ein Aufruf dieser Funktion führt automatisch dazu, daß die fortzusetzenden Queues spätestens ab diesem Zeitpunkt angehalten werden, falls vorher ein entsprechender STOP-Aufruf (abhängig von der Belegung des Parameters FORCE) noch nicht erfolgt war. Siehe auch Demo-Programm **RSTRFCQ4**.

## Beispielprogramme

- **RSTRFCT0**, **RSTRFCT1**, **RSTRFCT2** und **RSTRFCT3** sind Beispiel- bzw. Testprogramme für qRFC mit Ausgangsqueue.
- Mit den Programmen **RSTRFCQ1**, **RSTRFCQ2**, **RSTRFCQ3** und **RSTRFCQ4** kann man die Bearbeitung einer oder mehrerer Queues anhalten, mit/ohne Bedingung fortsetzen oder den Status eine Queue abfragen.

## Asynchrone RFC-Aufrufe

## Asynchrone RFC-Aufrufe

Asynchrone Aufrufe (aRFCs) ähneln transaktionalen RFCs insofern als der Benutzer nicht die erfolgreiche Ausführung abwarten muß, ehe er mit seinem Aufrufdialog fortfahren kann. Es gibt jedoch drei wesentliche Merkmale, in denen sich asynchrone und transaktionale RFCs unterscheiden:

- Wenn der Aufrufer einen asynchronen RFC startet, muß der gerufene Server verfügbar sein, um die Anforderung entgegenzunehmen.  
Die Parameter von asynchronen RFCs werden nicht in die Datenbank geschrieben, sondern direkt an den Server übergeben.
- Während eines asynchronen RFCs kann der Benutzer einen Dialog mit dem entfernten System führen.
- Das rufende Programm kann Ergebnisse des asynchronen RFCs empfangen.

Einen asynchronen Funktionsaufruf verwenden Sie immer dann, wenn Sie eine Kommunikation mit einem entfernten System aufbauen müssen, aber mit der Verarbeitung nicht auf die Ergebnisse des Funktionsbausteins warten wollen. Auch asynchrone RFCs können an dasselbe System geschickt werden. In diesem Fall öffnet das System einen neuen Modus (oder Fenster). Sie können dann zwischen dem rufenden Dialog und dem gerufenen Modus hin- und herspringen.

Um einen Funktionsaufruf asynchron zu starten, verwenden Sie folgende Syntax:

```
CALL FUNCTION EntfernteFunktion STARTING NEW TASK Aufgabename
```

```
Destination ...  
EXPORTING...  
TABLES ...  
EXCEPTIONS...
```



Es gibt folgende Aufrufparameter:

- TABLES  
übergibt Referenzen auf interne Tabellen. Alle Tabellenparameter des Funktionsbausteins müssen Werte enthalten.
- EXPORTING  
übergibt die Werte von Feldern und Strukturen aus dem rufenden Programm an den Funktionsbaustein. Im Funktionsbaustein sind die entsprechenden Formalparameter als Importparameter definiert.
- EXCEPTIONS  
siehe [Vordefinierte Ausnahmen für RFCs verwenden \[Seite 52\]](#)

**RECEIVE RESULTS FROM FUNCTION** `func` wird **innerhalb** einer FORM-Routine verwendet, um die Ergebnisse eines asynchronen RFCs zu erhalten. Es gibt folgende Ergebnisparameter:

## Asynchrone RFC-Aufrufe

- IMPORTING
- TABLES
- EXCEPTIONS

Der Zusatz **KEEPING TASK** sorgt dafür, daß eine asynchrone Verbindung nach der Ergebnisübergabe bestehen bleibt. Der entsprechende entfernte Kontext (Rollbereich) wird solange weiterverwendet, bis der Aufrufer die Verbindung beendet.



Rufen Sie eine Transaktion asynchron auf und zeigen Sie sie in einem amodalen Fenster an:

```
DATA: MSG_TEXT(80) TYPE C.      "Nachrichtentext
...
* Asynchroner Aufruf von Transaktion SM59 ->
* Neuen Modus öffnen
CALL FUNCTION 'ABAP4_CALL_TRANSACTION' STARTING NEW TASK
'TEST'
  DESTINATION 'NONE'
  EXPORTING
    TCODE = 'SM59'
  EXCEPTIONS
    COMMUNICATION_FAILURE = 1 MESSAGE MSG_TEXT
    SYSTEM_FAILURE        = 2 MESSAGE MSG_TEXT
  IF SY-SUBRC NE 0.
    WRITE: MSG_TEXT.
  ELSE.
    WRITE: 'O.K.'
  ENDIF.
```

Detaillierte Informationen finden Sie in folgenden Themen:

[Aufrufanforderungen für asynchrone RFCs \[Seite 42\]](#)

[Ergebnisse von einem asynchronen RFC empfangen \[Seite 43\]](#)

[Entfernte Kontexte aufrechterhalten \[Seite 46\]](#)

[Parallelverarbeitung mit asynchronem RFC \[Seite 47\]](#)

---

**Aufrufanforderungen für asynchrone RFCs**

## Aufrufanforderungen für asynchrone RFCs

Wenn Sie einen RFC mit dem optionalen Suffix **STARTING NEW TASK** aufrufen, startet das System den Funktionsbaustein in einem neuen Modus. Sobald der Funktionsbaustein im Zielsystem gestartet wurde, kann der Benutzer die Verarbeitung fortsetzen, ohne auf das Ende des entfernten Aufrufs warten zu müssen.

Der entfernt aufgerufene Funktionsbaustein kann z.B. mit **CALL SCREEN** ein neues Bild aufrufen und dem Benutzer so den direkten Dialog mit dem entfernten System ermöglichen:

### Client-System

```
CALL FUNCTION 'TRAVEL_CREATE_FLIGHT'  
  STARTING NEW TASK 'FLIGHT'  
  DESTINATION 'S11' .
```

### Server-System

```
FUNCTION TRAVEL_CREATE_FLIGHT.  
  CALL SCREEN 100.  
ENDFUNCTION.
```

Wenn Sie keine Destination angeben, startet der asynchrone RFC-Mechanismus einen neuen Modus im aufrufenden System.



Beim Aufruf von aRFCs dürfen Sie **kein** IMPORTING verwenden.

## Ergebnisse von einem asynchronen RFC erhalten

Um Ergebnisse eines asynchron gerufenen Funktionsbausteins zu erhalten, verwenden Sie folgende Syntax:

```
CALL FUNCTION EntfernteFunktion
  STARTING NEW TASK Aufgaben
  PERFORMING RETURN_FORM ON END OF TASK.
```

Sobald der gerufene Funktionsbaustein beendet ist, führt der nächste Dialogschritt im rufenden Programm (z.B. `AT USER-COMMAND`) das System in die FORM-Routine, die nach Ergebnissen sucht. Diese FORM-Routine besteht aus einer besonderen Syntax und muß mit einem USING-Parameter aufgerufen werden, der sich auf den Namen der Aufgabe bezieht:

### Client-System

```
CALL FUNCTION 'TRAVEL_CREATE_FLIGHT'
  STARTING NEW TASK 'FLIGHT'
  DESTINATION 'S11'
  PERFORMING RETURN_FLIGHT ON END OF TASK.
...
FORM RETURN_FLIGHT USING TASKNAME.
  RECEIVE RESULTS FROM FUNCTION 'TRAVEL_CREATE_FLIGHT'
    IMPORTING FLIGHTID = SFLIGHT-ID
    EXCEPTIONS SYSTEM_FAILURE MESSAGE SYSTEM_MSG.
  SET USER-COMMAND 'OKCD'.
ENDFORM.
```



- Wenn ein Funktionsbaustein kein Ergebnis zurückliefert, können Sie den Zusatz `PERFORMING RETURN_FORM ON END OF TASK` weglassen.
- Wenn ein asynchroner Aufruf mehrere aufeinander folgende Funktionsbausteine mit derselben Destination aufruft, müssen Sie jedem einen anderen Aufgabennamen zuordnen.
- Ein rufendes Programm, das einen asynchronen RFC mit `PERFORMING` startet, kann nicht den Rollbereich ändern oder in einen internen Modus wechseln, weil die Antwort auf den asynchronen Aufruf eines Funktionsbausteins dann nicht an das entsprechende Programm weitergeleitet werden kann. Einen Wechsel des Rollbereichs führen Sie mit `SUBMIT` oder `CALL TRANSACTION` durch.
- Wenn das rufende Programm, das den asynchronen Aufruf abgesetzt hat, beendet wird, obwohl es noch auf eine Antwort wartet, dann kann diese Antwort auf den asynchronen Aufruf nicht abgeliefert werden.

### Ergebnisse von einem asynchronen RFC erhalten

- Wenn Sie die Anweisung **WAIT** bei PERFORMING form ON END OF TASK verwenden, wartet das System auf die Antwort eines zuvor gestarteten asynchronen Aufrufs. WAIT muß hier im selben Programmkontext stehen.
- Das System setzt die Verarbeitung des Programms nach WAIT fort, wenn entweder die Bedingung eines logischen Ausdrucks von der Unterroutine erfüllt wurde, die die fragliche Aufgabe ausführt, oder wenn eine festgesetzte Zeitspanne überschritten wurde. Nähere Informationen zu WAIT finden Sie in der Online-Hilfe des ABAP-Editors.

Das Schlüsselwort **RECEIVE** gibt es nur beim Funktionsbaustein aufruf **CALL FUNCTION func STARTING NEW TASK taskname**. Wenn der Funktionsbaustein keine Ergebnisse liefert, können Sie diesen Teil weglassen.



Das Schlüsselwort **RECEIVE** gibt es seit dem R/3-Release 3.0. Daher müssen beide Partnersysteme, Client und Server, mindestens das R/3-Release 3.0 verwenden.

Nähere Informationen zu **RECEIVE** finden Sie in der Online-Hilfe des ABAP-Editors.

Die Anweisung **SET USER-COMMAND 'OKCD'** wirkt so als ob der Benutzer diese Funktion im Befehlsfeld eingegeben und bestätigt hätte; d.h. die aktuellen Positionen der Liste und des Cursors werden berücksichtigt.



Rückrufe werden nicht unterstützt.



Die Anweisung **SET USER-COMMAND 'OKCD'** ersetzt die Anweisung **REFRESH SCREEN**. **REFRESH SCREEN** wird nicht länger gepflegt und sollte daher **nicht** verwendet werden.



```
DATA: INFO LIKE RFCSI,
* Ergebnis des RFC_SYSTEM_INFO-Funktionsbausteins
  MSG(80) VALUE SPACE.
* Behandlung der Ausnahmen
CALL FUNCTION 'RFC_SYSTEM_INFO'
  STARTING NEW TASK 'INFO'
  PERFORMING RETURN_INFO ON END OF TASK
  EXCEPTIONS
    COMMUNICATION_FAILURE = 1 MESSAGE MSG
    COMMUNICATION_FAILURE = 2.MESSAGE MSG.
IF SY-SUBRC = 0.
  WRITE: 'Warte auf Antwort'.
```

## Ergebnisse von einem asynchronen RFC erhalten

```
ELSE.
    WRITE MSG
ENDIF.
...
AT USER-COMMAND.
* Rückkehr aus FORM-Routine RETURN_INFO über SET USER-COMMAND
    IF SY-UCOMM = 'OKCD'.
        IF MSG = SPACE.
            WRITE: 'Destination =', INFO-RFCDEST.
        ELSE.
            WRITE MSG.
        ENDIF.
    ENDIF.
...
FORM RETURN_INFO USING TASKNAME.
    RECEIVE RESULTS FROM FUNCTION 'RFC_SYSTEM_INFO'
        IMPORTING RFCSI_EXPORT = INFO
    EXCEPTIONS
        COMMUNICATION_FAILURE = 1 MESSAGE MSG
        SYSTEM_FAILURE        = 2 MESSAGE MSG.
    SET USER-COMMAND 'OKCD'.           "Setzt OK-Code
ENDFORM.
```

## Entfernte Kontexte aufrechterhalten

### Entfernte Kontexte aufrechterhalten

In der FORM-Routine, die mit RECEIVE RESULTS FROM FUNCTION nach Ergebnissen eines asynchron aufgerufenen Funktionsbausteins sucht, können Sie mit dem Zusatz **KEEPING TASK** verhindern, daß die Verbindung nach der Ergebnisübergabe abgebaut wird.

```
FORM RETURN_INFO USING TASKNAME.  
    RECEIVE RESULTS FROM FUNCTION 'RFC_SYSTEM_INFO'  
    KEEPING TASK  
  
    ...  
ENDFORM.
```

Der entsprechende entfernte Kontext (Rollbereich) bleibt dann erhalten, bis der Aufrufer die Verbindung abbricht. Wenn Sie denselben Aufgabennamen angeben, können Sie den entfernten Kontext und Rollbereich erneut verwenden.

Wenn der entfernte Funktionsbaustein Dialogaufgaben wie List- oder Dynproverarbeitung ausführt, werden die Bildschirmbilder solange angezeigt, bis das rufende Programm beendet ist. Wenn Sie den entfernten Aufruf im Debugging-Modus ausführen, bleibt dieser Modus sichtbar bis der Dialog des Aufrufers beendet ist.



Verwenden Sie den Zusatz KEEPING TASK nur dann, wenn Sie den aktuellen entfernten Kontext für den nächsten asynchronen Aufruf wiederbenutzen wollen.

Das Aufrechterhalten eines entfernten Kontextes erhöht die Systembelastung und verringert die Performance aufgrund der zusätzlich erforderlichen Rollbereichsverwaltung im System.

## Parallelverarbeitung mit asynchronem RFC

Um die Systembelastung gleichmäßig zu verteilen, können Sie Destinationszusätze verwenden, die Funktionsbausteine in parallelen Aufgaben auf einem beliebigen Anwendungs-Server oder einer vordefinierten Gruppe von Anwendungs-Servern eines R/3-Systems ausführen.



Parallelverarbeitung wird mit einer besonderen aRFC-Variante implementiert. Daher ist es wichtig, daß Sie für Ihre eigenen Parallelverarbeitungsanwendungen nur diese richtige Variante verwenden: die Anweisung CALL FUNCTION STARTING NEW TASK DESTINATION IN GROUP. Wenn Sie eine andere RFC-Variante verwenden, umgehen Sie damit die in der korrekten Anweisung eingebauten Absicherungen und können Ihr System zum Absturz bringen.

Detailliertere Informationen finden Sie unter:

- Voraussetzungen für die Parallelverarbeitung
- Funktionsbausteine und ABAP-Schlüsselwörter für die Parallelverarbeitung
- Ressourcenverwaltung in der Parallelverarbeitung

### Voraussetzungen für die Parallelverarbeitung

Ehe Sie die Parallelverarbeitung implementieren, vergewissern Sie sich, daß Ihre Anwendung und Ihr R/3-System folgende Anforderungen erfüllen:

- Logisch voneinander unabhängige Arbeitseinheiten:  
Die Datenverarbeitungsaufgabe, die parallel ausgeführt werden soll, muß von anderen Instanzen der Aufgabe logisch unabhängig sein. D.h. die Aufgabe kann ausgeführt werden ohne auf andere Sätze derselben Datenmenge Bezug zu nehmen, die auch parallel verarbeitet wird, und die Aufgabe ist unabhängig von den Ergebnissen anderer Aufgaben der Parallelverarbeitung. Parallelverarbeitung ist z.B. nicht geeignet für Daten, die nacheinander verarbeitet werden müssen, oder bei denen die Verarbeitung einer Information von der Verarbeitung einer anderen Information abhängt.  
Bei der Parallelverarbeitung gibt es keine Garantie, daß Daten in einer bestimmten Reihenfolge verarbeitet werden, oder daß zu einem bestimmten Verarbeitungszeitpunkt ein bestimmtes Ergebnis verfügbar ist.
- ABAP-Anforderungen:
  - Der Funktionsbaustein, den Sie aufrufen, muß als extern aufrufbar markiert sein. Markieren Sie dazu in der Funktionsbausteindefinition (Transaktion SE37) das Feld *Remote Function Call unterstützt*.
  - Der gerufene Funktionsbaustein darf keinen Funktionsaufruf mit der Destination "BACK" enthalten.
  - Das rufende Programm sollte nach einem asynchronen RFC-Aufruf nicht in einen neuen internen Modus wechseln. Verwenden Sie also nach der Anweisung CALL FUNCTION STARTING NEW TASK weder SUBMIT noch CALL TRANSACTION im rufenden Programm.

## Parallelverarbeitung mit asynchronem RFC

- Mit der Anweisung `CALL FUNCTION STARTING NEW TASK DESTINATION IN GROUP` können Sie keine externen Programme starten.
- R/3-Systemressourcen:  
Um Aufgaben aus parallelen Jobs verarbeiten zu können, muß ein Server in Ihrem R/3-System mindestens drei Dialogarbeitsprozesse haben. Zudem muß er der Arbeitsbelastung durch das Parallelverarbeitungssystem gewachsen sein: Dispatcher-Warteschlange zu weniger als 10% gefüllt, mindestens einen Dialogarbeitsprozeß frei zur Verarbeitung der Aufgaben aus dem parallelen Job.

## Funktionsbausteine und ABAP-Schlüsselwörter für die Parallelverarbeitung

Um die Parallelverarbeitung in Ihren Anwendungen zu implementieren, verwenden Sie folgende Funktionsbausteine und ABAP-Schlüsselwörter:

- **SPBT\_INITIALIZE:** Optionaler Funktionsbaustein.  
Stellt die Verfügbarkeit von Ressourcen für die Parallelverarbeitung fest.  
Sie können
  - prüfen, ob die Parallelverarbeitungsgruppe, die Sie angegeben haben, korrekt ist.
  - herausfinden, wieviele Arbeitsprozesse verfügbar sind, damit Sie die Größe der zu verarbeitenden Datenpakete entsprechend anpassen können.
- **CALL FUNCTION <Funktion> STARTING NEW TASK <Aufgabenname> DESTINATION IN GROUP:**  
Mit dieser ABAP-Anweisung weisen Sie das R/3-System an, den Funktionsbausteinaufruf parallel zu verarbeiten. Stellen Sie diese Anweisung in eine Schleife, in der Sie die zu verarbeitenden Daten in Arbeitspakete aufteilen. Übergeben Sie die zu verarbeitenden Daten in einer internen Tabelle (EXPORT-, TABLE-Argumente). Die Anweisung implementiert die Parallelverarbeitung, indem sie asynchrone RFC-Aufrufe an die Server absetzt, die in der für die Verarbeitung angegebenen RFC-Server-Gruppe verfügbar sind.  
Beachten Sie, daß die RFC-Aufrufe mit `CALL FUNCTION` in Arbeitsprozessen vom Typ `DIALOG` verarbeitet werden. Damit gilt die `DIALOG`-Begrenzung zur Verarbeitung eines Dialogschrittes (Standard: 300 Sekunden, System-Profil-Parameter *rdisp/max\_wprun\_time*) auch für diese RFC-Aufrufe. Denken Sie an diese Begrenzung, wenn Sie Ihre Datenpakete für die Parallelverarbeitung einteilen.
- **SPBT\_GET\_PP\_DESTINATION:** Optionaler Funktionsbaustein.  
Rufen Sie diesen Funktionsbaustein direkt nach der Anweisung `CALL FUNCTION` auf, um den Namen des Servers zu erhalten, auf dem die Parallelverarbeitung ausgeführt wird.
- **SPBT\_DO\_NOT\_USE\_SERVER:** Optionaler Funktionsbaustein.  
Rufen Sie diesen Funktionsbaustein auf, um einen bestimmten Server von der weiteren Verarbeitung paralleler Aufgaben auszuschließen. Verwenden Sie diesen Funktionsbaustein in Verbindung mit `SPBT_GET_PP_DESTINATION`, wenn Sie feststellen, daß ein bestimmter Server für die Parallelverarbeitung nicht verfügbar ist (z.B. nach einer `COMMUNICATION FAILURE`-Ausnahme, wenn ein Server nicht mehr verfügbar ist).

Parallelverarbeitung mit asynchronem RFC

- **WAIT:** ABAP-Schlüsselwort  
 WAIT UNTIL <logischer Ausdruck>  
 Dieses Schlüsselwort müssen Sie verwenden, wenn Sie auf die Rückmeldung aller mit CALL FUNCTION erzeugten asynchronen parallelen Aufgaben warten wollen. Dies ist z.B. für eine geordnete Hintergrundverarbeitung erforderlich. Sie dürfen das Schlüsselwort jedoch nur dann verwenden, wenn die CALL FUNCTION-Anweisung den Zusatz PERFORMING ON RETURN enthält.
- **RECEIVE:** ABAP-Schlüsselwort  
 RECEIVE RESULTS FROM FUNCTION <Funktion>  
 Dieses Schlüsselwort müssen Sie verwenden, wenn Sie die Ergebnisse der Verarbeitung eines asynchronen RFCs erhalten wollen. RECEIVE liefert sowohl IMPORT- und TABLE-Parameter als auch Nachrichten und Rückgabewerte.

**Ressourcenverwaltung bei der Parallelverarbeitung**

Um asynchrone Funktionsbausteinaufrufe im R/3-System parallel auszuführen, verwenden Sie folgende Destinationszusätze:

In einer vordefinierten Gruppe von Anwendungs-Servern:

```
CALL FUNCTION EntfernteFunktion STARTING NEW TASK Aufgabename
Destination IN GROUP Gruppenname
```

In allen gegenwärtig verfügbaren Anwendungs-Servern:

```
CALL FUNCTION EntfernteFunktion STARTING NEW TASK Aufgabe
Destination IN GROUP DEFAULT
```

Der Zusatz stellt zunächst die gegenwärtig verfügbaren Ressourcen fest (Anzahl der Arbeitsprozesse in allen Servern oder in einer Gruppe von Anwendungs-Servern, vergleichbar mit Login-Servern). Die auf den einzelnen Anwendungs-Servern verfügbaren Ressourcen zur Verarbeitung asynchroner Aufrufe hängen ab von der aktuellen Systembelastung.

Der Anwendungsentwickler ist für die Verfügbarkeit von RFC-Gruppen im Produktionssystem (d.h. im Kundensystem) verantwortlich. Detaillierte Informationen zur Pflege von RFC-Gruppen finden Sie unter [Gruppendestinationen pflegen zur Verteilung der Systembelastung \[Seite 66\]](#).

Nachdem die verfügbaren Ressourcen festgestellt wurden, wird der asynchrone Aufruf auf einem verfügbaren Anwendungs-Server ausgeführt. Wenn zu einem bestimmten Zeitpunkt keine Ressourcen verfügbar sind, führt das System die Ausnahmeroutine RESOURCE\_FAILURE aus (siehe Zusatz Exceptions). Bei einem asynchronen Funktionsbausteinaufruf **muß** diese Ausnahme vom Anwendungsprogramm verarbeitet werden.

Die Ermittlung verfügbarer Ressourcen in einer RFC-Gruppe erfolgt so:

Zunächst ermittelt das System die Länge der Dispatcher-Warteschlange für den entsprechenden Anwendungs-Server. Überschreitet sie 10% der Gesamtlänge, verfügt der Server über keine Ressourcen für die Parallelverarbeitung. Ist sie kürzer, ermittelt das System die aktuelle Anzahl freier Dialogprozesse minus 2 (als Reserve-Instanz für andere Zwecke wie z.B. den Zugriff auf System- oder Verwaltungsprogramme). Daher **muß** ein Anwendungs-Server mindestens drei Dialogprozesse haben, um für die RFC-Parallelverarbeitung in Frage zu kommen.

---

**Parallelverarbeitung mit asynchronem RFC**

- Gegenwärtig wird für die parallele Verarbeitung asynchroner Aufrufe **nur eine** RFC-Gruppe pro Programmumfeld unterstützt. Sie dürfen daher nicht beide Zusätze (**DESTINATION IN GROUP <Gruppenname>** und **DESTINATION IN GROUP DEFAULT**) in einem Programm verwenden.
- Die Ausnahmeroutine **RESOURCE\_FAILURE** wird nur in Verbindung mit asynchronen RFCs mit den Zusätzen **DESTINATION IN GROUP <Gruppenname>** und **DESTINATION IN GROUP DEFAULT** ausgelöst.

Aus Performance-Gründen empfehlen wir, für die parallele Verarbeitung asynchroner Aufrufe nur eine RFC-Gruppe mit genügend Ressourcen zu verwenden.

## Berechtigungen für RFC prüfen

Wenn der System-Profile-Parameter *auth/rfc\_authority\_check* gesetzt ist (Wert 1), prüft das System beim CALL FUNCTION-Schlüsselwort automatisch, ob der Benutzer die erforderliche RFC-Berechtigung hat.

Das RFC-Berechtigungsobject ist *S\_RFC Authorization check at RFC access*. Diese Berechtigung prüft das Zugriffsrecht auf Funktionsbausteine auf der Ebene von Funktionsbausteingruppen, d.h. ob ein Benutzer das Recht hat, Funktionsbausteine auszuführen, die einer bestimmten Gruppe angehören.

Ob ein Benutzer eine bestimmte RFC-Berechtigung hat, können Sie mit dem Funktionsbaustein *AUTHORITY\_CHECK\_RFC* überprüfen. Dieser Funktionsbaustein liefert den Rückgabewert 0, wenn der Benutzer auf die angegebene Gruppe zugreifen darf. Der Funktionsbaustein prüft jedoch nicht, ob eine Berechtigungsprüfung auch tatsächlich stattfindet.

## Vordefinierte Ausnahmen für RFCs verwenden

### Vordefinierte Ausnahmen für RFCs verwenden

Während alle im gerufenen Funktionsbaustein auftretenden Ausnahmen über den Zusatz `PERFORMING form ON END OF TASK` abgedeckt sind, definiert die RFC-Schnittstelle selbst zwei weitere Ausnahmetypen:

- **SYSTEM\_FAILURE**  
Diese Ausnahme meldet alle Ausfälle und Systemprobleme auf der entfernten Maschine.
- **COMMUNICATION\_FAILURE**  
Diese Ausnahme wird angestoßen, wenn ein Verbindungs- oder Kommunikationsausfall stattfindet. Sie meldet keine Systemprobleme (z.B. Abbrüche) auf der entfernten Maschine.

### Fehlernachrichten anfordern

In den Funktionsbausteinen, die Sie aufrufen, sollten Sie für alle Arten von Fehlernachrichten Ausnahmen definieren und nicht nur das Schlüsselwort `MESSAGE` verwenden.

**CALL FUNCTION** EntfernteFunktion

```
DESTINATION 'hw1071_53'  
EXPORTING...  
IMPORTING...  
TABLES...  
EXCEPTIONS  
    SYSTEM_FAILURE = 1 MESSAGE msg  
    COMMUNICATION_FAILURE = 2 MESSAGE msg
```

Das System stellt die Systemnachricht in die Nachrichtenvariable (`msg`). Sie können nun die Nachricht anzeigen oder in einer Datei protokollieren. Versuchen Sie nicht, Nachrichtentexte in Ihrem Programm zu interpretieren.



`MESSAGE` können Sie **nur** zusammen mit den beiden oben beschriebenen Systemausnahmen verwenden.

## RFC-Funktionsbausteine in ABAP schreiben

Dieser Abschnitt befaßt sich mit dem Schreiben von Funktionsbausteinen, die entfernt aufgerufen werden können. Informationen zum Schreiben allgemeiner Funktionsbausteine finden Sie in der Systemdokumentation "Funktionsbausteine schreiben".

Der Abschnitt enthält folgende Themen:

[Schritte bei der Implementierung von RFC-Funktionsbausteinen \[Seite 54\]](#)

[Programmierungshilfen \[Seite 55\]](#)

[RFC-Funktionsbausteine im Debugger \[Seite 57\]](#)

---

**Schritte bei der Implementierung von RFC-Funktionsbausteinen**

## Schritte bei der Implementierung von RFC-Funktionsbausteinen

Um einen entfernten Funktionsbaustein in ABAP zu implementieren, führen Sie folgende Schritte durch:

1. Registrieren Sie den Funktionsbaustein im RFC-Server-System als entfernt aufrufbar.  
Markieren Sie im Verwaltungsbild des Funktionsbausteins (Transaktion SE37) das Feld *Remote Function Call unterstützt*. Sobald Sie einen Funktionsbaustein als entfernt registrieren, generiert das System ein entsprechendes RFC-Stub-Programm.
2. Schreiben Sie das Coding für den Funktionsbaustein.  
Richtlinien für das Erstellen von Funktionsbausteinen im R/3-Repository finden Sie in der Systemdokumentation "Funktionsbausteine schreiben".
3. Definieren Sie die Destination des RFC-Servers in dem RFC-Client-System, das den entfernten Funktionsbaustein aufruft.

Entweder Sie oder Ihr Systemadministrator kann mit Transaktion SM59 (*Werkzeuge* → *Administration, Verwaltung* → *Netzwerk* → *RFC Destinationen*) die Tabelle RFCDES pflegen. Genauere Informationen hierzu finden Sie unter [Destinationen pflegen \[Seite 58\]](#).

## Programmierungshilfen

Die im folgenden beschriebenen Punkte sollten Sie beim Schreiben entfernter Funktionsbausteine beachten.

### Parameter deklarieren

Wenn bei normalen (nicht entfernten) Funktionsbausteinen eine Parameterdefinition sich nicht auf ein ABAP Dictionary-Feld bezieht, nimmt der Parameter zur Laufzeit den Datentyp des Aktualparameters an. Einem entfernten Funktionsbaustein steht diese Information jedoch nicht zur Verfügung. Daher müssen Sie alle Parameterfelder eines entfernten Funktionsbausteins als Bezugsfelder definieren, d.h. mit LIKE auf ein ABAP Dictionary-Feld Bezug nehmen. (Dies gilt für IMPORTING-, EXPORTING- und TABLES-Parameter.)

Bei Zeichenstrukturen oder -feldern müssen die Parameter des Aufrufers nicht so lang sein wie sie das gerufene Programm erwartet. Wenn Eingangsparameter kürzer sind, füllt RFC sie einfach mit Leerzeichen auf. Das bedeutet, daß die ABAP Dictionary-Definition von Zeichenparametern bei Aufrufer und aufgerufenem Programm nicht absolut identisch sein müssen. Allerdings dürfen die Parameter des Aufrufers **nicht länger** sein als auf der gerufenen Seite erwartet.

### Funktionsbausteine für transaktionale Ausführung



Wenn Sie Funktionsbausteine schreiben, die transaktional aufgerufen werden sollen, müssen Sie zwei Einschränkungen beachten:

- Transaktionale Aufrufe können keine Parameterwerte zurückliefern. Geben Sie daher in der Schnittstelle eines solchen Funktionsbausteins keine EXPORT-Parameter an.
- Transaktional laufende Funktionsbausteine können keine Rückrufe ausführen: Zu dem Zeitpunkt, zu dem ein Rückruf das Ursprungssystem erreichen würde, kann der Kontext des Aufrufers schon längst beendet sein.

### Ausnahmen

Ausnahmen stoßen Sie in einem entfernten Funktionsbaustein genauso an wie in einem lokal gerufenen Funktionsbaustein.

Da das System die Ausnahmen COMMUNICATION\_FAILURE und SYSTEM\_FAILURE intern auslöst, müssen Sie sich in Ihrem Programm darum nicht kümmern. Das Schlüsselwort MESSAGE ist nur in Verbindung mit diesen beiden Systemausnahmen sinnvoll; Sie brauchen es daher in Ihrem Funktionsbaustein nicht zu berücksichtigen.

### Aufrufe anderer RFC-Funktionen

Wie jeder normale Funktionsbaustein kann ein entfernter Funktionsbaustein andere entfernte Funktionsbausteine aufrufen.

Insbesondere kann ein entfernter Funktionsbaustein die Rückruffunktionalität verwenden, um Funktionsbausteine aufzurufen, die im selben System laufen wie der ursprüngliche Aufrufer.

### Programmierungshilfen

Detaillierte Informationen zu Rückrufen zwischen R/3-Systemen finden Sie unter [RFC-Funktionen zurückrufen \[Seite 19\]](#)

Detaillierte Informationen zu Rückrufen zwischen einem R/3- und einem externen System finden Sie unter [Rückrufe zwischen R/3- und externen Systemen \[Extern\]](#).

## RFC-Funktionsbausteine im Debugger

Zum Testen von "ABAP-zu-ABAP"-RFC-Aufrufen können Sie den ABAP-Debugger verwenden. Dort können Sie die Ausführung der RFC-Funktion im entfernten System überwachen. Sie können statische Haltepunkte setzen, in Einzelschritten vorwärts gehen, die Variablen beobachten und den Quellcode anzeigen.

Bei entfernten Aufrufen läuft der ABAP-Debugger (einschließlich der Debugger-Schnittstelle) auf dem lokalen System. Datenwerte und andere Laufzeitinformationen für den entfernten Funktionsbaustein werden vom entfernten System geliefert.

---

**Entfernte Destinationen pflegen**

## Entfernte Destinationen pflegen

Wählen Sie *Werkzeuge* → *Administration* → *Verwaltung* → *Netzwerk* → *RFC-Destinationen*.

Einzelheiten finden Sie in folgenden Themen:

[Destinationstypen \[Seite 63\]](#)

[Destinationen anzeigen, pflegen und testen \[Seite 59\]](#)

[Destinationsparameter eingeben \[Seite 61\]](#)

[Gruppendestinationen pflegen \[Seite 66\]](#)

## Destinationen anzeigen, pflegen und testen

Um Destinationen anzuzeigen, zu erstellen oder zu ändern, wählen Sie *Werkzeuge* → *Administration, Verwaltung* → *Netzwerk* → *RFC Destinationen*.

Entfernte Destinationen werden in Tabelle RFCDES gespeichert. Die Tabelle RFCDES beschreibt logische Destinationen für entfernte Funktionsaufrufe (RFCs).

Sie können die Tabelle RFCDES nicht direkt pflegen.



Sie können logische Destinationen auch über den R/3-Einführungsleitfaden (IMG) pflegen. Wählen Sie *Werkzeuge* → *AcceleratedSAP* → *Customizing Projektbearbeitung, SAP Referenz-IMG*.

Im Einführungsleitfaden expandieren Sie folgende Hierarchiestruktur:

*Basis*

*Application Link Enabling (ALE)*

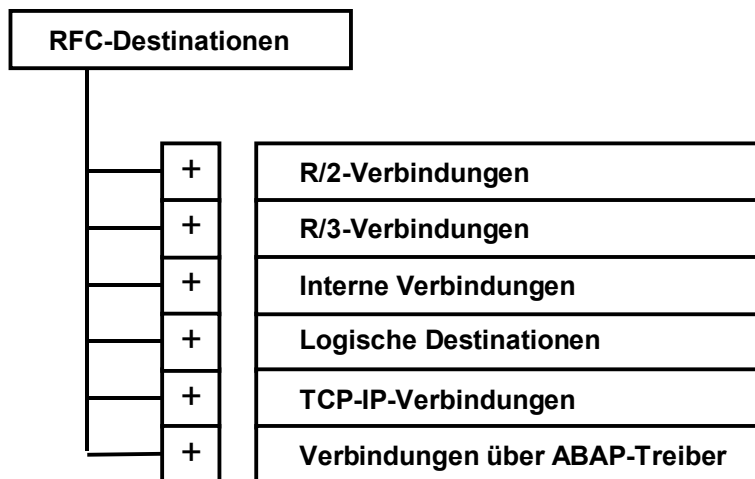
*Sender- und Empfängersysteme vorbereiten*

*Systeme im Netzwerk konfigurieren*

*Zielsysteme für RFC-Aufrufe definieren*

## Destinationen anzeigen

Das Einstiegsbild zeigt einen Baum:



Unterschiedliche Verbindungstypen (z.B. Partnersysteme oder -programme) sind möglich. Nähere Informationen finden Sie unter [Destinationstypen \[Seite 63\]](#).

Um eine Destination zu suchen, wählen Sie *Suchen* und geben Ihre Selektion an. Sie erhalten eine Liste aller passenden Einträge. Zu jedem Eintrag können Sie alle verfügbaren Informationen anzeigen.

## Destinationen anlegen

Auf dem Einstiegsbild für Destinationen sehen Sie die Verbindungstypen und alle bestehenden Destinationen in einer Baumstruktur.

## Destinationen anzeigen, pflegen und testen

Eine Erläuterung der verfügbaren Verbindungstypen finden Sie unter [Destinationstypen \[Seite 63\]](#).

Um eine neue RFC-Destination anzulegen, wählen Sie *Anlegen*. Es erscheint ein neues Bild mit leeren Feldern, die Sie ausfüllen müssen.

Wenn Sie eine entfernte Destination anlegen, können Sie einen bestimmten Anwendungs-Server oder eine Gruppe von Servern angeben, um die Systembelastung zu verteilen.

Nähere Informationen zu Destinationsparametern finden Sie unter [Destinationsparameter eingeben \[Seite 61\]](#).

## Bestehende Destinationen ändern

Auf dem Einstiegsbild für Destinationen (Transaktion SM59) sehen Sie die Verbindungstypen und alle bestehenden Destinationen in einer Baumstruktur.

Um eine bestehende RFC-Destination zu ändern, wählen Sie *Ändern*.

Nähere Informationen zu Destinationsparametern finden Sie unter [Destinationsparameter eingeben \[Seite 61\]](#).

## Destinationen testen

Um eine Destination zu testen, wählen Sie aus dem Menü *Test* die gewünschte Funktion.

- *Verbindung* (auch als Drucktaste)
- *Berechtigung* (prüft Logon-Daten)
- *Netz d. Appl. Server* (liefert eine Liste der Anwendungs-Server)

## Destinationsparameter eingeben

Zusätzlich zur *RFC-Destination* müssen Sie folgende Informationen eingeben:

### Technische Einstellungen

- *Verbindungstyp*  
Geben Sie einen bestehenden Verbindungstyp ein.  
Eine Erläuterung der verfügbaren Verbindungstypen finden Sie unter [Destinationstypen \[Seite 63\]](#).
- *Trace*  
Um die RFC-Kommunikation zu protokollieren und in einer Datei zu sichern, markieren Sie die Option *Trace*. Sie können die Datei dann mit Hilfe des Reports RSRFCTRC sowohl im rufenden als auch im gerufenen System anzeigen.
- *Lastverteilung*  
Wenn Sie Lastverteilung wählen, müssen Sie folgendes angeben:
  - *Zielsystem-ID* (Um eine Liste der verfügbaren Server zu erhalten, melden Sie sich im Zielsystem an und wählen Sie *Werkzeuge* → *Administration*, *Monitor* → *Systemüberwachung* → *Server*.)
  - *Messageserver* (Melden Sie sich im Zielsystem an und wählen Sie aus dem CCMS-Hauptmenü *Steuerung/Monitoring* → *Control Panel*. Der Message-Server ist der Server, der den Dienst M anbietet.)
  - *Gruppe* (SAP Logon-Gruppe von Servern).Sonst müssen Sie folgendes angeben:
  - *Zielmaschine*  
Den Namen einer Servermaschine des Zielsystems, den Sie als Port zum System verwenden wollen.
  - *Systemnummer*  
Den im Zielsystem verwendeten Kommunikationsservice. Um ihn zu erhalten, wählen Sie *Werkzeuge* → *Administration* → *Monitor* → *Systemüberwachung* → *Server*.

### Sicherheitsoptionen

Die folgenden Optionen sind nicht bei allen Verbindungstypen verfügbar:

- *Trusted System* (nur für Typ 3)  
Wenn das Zielsystem ein Trusted System (Vertrauenssystem) ist, wählen Sie *Yes*.  
Detaillierte Informationen zu Trusted Systems finden Sie unter [Trusted System: Vertrauensbeziehungen zwischen R/3-Systemen pflegen \[Seite 67\]](#).
- *SNC* (Secure Network Communications, verfügbar nur für die Typen 3 und T)  
Wenn Sie ein aktives SNC-unterstütztes Sicherheitssystem haben, können Sie zusätzliche Sicherheitsoptionen aktivieren, die Sie über *Destination* → *SNC-Optionen* setzen müssen.

### Beschreibung

**Destinationsparameter eingeben**

Beschreibung des Eintrags.

**Anmeldung**

- *Sprache*  
zu verwendende Systemsprache
- *Mandant*  
Mandantenummer
- *Benutzer*  
Benutzername für entfernte Anmeldung, falls abweichend von aktuellem Benutzernamen
- *Passwort*  
Benutzerkennwort
- *Aktueller Benutzer*  
Die Anmeldung am entfernten System soll unter dem aktuellen Benutzernamen erfolgen.
- *Passwort unverschlüsselt (2.0)*  
Wenn das Zielsystem ein R/3-System mit Release 2.0 ist, darf das Kennwort nicht verschlüsselt sein.

Unter *Attribute* finden Sie den Ersteller und den letzten Änderer mit den zugehörigen Daten.

## Destinationstypen

Jede Destination hat ein Feld *Verbindungstyp*, das über die Art der Systemverbindung Auskunft gibt:

- **R/2-Verbindungen (Typ 2)**

Einträge vom Typ 2 geben R/2-Systeme an. Wenn Sie einen Eintrag vom Typ 2 anlegen, müssen Sie nur den Host-Namen angeben; alle Kommunikationsinformationen sind bereits in der Hintergrundinfotabelle des SAP-Gateway-Hosts gespeichert. Falls erwünscht, können Sie Angaben zur Anmeldung machen.

Beispiel eines Namenseintrags: K50

- **R/3-Verbindungen (Typ 3)**

Einträge vom Typ 3 geben R/3-Systeme an. Wenn Sie einen Eintrag vom Typ 3 anlegen, müssen Sie den Host-Namen und den Kommunikationsservice angeben. Falls erwünscht, können Sie Angaben zur Anmeldung machen. Ab R/3-Release 3.0 können Sie zusätzlich *Lastverteilung* als Option wählen.



Ab R/3-Release 3.0 können Sie einen Anwendungs-Server des R/3-Message-Servers angeben. Dieser Anwendungs-Server wird dann entsprechend des Lastverteilungsprozesses ermittelt. Dies gilt sowohl für RFCs zwischen zwei R/3-Systemen als auch für RFCs zwischen einem R/3- und einem externen System.

Beispiel eines Namenseintrags: K11

- **Interne Verbindungen (Typ I)**

Einträge vom Typ I geben R/3-Systeme an, die mit derselben Datenbank verbunden sind wie das aktuelle System. Diese Einträge sind vordefiniert und können nicht geändert werden. Die Eintragsnamen entsprechen den Namen im SAP-Message-Server (Transaktion SM51).

Beispiel eines Namenseintrags: hs0010\_K11\_24

- hs0010=Host-Name
- K11=Systemname (Datenbankname)
- 24=TCP-Servicename

- **Logische Destinationen (Typ L)**

Anstatt eine Systemverbindung anzugeben, beziehen sich Einträge vom Typ L auf eine physikalische Destination. Dabei kann sich eine Destination vom Typ L auf weitere Destinationen vom Typ L beziehen. Ein Eintrag vom Typ L verwendet die Informationen aus dem Bezugseintrag und fügt eigene Informationen hinzu. Der Bezugseintrag enthält in der Regel die Host-Informationen, während der Eintrag vom Typ L die Anmeldeinformationen enthält. Sie können auch einen Benutzernamen, ein bestimmtes Paßwort, eine Anmeldesprache oder einen bestimmten Mandanten angeben.

Ein Eintrag vom Typ L kann sich auf andere Einträge vom Typ L beziehen.

Beispiel eines Namenseintrags: K11\_SD oder K11\_01

## Destinationstypen

- K11=Name des RFCDES-Eintrags für das R/3-System K11
- SD oder 01: für die Felder Benutzer='SD\_INPUT' oder Mandant='001'

- **Verbindungen über ABAP-Treiber (Typ X)**

Einträge vom Typ X geben Systeme an, in denen Gerätetreiber in ABAP gesondert installiert wurden. Wenn Sie einen Eintrag vom Typ X anlegen, müssen Sie den Namen des ABAP-Gerätetreibers angeben.

- **TCP/IP-Verbindungen (Typ T)**

Destinationen vom Typ T sind Verbindungen zu externen Programmen, die die RFC-Bibliothek zum Empfang von RFCs verwenden. Die Aktivierungsart kann entweder *Anstarten* oder *Registrierung* sein.

Bei *Anstarten auf* müssen Sie den Host-Namen und den Pfadnamen des Programmes angeben, das Sie starten wollen.

### **Aktivierungsart *Anstarten auf***

Die Kommunikationsmethode hängt davon ab, wie Sie die Programmposition wählen:

- **Explizitem Host**

In diesem Fall wird das Programm entweder vom Standard-Gateway des Systems oder vom explizit angegebenen Gateway (gwr) über die entfernte Shell gestartet. Geben Sie `/etc/ping <Hostname>` ein, um sicherzustellen, daß der Computer mit dem Gateway-Prozeß auf den angegebenen Computer zugreifen kann.

Damit Sie über die entfernte Shell ein Programm auf einem anderen Computer starten können, muß das Zielsystem bestimmte Bedingungen erfüllen.

- Die Benutzer- ID des Gateway-Prozesses muß existieren und eine Datei mit Namen `.rhosts` muß im Verzeichnis des Benutzers liegen.
- Die Datei `.rhosts` muß den Namen des rufenden Computers enthalten.

Um dies zu überprüfen, melden Sie sich mit der geeigneten Benutzer-ID an dem Computer an, der den Gateway-Prozeß enthält, und geben folgenden Befehl ein: `remsh <Hostname> <Programmname>`. Dabei müssen `<Hostname>` und `<Programmname>` mit denen aus SM59 übereinstimmen. (Wenn Sie ein RFC-Server-Programm ohne Parameter aufrufen, gibt der *RfcAccept*-Aufruf immer einen Rückgabewert zurück (RFC\_HANDLE\_NULL), und das Programm bricht sofort ab.)

- **Applikationsserver**

Wenn Sie *Applikationsserver* wählen und Ihr Programm angeben, können Sie das Programm vom SAP-Anwendungs-Server aus starten.

Stellen Sie jedoch zuerst sicher, daß der SAP-Anwendungs-Server auf das Programm zugreifen kann und daß er die Berechtigung hat, das Programm zu starten.

Um dies zu überprüfen, melden Sie sich mit der Benutzer-ID des SAP-Anwendungs-Servers an (z.B. c11adm). Falls möglich, wechseln Sie in das Arbeitsverzeichnis des SAP-Anwendungs-Servers (`/usr/sap/.../D.../work`) und versuchen Sie, das RFC-Server-Programm von dort aus von Hand zu starten. (Wenn Sie ein RFC-Server-Programm ohne Parameter aufrufen, gibt der *RfcAccept*-Aufruf immer einen Rückgabewert zurück (RFC\_HANDLE\_NULL), und das Programm bricht sofort ab.)

## Destinationstypen

– **Front End-Workstation**

Wenn Sie *Front End-Workstation* wählen und Ihr Programm angeben, können Sie das Programm aus dem SAPGUI starten.

Stellen Sie sicher, daß Sie mit SAPGUI auf das Programm zugreifen können.

Stellen Sie sicher, daß SAPGUI die Berechtigung hat, das Programm zu starten.

Um dies zu überprüfen, rufen Sie einfach das RFC-Server-Programm in Ihrer Umgebung auf.

Der Aufruf des Funktionsbausteins kann auch transaktional sein (CALL FUNCTION... IN BACKGROUND TASK DESTINATION...).

**Aktivierungsart *Registrierung***

Wenn Sie *Registrierung* wählen, müssen Sie ein registriertes RFC-Programm angeben. Mit einem SAP-Gateway können Sie ein RFC-Server-Programm unter dieser ID registrieren und dann auf RFC-Aufrufe aus anderen SAP-Systemen warten.

Beispiel eines Namenseintrags: SERVER\_EXEC

• **Typ M**

Einträge vom Typ M sind asynchrone RFC-Verbindungen zu R/3-Systemen über CMC (Protokoll X.400).

• **Typ S**

Typ S entspricht Typ 2, außer daß die Destination SNA oder APPC ist.

## Gruppendestinationen pflegen

## Gruppendestinationen pflegen

Um eine gleichmäßige Lastverteilung im Zielsystem zu erreichen, müssen Sie eine Gruppe von Anwendungs-Servern als eine RFC-Destination definieren. Die Gruppendestination kann bei der Durchführung von Parallelverarbeitungsaufgaben nur in Verbindung mit dem [asynchronen RFC \[Seite 40\]](#) verwendet werden.

Die auf jedem einzelnen Anwendungs-Server verfügbaren Ressourcen hängen von der aktuellen Systembelastung ab.

So zeigen Sie RFC-Gruppen an und pflegen sie:

1. Auf dem Übersichtsbild über die RFC-Destinationen wählen Sie *RFC* → *RFC-Gruppen*.

Sie sehen dann

- die Namen aller bereits definierten RFC-Gruppen
- eine Liste der Instanzen in Ihrem R/3-System (Host, System- und Instanznummer)
- den aktuellen Status (aktiv oder nicht) eines jeden Servers.

2. Um eine RFC-Gruppe zu definieren, wählen Sie *Bearbeiten* → *Anlegen*. Im folgenden Dialogfenster geben Sie einen Server-Gruppennamen und eine Instanz ein.

Um Instanzen zu einer bestehenden Gruppe hinzuzufügen, klicken Sie zweimal auf die entsprechende Gruppe und geben im nachfolgenden Dialogfenster die neue Instanz ein.

Sie können einen Server auch zu mehreren Gruppen zuordnen, indem Sie Einträge doppelt anlegen. Jobs, die diese Gruppen verwenden, werden dann auf dem oder den gemeinsam genutzten Server(n) um freie Arbeitsprozesse streiten.



### Beispiele zur Verwendung:

Sie können Gruppen einsetzen, um verschiedene parallelverarbeitete Jobs zur selben Zeit laufen zu lassen, ohne daß sie sich um denselben Server streiten. Dazu müssen Sie für die von den Jobs verwendeten unterschiedlichen Gruppen unterschiedliche Server angeben.

Sie können Gruppen auch dazu verwenden, um die Verarbeitung von den Servern fernzuhalten, auf denen Benutzerdialoge aktiv sind. In diesem Fall müssen Sie für die zur Verarbeitung verwendeten Gruppen andere Server angeben als in den Anmeldegruppen für Benutzer.

## Trusted System: Vertrauensbeziehungen zwischen R/3-Systemen pflegen

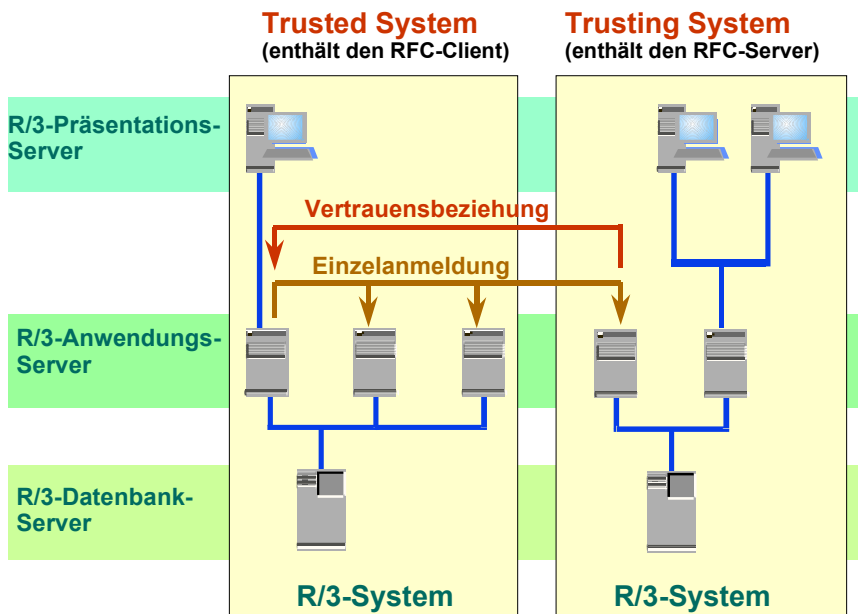


## Trusted System: Vertrauensbeziehungen zwischen R/3-Systemen pflegen

R/3-Systeme können Vertrauensbeziehungen zueinander aufbauen.

Wenn ein rufendes R/3-System dem gerufenen System als Trusted System bekannt ist, ist kein Paßwort erforderlich.

Das rufende R/3-System muß im gerufenen System als Trusted System registriert sein. Das gerufene System wird als Trusting System (vertrauendes System) bezeichnet.



Vertrauensbeziehungen zwischen R/3-Systemen haben folgende Vorteile:

- Einzelanmeldung ist über Systemgrenzen hinweg möglich.
- Im Netzwerk werden keine Paßwörter übertragen.
- Auszeitmechanismen für die Anmeldedaten schützen vor widerrechtlichen Anmeldeversuchen.
- Benutzerspezifische Anmeldedaten werden im Trusting System geprüft.

Mit dieser Funktionalität können Sie ein virtuelles R/3-System anlegen, das aus mehreren entfernt gerufenen R/3-Systemen besteht. Daten für die entfernte Anmeldung werden im Trusting System geprüft.

Die Vertrauensbeziehung ist nicht gegenseitig, d.h. sie besteht nur in eine Richtung. Um eine beidseitige Vertrauensbeziehung zwischen zwei Partnersystemen aufzubauen, müssen Sie jedes der beiden Systeme im entsprechenden Partnersystem als Trusted System definieren.

Um zusätzliche Sicherheit zu erhalten, können Sie die SAP-SNC-Schnittstelle (Secure Network Communications) für zusätzliche Sicherheitssysteme wie Kerberos und SECUDE verwenden.

## Trusted System: Vertrauensbeziehungen zwischen R/3-Systemen pflegen

### Trusted Systeme anzeigen, pflegen und testen

So zeigen Sie ein Trusted System im Trusting System an oder pflegen es:

1. Um ein R/3-System als Trusted System zu definieren, müssen Sie zunächst eine logische Destination anlegen, die eine solche Vertrauensbeziehung ermöglicht. Näheres dazu finden Sie unter [Entfernte Destinationen pflegen \[Seite 58\]](#).
2. Auf dem Übersichtsbild über die RFC-Destinationen (Transaktion SM59) wählen Sie *RFC → Trusted Systeme* (oder Transaktionscode SMT1).
3. Gibt es bereits Trusted Systeme, so werden Sie in einem Hierarchiebaum angezeigt. Um bestehende Trusted Systeme anzuzeigen, expandieren Sie die entsprechenden Knoten.
4. Um ein Trusted System anzulegen, wählen Sie *Anlegen*.
5. Im Dialogfenster geben Sie die Destination für das entfernte System ein. Um eine Destination zu ändern, lesen Sie *Destinationen für Trusted Systeme ändern* weiter unten.
6. Alle benötigten Informationen wie Name des Anwendungs-Servers und Sicherheitsschlüssel werden automatisch bereitgestellt.
7. Wenn Sie die Gültigkeitsdauer der Anmeldedaten begrenzen wollen, geben Sie im Feld *Gültigkeitsdauer* ein Enddatum ein.
8. Wenn Sie den Transaktionscode des rufenden Programms in das gerufene System übernehmen wollen, markieren Sie das entsprechende Ankreuzfeld.  
  
Nur dann wird im gerufenen System eine Berechtigungsprüfung für den Transaktionscode durchgeführt (Feld RFC\_TCODE des Berechtigungsobjekts S\_RFCACL, siehe *Anmeldeberechtigungsprüfungen im Trusting System* weiter unten).
9. Um eine Vertrauensbeziehung zu löschen, zeigen Sie die entsprechende Beziehung durch Doppelklick an und wählen dann *Löschen*.



Beim Löschen einer Vertrauensbeziehung erscheint das Anmeldebild des entsprechenden Systems, wenn keine gültigen Anmeldedaten vorliegen. In diesem Fall müssen Sie sich in diesem System anmelden, um den Löschvorgang durchzuführen.

### Destinationen für Trusted Systeme ändern

Sie können bestehende Destinationen für jedes System auf dem Pflegebild für Trusted Systeme ändern (*RFC → Trusted Systeme*, Transaktionscode SMT1), indem Sie *Destinationspflege* wählen.

In Trusted Systemen werden Destinationen für Trusting Systeme automatisch generiert. Diese Destinationen werden verwendet, wenn Sie Trusting Systeme über *RFC → Trusting Systeme* anzeigen (Transaktionscode SMT2).

Um zu verhindern, daß andere Benutzer Ihre Destinationen ändern, markieren Sie das Ankreuzfeld *Destination nicht änderbar* im *Attribute*-Teil. Um die Destination wieder für Änderungen freizugeben, machen Sie einen Doppelklick auf das Ankreuzfeld.

Beachten Sie, daß Destinationen konsistent bleiben müssen. Aus diesem Grund dürfen Sie weder die ID des Zielsystems noch die Systemnummer oder den Destinationsnamen ändern.

## Trusted System: Vertrauensbeziehungen zwischen R/3-Systemen pflegen

### Trusting Systeme anzeigen

In einem Trusted System können Sie eine Liste aller Trusting Systeme anzeigen. Wählen Sie dazu *RFC* → *Trusting Systeme*.

Um den Anwendungs-Server eines Trusting Systems anzuzeigen, klicken Sie auf den Namen des entsprechenden Systems. Die Namen von Anwendungs-Servern enthalten das Suffix *\_TRUSTED*.

Durch einen Doppelklick auf den Namen eines Anwendungs-Servers erhalten Sie ein Dialogfenster, in dem Sie den Transaktionscode einer Transaktion eingeben können, die Sie im Trusting System ausführen wollen. Zusätzlich können Sie angeben, ob die Transaktion im selben Modus oder in einem neuen Modus ausgeführt werden soll.

### Prüfung der Anmeldeberechtigung im Trusting System

Die Anmeldedaten zur Anmeldung an einem Trusting System werden einer Berechtigungsprüfung unterzogen.

In den vom Trusted System gelieferten Daten werden Systemname, Mandant, Benutzername und andere optionale Daten gesucht und gegen die Feldwerte des Berechtigungsobjekts *S\_RFCACL* geprüft.

Der Systemadministrator kann die Anmeldedaten eines Benutzers mit Hilfe des Funktionsbausteins *AUTHORITY\_CHECK\_TRUSTED\_SYSTEM* prüfen.

Eine Beschreibung der möglichen Rückgabewerte finden Sie unter *Fehlersuche in Trusted/Trusting Systemen* weiter unten.

### Trusted Systeme testen

Um ein Trusted System zu testen, können Sie Berechtigungsprüfungen für den aktuellen Server und das Trusting System durchführen. Wählen Sie dazu das Menü *Eintrag*. Stehen keine gültigen Anmeldedaten bereit, so erscheint das Anmeldebild des Trusted Systems. Melden Sie sich dort an. Schlägt Ihr Test fehl, lesen Sie *Fehlersuche in Trusted/Trusting Systemen* weiter unten.

### Fehlersuche in Trusted/Trusting Systemen

Nach dem Anlegen eines Trusted Systems müssen Sie die Destination testen. Melden Sie sich dazu über *Remote Login* im Trusted System an.

Alternativ dazu können Sie auch eine Berechtigungsprüfung für den Trusted Server durchführen. Wählen Sie dazu die geeignete Funktion aus dem *Test*-Menü.

Schlägt Ihr Anmeldeversuch fehl, erhalten Sie eine entsprechende Nachricht mit einem Fehlercode. Beachten Sie, daß Sie die Benutzer *DDIC* und *SAP\** nicht verwenden dürfen.

Folgende Fehlercodes können auftreten:

- 0 Ungültige Anmeldedaten (Benutzer-ID und Mandant) für das Trusting System  
Lösung: Legen Sie die Benutzer-ID für den Mandanten im Trusting System an.
- 1 Für das rufende System existiert kein Trusted System-Eintrag, oder der Sicherheitsschlüssel für das System ist ungültig.  
Lösung: Legen Sie den Eintrag für das Trusted System erneut an.
- 2 Der Benutzer hat keine Berechtigung für das Trusted System (Objekt *S\_RFCACL*).

**Trusted System: Vertrauensbeziehungen zwischen R/3-Systemen pflegen**

Lösung: Geben Sie dem Benutzer die erforderlichen Berechtigungen.

- 3 Der Zeitstempel der Anmeldedaten ist ungültig.

Lösung: Überprüfen Sie die Zeiteinstellung sowohl auf dem Client- als auch auf dem Server-Host sowie das Ablaufdatum der Anmeldedaten. (Beachten Sie, daß die Standardeinstellung 00:00:00 "ohne zeitliche Begrenzung" bedeutet.)

Mit Hilfe des Funktionsbausteins `AUTHORITY_CHECK_TRUSTED_SYSTEM` können Sie feststellen, ob im Trusting System korrekte Anmeldedaten für das Trusted System vorliegen.

Wenn alle Tests erfolgreich verlaufen sind und Sie dennoch nicht auf das Trusting System zugreifen können, aktualisieren Sie die entsprechenden Datenbankpuffer. Wählen Sie dazu *Werkzeuge* → *Administration, Benutzerpflege* → *Benutzer anzeigen, Umfeld* → *Massenänderungen* → *alle B.puffer rücks.*



Um den Auslöser eines Fehlers zu finden, aktivieren Sie den Trace auf dem Destinationen-Detailbild, reproduzieren den Fehler, und lesen im Kurz-Dump des Trusting Systems die zur Fehler-ID `CALL_FUNCTION_SINGLE_LOGIN_REJ` angezeigten Informationen.