Natural

Natural RPC (Remote Procedure Call)

Version 8.2.8

April 2023

ADABAS & NATURAL

Dieses Dokument gilt für Natural ab Version 8.2.8.

Hierin enthaltene Beschreibungen unterliegen Änderungen und Ergänzungen, die in nachfolgenden Release Notes oder Neuausgaben bekanntgegeben werden.

Dokument-ID: NATMF-NATRPC-828-20230425DE

Inhaltsverzeichnis

Vorwort

Remote-Procedure-Call (RPC)-Techniken bilden einen Rahmen für die Kommunikation zwischen Server- und Client-Systemen, die sich auf demselben Computer oder in einem Netz aus identischen oder heterogenen Maschinen und Betriebssystemen befinden können. Es sind mehrere grundsätzlich ähnliche Methoden bekannt.

Diese Dokumentation beschreibt die Funktionsweise und den Einsatz der von Natural bereitgestellten RPC-Techniken, um den Entwurf und die Anwendung verteilter Softwaresysteme zu vereinfachen. Informationen zu anderen Produkten, die an einer Natural RPC-basierten Umgebung beteiligt sein können, finden Sie in der Dokumentation zu EntireX RPC for 3GL, Entire Network und EntireX Broker.

Ausführliche Informationen zu den Funktionen, die zur Verwaltung von Remote Procedure Calls zur Verfügung stehen, finden Sie im Kapitel *SYSRPC Utility* in der *Debugger und Dienstprogramme*-Dokumentation.

Diese Dokumentation ist in die folgenden Kapitel gegliedert:

Einführung in Natural RPC	Enthält grundlegende Informationen, z. B. über den Betrieb eines Natural RPC im nicht-konversationellen und im konversationellen Modus; beschreibt die Datenbank-Transaktionen auf Client- und Serverseite und enthält eine Liste wichtiger Schlüsselbegriffe, die in der `SYSRPC` Utility-Dokumentation und in der Natural RPC-Dokumentation verwendet werden.
Voraussetzungen und vorbereitende Informationen	Bietet einen Überblick über die allgemeinen Voraussetzungen und eine kurze Beschreibung der Möglichkeiten, die in Natural für die Implementierung einer Natural RPC (Remote Procedure Call)-Umgebung vorhanden sind, sowie Informationen über die spezifische Abbildung (Mapping) von Software AG IDL-Datentypen, -Gruppen, -Arrays und -Strukturen in der Programmiersprache Natural.
Grenzen und Einschränkungen	Informiert Sie über einige Grenzen und Einschränkungen, die Sie bei der Verwendung des Natural RPC beachten sollten.
Einrichten einer Natural RPC-Umgebung	Beschreibt die grundlegenden Schritte, die Sie bei allen Client- und Server-Naturals durchführen müssen, um eine Natural RPC-Umgebung einzurichten.
Starten eines Natural RPC Servers	Beschreibt, wie Sie einen Natural RPC Server auf den verschiedenen Plattformen starten.
Beenden eines Natural RPC Servers	Beschreibt die verschiedenen Methoden zum Beenden eines Natural RPC Servers, zum Beispiel durch Beenden des EntireX Broker-Dienstes, der die RPC-Verbindung unterstützt.
Betrieb einer Natural RPC-Umgebung	Beschreibt hauptsächlich Aufgaben, die für den Betrieb einer Natural-RPC-Umgebung auszuführen sind.

Verwendung des Natural RPC im konversationellen Modus	Beschreibt das Öffnen/Schließen einer Konversation, das Definieren eines Konversationskontexts und das Ändern der Systemvariablen `*CONVID`, wenn mehrere Konversationen parallel verwendet werden.
Reliable RPC	Beschreibt den Reliable RPC, die Natural RPC-Implementierung eines zuverlässigen Messaging-Systems.
Verwendung eines Remote Directory Servers (RDS)	Beschreibt das Prinzip und die Verwendung eines RDS: wie eine RDS-Schnittstelle, eine Remote-Directory-Service-Routine erstellt wird. Enthält Informationen über das RDS-Directory-Service-Programm `RDSSCDIR`, das zum Lesen von Verzeichnisinformationen aus einer Arbeitsdatei erforderlich ist.
Verwendung von Security	Beschreibt, wie der Natural RPC mit Natural Security oder EntireX Security verwendet werden kann.
EntireX Broker-Unterstützung	Besondere Aspekte der EntireX Broker-Unterstützung.
API zur Bereitstellung eines RPC-Kontexts von der Natural RPC-Client-Seite	Beschreibt die Anwendungsprogrammierschnittstelle RPC-CNTX.
APIs zur Verwendung beim Natural RPC	Übersicht über Natural-Anwendungsprogrammierschnittstellen (APIs) in der Natural Library SYSEXT, die beim Natural RPC verwendet werden können, um Informationen abzufragen oder zu ändern oder Dienste zu benutzen, die nicht über Natural-Statements erreichbar sind.

Anmerkung: Dieses Dokument gilt für alle Plattformen, auf denen Natural verwendet werden kann. Je nachdem, welche Natural-Dokumentation Sie gerade verwenden, gibt es jedoch die folgenden Unterschiede:

- Die Beispiele für die Verwendung der Natural Utility `SYSRPC` zeigen plattformspezifische Abbildungen (entweder mit GUI- oder CUI-Schnittstelle).
- Bei Natural for Windows, UNIX und OpenVMS sind die RPC-spezifischen Parameter als Profilparameter verfügbar.
- Bei Natural für Großrechner sind die Parameter als Schlüsselwort-Subparameter des Profilparameters `RPC` oder des Parameter-Makros `NTRPC` verfügbar.

Notation *vrs* oder *vr*

Die in dieser Dokumentation verwendete Schreibweise *vrs* oder *vr* steht für die jeweilige Produktversion (siehe auch *Version* im *Glossar*).

1 Über diese Dokumentation

Dokumentationskonventionen

Konvention	Beschreibung
Fettschrift	>Kennzeichnet Elemente auf einem Bildschirm.
`Nichtproportionale Schrift`	Kennzeichnet Namen und Orte von Diensten im Format *`Ordner.Unterordner.Dienst`*, Programmierschnittstellen (APIs), Namen von Klassen, Methoden und Properties in Java.
`Kursivschrift`	Kennzeichnet: Variablen, für die Sie situations- oder umgebungsspezifische Werte angeben müssen. Neue Begriffe, wenn sie erstmals im Text auftreten. Verweise auf andere Dokumentationsquellen.
`Nichtproportionale Schrift`	Kennzeichnet: Text, den Sie eingeben müssen. Meldungen, die vom System angezeigt werden. Programmcode.
{ }	Zeigt eine Reihe von Auswahlmöglichkeiten an, von denen Sie eine auswählen müssen. Geben Sie nur die innerhalb der geschweiften Klammern vorhandenen Informationen ein. Geben Sie nicht die Klammersymbole { } ein.
|	Trennt zwei sich gegenseitig ausschließende Auswahlmöglichkeiten in einer Syntaxzeile voneinander ab. Geben Sie eine der Auswahlmöglichkeiten ein. Geben Sie nicht das Symbol | ein.
[ ]	Zeigt eine oder mehrere Optionen an. Geben Sie nur die innerhalb der eckigen Klammern vorhandenen Informationen ein. Geben Sie nicht die Klammersymbole [ ] ein.
...	Zeigt an, dass Sie mehrere Auswahlmöglichkeiten desselben Typs eingeben können. Geben Sie nur die Informationen ein. Geben Sie nicht die drei Auslassungspunkte (...) ein.

Online-Informationen und Support

Produktdokumentation

Sie finden die Produktdokumentation auf unserer Dokumentationswebsite unter **https://documentation.softwareag.com**.

Zusätzlich können Sie auch über **https://www.softwareag.cloud** auf die Dokumentation für die Cloud-Produkte zugreifen. Navigieren Sie zum gewünschten Produkt und gehen Sie dann, je nach Produkt, zu „Developer Center“, „User Center“ oder „Documentation“.

Produktschulungen

Sie finden hilfreiches Produktschulungsmaterial auf unserem Lernportal unter **https://knowledge.softwareag.com**.

Tech Community

Auf der Website unserer Tech Community unter **https://techcommunity.softwareag.com** können Sie mit Experten der Software AG zusammenarbeiten. Von hier aus können Sie zum Beispiel:

- Unsere umfangreiche Wissensdatenbank durchsuchen.
- In unseren Diskussionsforen Fragen stellen und Antworten finden.
- Die neuesten Nachrichten und Ankündigungen der Software AG lesen.
- Unsere Communities erkunden.
- Unsere öffentlichen Repositories auf GitHub and Docker unter **https://github.com/softwareag** und **https://hub.docker.com/publishers/softwareag>** besuchen und weitere Ressourcen der Software AG entdecken.

Produktsupport

Support für die Produkte der Software AG steht lizenzierten Kunden über unser Empower-Portal unter **https://empower.softwareag.com>** zur Verfügung. Für viele Dienstleistungen auf diesem Portal benötigen Sie ein Konto. Wenn Sie noch keines haben, dann können Sie es unter **https://empower.softwareag.com/register>** beantragen. Sobald Sie ein Konto haben, können Sie zum Beispiel:

- Produkte, Aktualisierungen und Programmkorrekturen herunterladen.
- Das Knowledge Center nach technischen Informationen und Tipps durchsuchen.
- Frühwarnungen und kritische Alarme abonnieren.
- Supportfälle öffnen und aktualisieren.
- Anfragen für neue Produktmerkmale einreichen.

Datenschutz

Die Produkte der Software AG stellen Funktionen zur Verarbeitung von personenbezogenen Daten gemäß der Datenschutz-Grundverordnung (DSGVO) der Europäischen Union zur Verfügung. Gegebenenfalls sind in der betreffenden Systemverwaltungsdokumentation entsprechende Schritte dokumentiert.

2 Einführung in Natural RPC

Allgemeine Informationen zum Natural RPC

Zweck des Natural RPC

Der Natural Remote Procedure Call (RPC) ermöglicht es einem Natural-Client-Programm ein `CALLNAT`-Statement abzusetzen, um ein Subprogramm in einem Natural-Server aufzurufen. Die Natural-Client- und Server-Sitzungen können auf demselben oder auf einem anderen Computer laufen. So kann beispielsweise ein Natural-Client-Programm auf einem Windows-Computer ein `CALLNAT`-Statement an einen Großrechner-Server senden, um Daten aus einer Großrechner-Datenbank abzurufen. Derselbe Windows-Computer kann als Server fungieren, wenn ein Natural-Client-Programm, das z. B. unter UNIX läuft, ein `CALLNAT`-Statement absetzt, um Daten von diesem Natural-Server anzufordern.

Vorteile von Natural RPC-Aufrufen

Beim Natural RPC werden die Vorteile des Client-Server-Computing genutzt. In einem typischen Szenario greift Natural auf einem Windows-Client-Computer über eine Middleware-Schicht auf Serverdaten von einem Natural auf einem Großrechner zu. Daraus ergeben sich folgende Vorteile:

- Der Endbenutzer auf der Client-Seite kann eine Natural-Anwendung mit einer grafischen Benutzeroberfläche nutzen.
- Der Zugriff auf eine große Datenbank kann auf einem Großrechner-Server erfolgen.
- Der Netzwerkverkehr kann minimiert werden, wenn nur relevante Daten vom Client zum Server und zurückgesendet werden.

Natural RPC-Betriebsarten

Der Natural RPC bietet folgende Betriebsarten:

- **Nicht-konversationeller Modus** (in den folgenden Texten ist, wenn nicht anders angegeben, dieser Modus gemeint)
- **Konversationeller Modus**

Diese Betriebsarten werden in den folgenden Abschnitten beschrieben. Eine Gegenüberstellung der Vor- und Nachteile dieser Betriebsarten finden Sie unter *Konversationeller versus nicht-konversationeller Modus*.

Verfügbarkeit auf verschiedenen Plattformen

Sie können den Natural RPC auf verschiedenen Plattformen unter den folgenden Betriebssystemen verwenden:

Großrechner-Umgebungen

- z/OS
- z/VSE
- BS2000

Natural RPC auf Großrechnern wird unter den folgenden TP-Monitoren unterstützt:

- Com-plete
- CICS
- IMS TM
- TSO
- TIAM
- *open*UTM

Natural RPC ist auch im Batch-Modus verfügbar.

Andere Umgebungen

- Windows
- UNIX
- OpenVMS

Auf all diesen Plattformen kann Natural sowohl als Client-Natural als auch als Server-Natural fungieren.

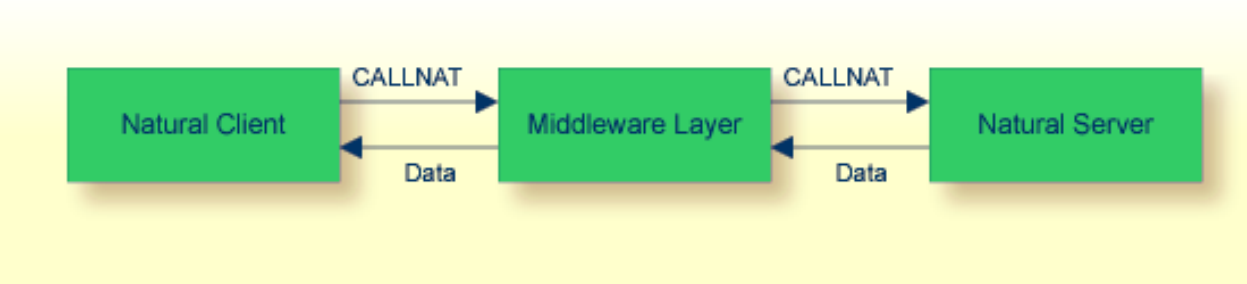
Unterstützung von Nicht-Natural-Umgebungen (EntireX RPC)

Nicht-Natural-Umgebungen (3GL und andere Programmiersprachen) werden sowohl auf der Client- als auch auf der Server-Seite unterstützt. So kann ein Nicht-Natural-Client mit einem Natural RPC Server und ein Natural-Client mit einem Nicht-Natural-RPC-Server kommunizieren. Ermöglicht wird dies durch die Verwendung des EntireX RPC.

Natural RPC-Betrieb im nicht-konversationellen Modus

Der nicht-konversationelle Modus sollte nur verwendet werden, um einen einzelnen Datenaustausch mit einem Partner durchzuführen. Siehe auch **Konversationeller versus nicht-konversationeller Modus**.

Die Natural RPC-Technik verwendet das Natural-Statement `CALLNAT`, so dass sowohl lokale als auch entfernte Subprogramm-Aufrufe parallel abgesetzt werden können. Entfernte Programmaufrufe arbeiten synchron. Als entfernter Prozeduraufruf würde ein `CALLNAT`, einfach ausgedrückt, den folgenden Weg nehmen:



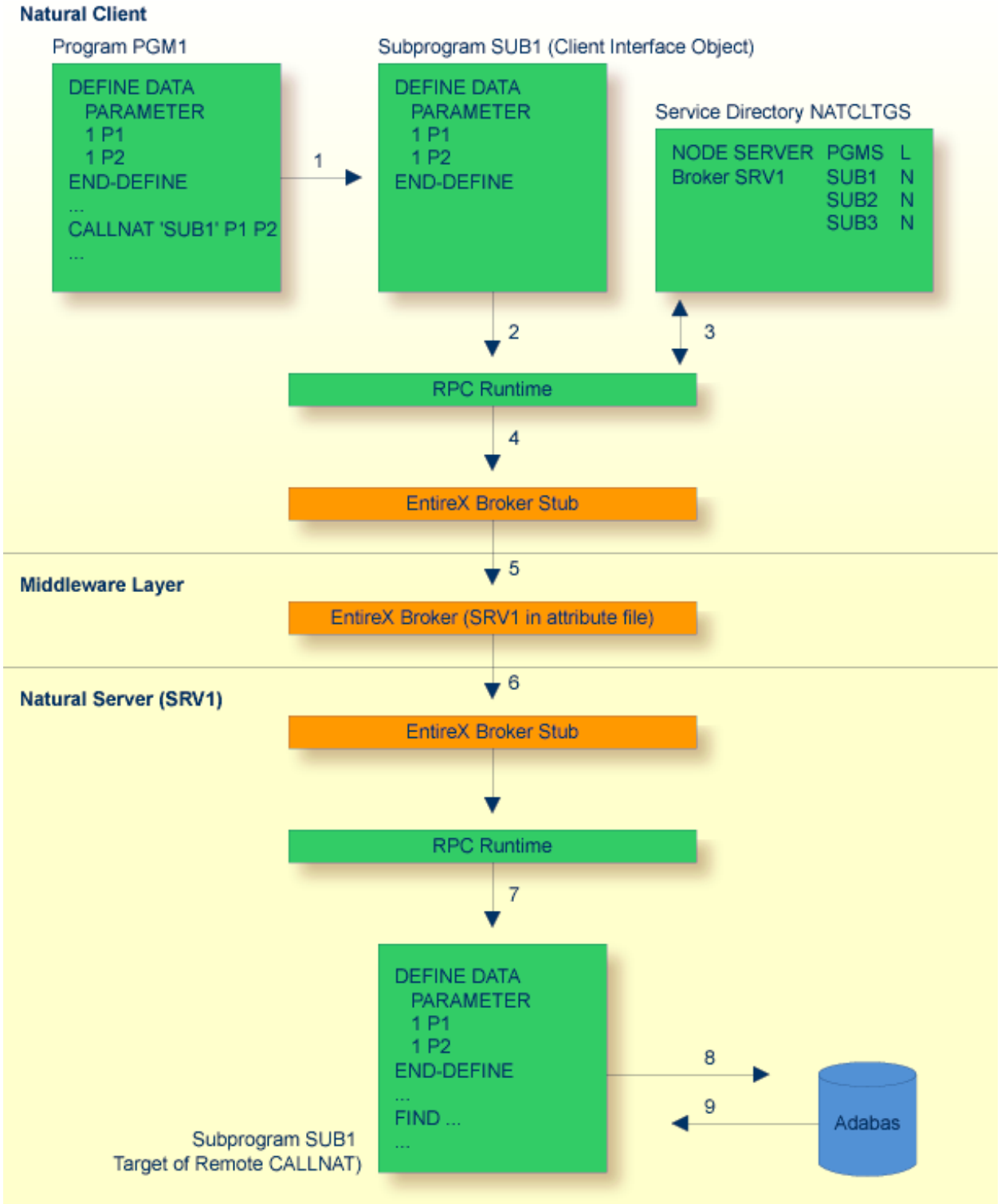
Das vom Natural-Client abgesetzte `CALLNAT` wird über eine Middleware-Schicht an den Natural-Server geleitet, der die Daten an den Client zurückgibt.

In der Regel besteht die Middleware-Schicht aus dem Software AG-Produkt EntireX Broker, das das ACI-Protokoll verwendet. EntireX Broker verwendet entweder Entire Net-Work oder TCP/IP als Kommunikationsschicht.

Ein detailliertes Beispiel für den RPC-Kontrollfluss wird im Folgenden beschrieben.

Ausgabe von CALLNATs in einer RPC-Umgebung

Einzelheiten des `CALLNAT`-Kontrollflusses in einer Remote-Prozedur werden im Folgenden dargestellt. Aus Gründen der Übersichtlichkeit ist der Rückweg nicht dargestellt, aber er verläuft analog; die Zahlen beziehen sich auf die Beschreibung:



1. Vom Natural-Client aus sendet das Programm `PGM1` einen `CALLNAT` an das Subprogramm `SUB1`. `PGM1` weiß nicht, ob das Ergebnis seines `CALLNAT` ein lokales oder ein entferntes `CALLNAT` sein wird.

Da sich das Ziel SUB1 auf einem Server befindet, greift das CALLNAT stattdessen auf ein Interface-Objekt SUB1 zu. Dieses Client-Interface-Objekt wurde automatisch oder manuell (mit der Funktion **Interface Object Generation** (IG) der SYSRPC-Utility) erstellt.

Das Interface-Objekt hat den gleichen Namen wie das Ziel-Subprogramm und enthält Parameter, die mit den im Programm PGM1 und auf dem Server im Ziel-Subprogramm SUB1 verwendeten Parametern identisch sind. Außerdem enthält es Steuerinformationen, die intern vom RPC verwendet werden.

Wenn der Schlüsselwort-Subparameter AUTORPC des Natural-Profilparameters RPC oder des Parametermakros NTRPC auf ON gesetzt ist und Natural das Subprogramm in der lokalen Umgebung nicht finden kann, interpretiert Natural dies als Remote Procedure Call und generiert den Parameterdatenbereich (PDA) dynamisch zur Laufzeit.

Außerdem versucht Natural, dieses Subprogramm im Service Directory NATCLTGS zu finden.

Weitere Informationen zur Interface-Objekt-Generierungsfunktion der SYSRPC-Utility siehe *Interface-Objekte erstellen*.

Wenn Sie ohne Interface-Objekte arbeiten wollen, lesen Sie bitte den Abschnitt *Mit automatischer Natural RPC-Ausführung arbeiten*.

2. Das Interface-Objekt richtet dann ein CALLNAT zu einer RPC-Client-Service-Routine ein.
3. Die Client-RPC-Laufzeit überprüft im Service Directory NATCLTGS, auf welchem Knoten und Server das CALLNAT ausgeführt werden soll und ob eine Anmeldung erforderlich ist.

   Die CALLNAT-Daten einschließlich der Parameterliste und, falls erforderlich, der Anmeldedaten werden an eine Middleware-Schicht übergeben.
4. In diesem Beispiel besteht die Middleware-Schicht aus dem Software AG-Produkt EntireX Broker. Daher werden die CALLNAT-Daten zunächst an einen EntireX Broker Stub auf dem Client übergeben.
5. Vom EntireX Broker Stub aus werden die CALLNAT-Daten an den EntireX Broker weitergeleitet. Der EntireX Broker ist ein Produkt, das an folgenden Stellen installiert sein kann:
   - auf dem Client-Computer,
   - auf dem Server-Computer oder
   - auf einer dritten Plattform.

   Damit die Daten erfolgreich weitergeleitet werden können, muss der Server SRV1 in der EntireX Broker-Attributdatei definiert sein und SRV1 muss bereits aktiv sein, sich also beim EntireX Broker registriert haben.

   Informationen zur Definition von Servern in der EntireX Broker-Attributdatei finden Sie in der *EntireX Broker*-Dokumentation.
6. Von der Middleware-Schicht werden die CALLNAT-Daten an den EntireX Broker-Stub auf der Natural Server-Plattform und von dort an die RPC-Server-Service-Routine weitergeleitet.

Die RPC-Server-Service-Routine validiert die Anmeldedaten (falls vorhanden) und führt eine Anmeldung durch (falls angefordert).

7. Die RPC-Server-Service-Routine ruft das Ziel-Subprogramm `SUB1` auf und übergibt die Daten, falls angefordert.

   Zu diesem Zeitpunkt verfügt das Ziel-Subprogramm `SUB1` über alle erforderlichen Daten zur Ausführung, als ob es von einem lokalen Programm `PGM1` aufgerufen worden wäre.
8. Dann kann das Subprogramm `SUB1` z.B. ein `FIND`-Statement an die Adabas-Datenbank des Servers senden. `SUB1` weiß nicht, ob es von einem lokalen oder von einem entfernten `CALLNAT` gestartet wurde.
9. Adabas findet die Daten und übergibt sie an `SUB1`.

   Anschließend gibt `SUB1` die Adabas-Daten an die aufrufende Server-Service-Routine zurück. Von dort werden sie über die Middleware-Schicht an `PGM1` zurückgegeben. Dies geschieht auf demselben Weg wie in den Schritten 1 bis 8 beschrieben, jedoch in umgekehrter Reihenfolge.

Natural RPC-Betrieb im konversationellen Modus

Ein konversationeller RPC ist eine statische Verbindung von begrenzter Dauer zwischen einem Client und einem Server. Sie bietet eine Reihe von Diensten (Subprogrammen), die vom Client definiert werden und die alle innerhalb einer Server-Task ausgeführt werden, die für die Dauer der Konversation ausschließlich dem Client zur Verfügung steht. Sie wird in einem Programm mit einem `OPEN CONVERSATION`-Statement und einem `CLOSE CONVERSATION`-Statement implementiert.

Es können mehrere Verbindungen (Konversationen) gleichzeitig bestehen. Sie werden vom Client mit Hilfe von Konversationskennungen (Conversation-IDs) verwaltet, und jede von ihnen wird auf einem anderen Server ausgeführt. Remote Prozeduraufrufe, die nicht zu einer gegebenen Konversation gehören, werden innerhalb einer anderen Server-Task auf einem anderen Server ausgeführt.

Während einer Konversation können die entfernten Subprogramme auf der Serverseite einen Datenbereich, den sogenannten Kontext-Bereich, definieren und gemeinsam nutzen. Weitere Informationen finden Sie unter *Definition von Kontext-Variablen für den Natural RPC* in der Natural *Statements*-Dokumentation.

Definieren von Kontextvariablen für Natural RPC in der Dokumentation Natural Statements.

Eine Conversation kann lokal oder remote sein.

Beispiel:

```
OPEN CONVERSATION USING SUBPROGRAM 'S1''S2'
     CALLNAT 'S1' PARMS1
     CALLNAT 'S2' PARMS2
CLOSE CONVERSATION ALL
```

Beide Subprogramme (S1 und S2) müssen an der gleichen Stelle aufgerufen werden, d.h. entweder lokal oder fern (remote). Es ist nicht zulässig, lokale und ferne CALLNATs innerhalb einer Konversation zu vermischen. Wenn die Subprogramme fern ausgeführt werden, werden beide Subprogramme von derselben Server-Task ausgeführt.

So wie bei nicht-konversationellen RPC-`CALLNAT`s können Konversationen zunächst lokal geschrieben und getestet und dann an die Server übertragen werden.

Allgemeine Regeln für die lokale/ferne Ausführung von Subprogrammen

Lokale Ausführung von Subprogrammen

Wenn Sie Subprogramme lokal ausführen, gilt die folgende Regel:

- Ein Subprogramm darf kein anderes Subprogramm aufrufen, das Teilnehmer der Konversation ist.

Andere Subprogramme, die nicht in dem `OPEN CONVERSATION`-Statement aufgeführt sind, können aufgerufen werden. Sie werden jedoch im **nicht-konversationellen Modus** ausgeführt.

Entfernte Ausführung von Subprogrammen

Wenn Sie Subprogramme remote ausführen, gilt die folgende Regel:

- A subprogram `S1` may call another subprogram `S2` which is a member of the conversation.

  Ein Subprogramm S1 darf ein anderes Subprogramm `S2` aufrufen, das Teilnehmer der Konversation ist.

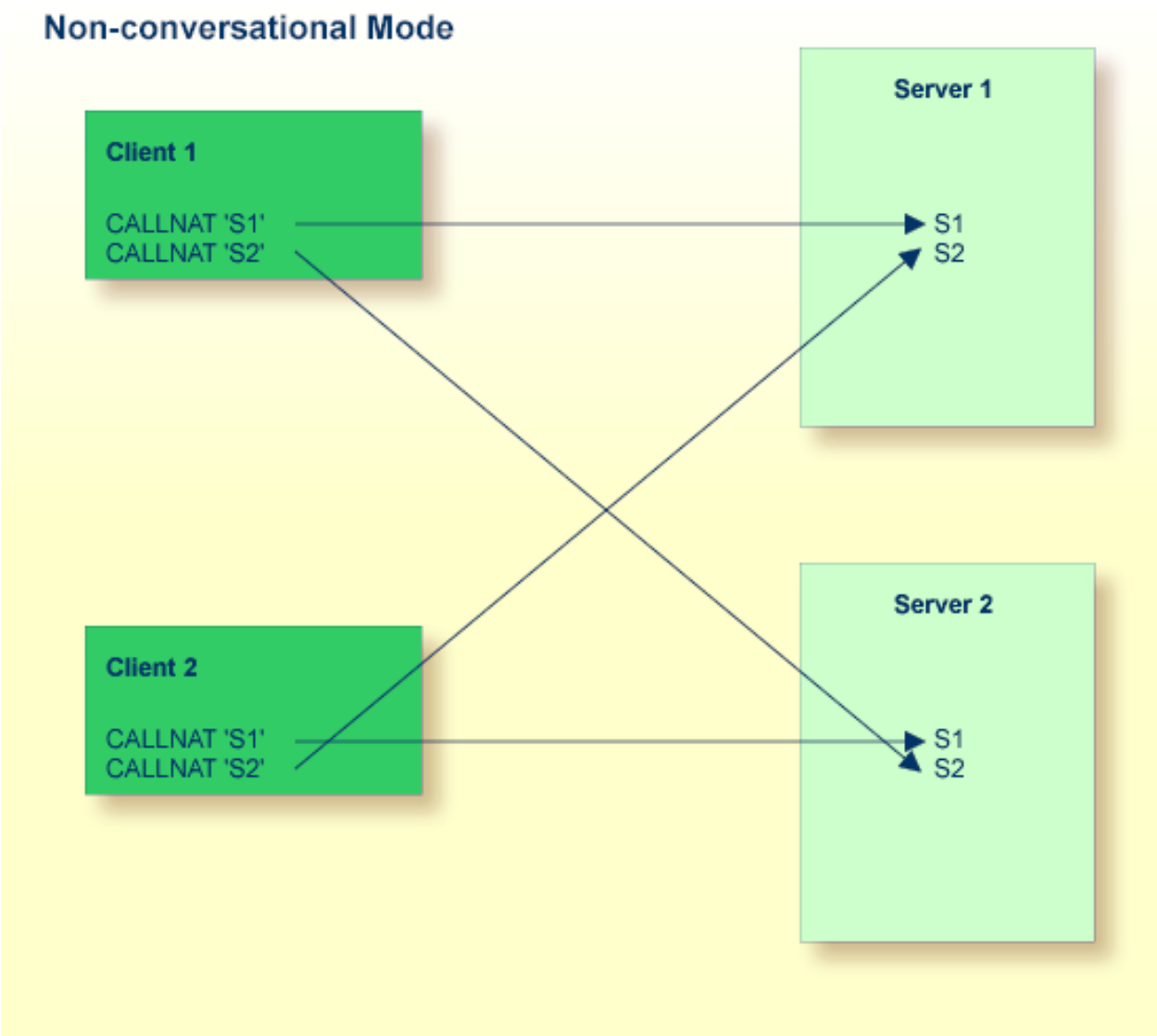
Dieses `CALLNAT` wird im nicht-konversationellen Modus ausgeführt, da es indirekt aufgerufen wurde. Das Subprogramm `S2` hat also keinen Zugriff auf den Kontextbereich.

Konversationeller versus nicht-konversationeller Modus

In einer Client-Server-Umgebung, in der mehrere Clients im nicht-konversationellen Modus auf mehrere Server zugreifen, kann es zu dem Problem kommen, dass identische `CALLNAT`-Anforderungen von verschiedenen Clients auf demselben Server ausgeführt werden.

Das bedeutet z.B., dass ein `CALLNAT 'S1'` von Client 1 das Subprogramm `S1` auf Server 1 ausführt (`S1` schreibt einen Datensatz in die Datenbank). Die Transaktion für Client 1 ist noch nicht abgeschlossen (kein `END TRANSACTION`), wenn Client 2 ebenfalls einen CALLNAT 'S1' an Server 1 sendet und damit die Daten von Client 1 überschreibt. Sendet Client 1 daraufhin ein `CALLNAT 'S2'` (d.h. `END TRANSACTION`), geht Client 1 davon aus, dass seine Daten korrekt gespeichert wurden, obwohl in Wirklichkeit die Daten des identischen `CALLNAT` von Client 2 gespeichert wurden.

Das folgende Diagramm veranschaulicht dies mit zwei Clients und zwei Servern. In einem solchen Szenario können Sie nicht kontrollieren, ob zwei identische `CALLNAT`s von zwei verschiedenen Clients auf dasselbe Subprogramm auf demselben Server zugreifen:

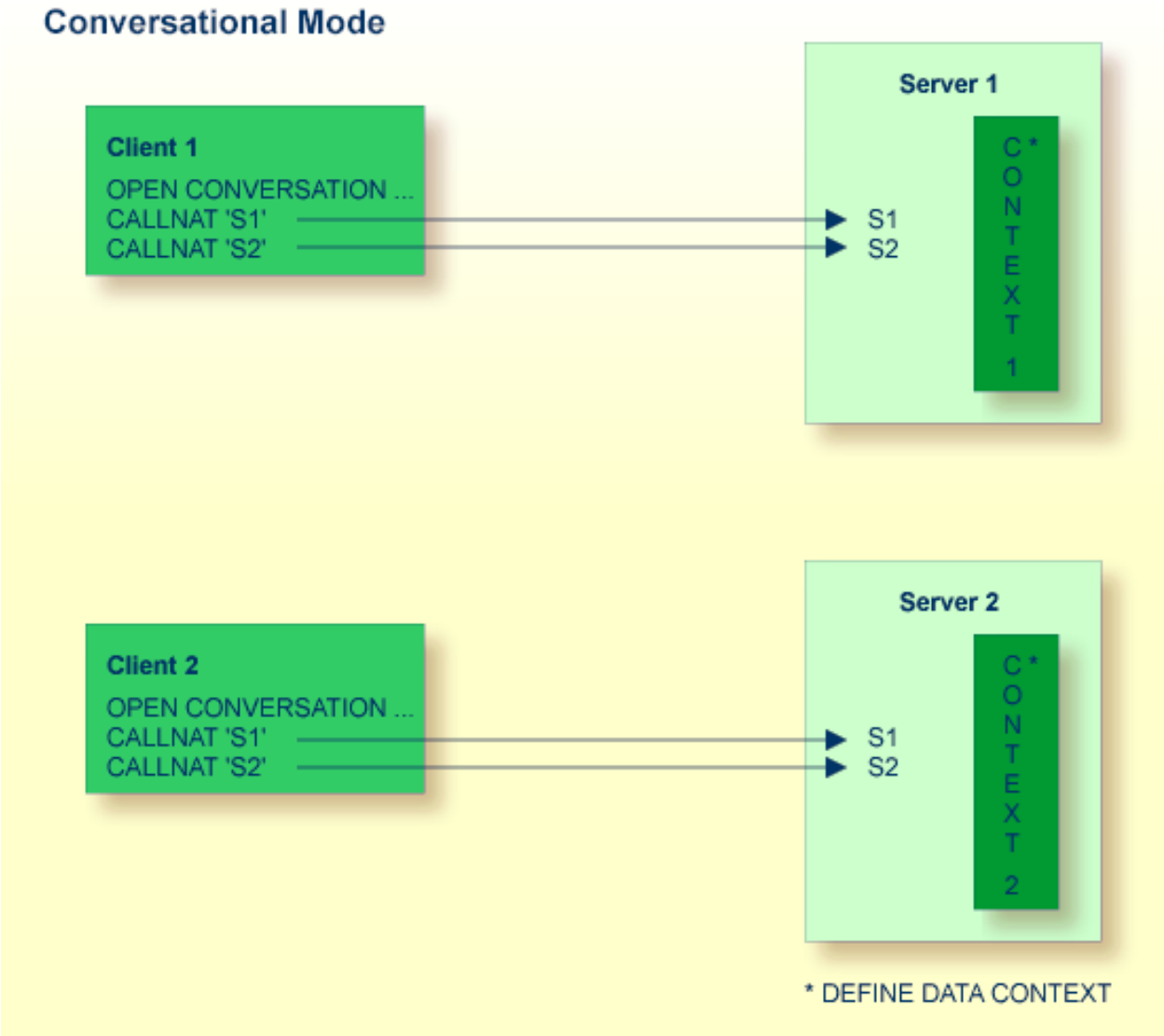


Im obigen Beispiel kann `CALLNAT 'S2'` von Client 1 auf das Subprogramm `S2` auf Server 1 und auf Server 2 zugreifen. `CALLNAT 'S2'` von Client 2 hat die gleiche Wahl.

Ähnlich könnte `CALLNAT 'S1'` von Client 1 auf das Subprogramm `S1` auf Server 1 und auf Server 2 zugreifen, während `CALLNAT 'S1'` von Client 2 die gleiche Wahl hat.

Es ist offensichtlich, dass es hier zu Überschneidungen kommen kann, wenn die Subprogramme so konzipiert sind, dass sie innerhalb eines Server-Task-Kontextes ausgeführt werden.

Sie können die potenziellen Probleme eines nicht-konversationellen RPC vermeiden, indem Sie eine komplexere RPC-Transaktion im konversationellen Modus definieren:



Dazu eröffnen Sie eine Konversation. Dazu wird auf der Client-Seite das `OPEN CONVERSATION`-Statement verwendet, das sich auf `CALLNAT 'S1'` und `CALLNAT 'S2'` bezieht. Durch das Eröffnen einer solchen Konversation wird eine ganze Server-Task (z. B. Server 1) reserviert, und keine anderen entfernten `CALLNAT`s dürfen diese Konversation auf diesem Server unterbrechen, bevor diese Konversation geschlossen wurde. Zusätzlich können Sie mit dem `DEFINE DATA CONTEXT`-Statement einen gemeinsamen Kontextbereich für die beiden Subprogramme auf der Server-Seite definieren.

Allgemeine Regeln für die Verwendung von konversationellem/nicht-konversationellem RPC

Als allgemeine Regel gilt Folgendes:

- Verwenden Sie **konversationellen RPC**, um sicherzustellen, dass eine definierte Liste von Subprogrammen ausschließlich in einem Kontext ausgeführt wird.
- Verwenden Sie **nicht-konversationellen RPC**, wenn jedes Ihrer Subprogramme innerhalb einer anderen Server-Task verwendet werden kann oder wenn sich die Transaktion nicht über mehr als einen Server-Aufruf erstreckt. Dies hat den Vorteil, dass kein Server über einen längeren Zeitraum blockiert und Sie nur eine relativ geringe Anzahl von Server-Tasks benötigen.

Möglicher Nachteil bei Verwendung von konversationellem RPC

Ein möglicher Nachteil von konversationellen RPCs ist, dass Sie eine ganze Server-Task reservieren und damit alle anderen Subprogramme auf diesem Server blockieren. Dies kann dazu führen, dass andere `CALLNAT`s warten müssen oder weitere Server-Tasks gestartet werden müssen.

Datenbank-Transaktionen

Die Datenbank-Transaktionen auf der Client- und der Server-Seite laufen unabhängig voneinander. Das heißt, ein `END TRANSACTION` oder `BACKOUT TRANSACTION`, das auf der Server-Seite ausgeführt wird, hat keinen Einfluss auf die Datenbank-Transaktion auf der Client-Seite und umgekehrt.

Am Ende jedes nicht-konversationellen `CALLNAT` und am Ende jeder Konversation wird auf der Server-Seite ein implizites `BACKOUT TRANSACTION` ausgeführt. Um die von dem/den entfernten `CALLNAT`(s) vorgenommenen Änderungen zu übernehmen, haben Sie folgende Möglichkeiten:

Nicht-konversationelles CALLNAT

1. Führen Sie ein explizites `END TRANSACTION` aus, bevor Sie den `CALLNAT` verlassen.
2. Setzen Sie den Natural-Profilparameter `ETEOP` auf `ON`. Dies führt zu einem impliziten `END TRANSACTION` am Ende eines jeden nicht-konversationellen `CALLNAT`s.

   Je nach Einstellung des Subparameters `SRVCMIT` wird das `END TRANSACTION` entweder vor dem Senden der Antwort an den Client (`SRVCMIT=B`) oder nach dem erfolgreichen Senden der Antwort an den Client (`SRVCMIT=A`) ausgeführt. `SRVCMIT=B` ist die Standardeinstellung und ist mit früheren Versionen des RPC kompatibel.

Konversationelles CALLNAT

1. Führen Sie ein explizites `END TRANSACTION` aus, bevor die Konversation durch den Client beendet wird.
2. Setzen Sie den Natural-Profilparameter `ETEOP` auf `ON`. Dies führt zu einem impliziten `END TRANSACTION` am Ende jeder Konversation.

Je nach Einstellung des Subparameters SRVCMIT wird das END TRANSACTION entweder vor dem Senden der Antwort an den Client (SRVCMIT=B) oder nach dem erfolgreichen Senden der Antwort an den Client (SRVCMIT=A) ausgeführt. SRVCMIT=B ist die Standardeinstellung und ist mit früheren Versionen des RPC kompatibel.

3. Bevor Sie das CLOSE CONVERSATION-Statement ausführen, rufen Sie die Anwendungsprogrammierschnittstelle USR2032N auf der Client-Seite auf. Dadurch wird am Ende der einzelnen Konversation ein implizites END TRANSACTION ausgelöst.

Behandlung der mit den Profilparametern LT, MAXCL, MADIO und MT gesetzten Limits auf dem Server

The following Natural limits apply to each individual remote CALLNAT execution on the Natural RPC Server:

Die folgenden Natural-Limits gelten für jede einzelne Remote-CALLNAT-Ausführung auf dem Natural RPC Server:

Natural-Profilparameter	Art des Limits
LT	Limit für Verarbeitungsschleifen
MADIO	Maximale Anzahl der DBMS-Aufrufe zwischen Bildschirm-Ein-/Ausgaben
MAXCL	Maximale Anzahl an Programmaufrufen
MT	Maximale CPU-Zeit

Diese Limits werden nach jeder Remote-CALLNAT-Ausführung zurückgesetzt, und Sie können bestimmte Limits für jede Remote-CALLNAT-Ausführung erzwingen.

Diese Maßnahme trägt dazu bei, unvorhersehbare Fehlersituationen zu vermeiden, wenn die Ausführung einer Client-Anforderung aufgrund der Verarbeitung einer früheren Client-Anforderung eines der Limits erreicht

Ort der Konversationen

Die beiden Subprogramme S1 und S2 (in der obigen Abbildung dargestellt) müssen am selben Ort aufgerufen werden, d.h. entweder lokal oder entfernt (remote). Lokale und entfernte CALLNATs dürfen innerhalb einer Konversation nicht vertauscht werden. Wenn die Subprogramme entfernt ausgeführt werden, werden beide Subprogramme von der gleichen Server-Task ausgeführt.

Natural RPC Terminology

Die folgende Tabelle enthält wichtige Schlüsselbegriffe, die in der SYSRPC Utility und in der Natural RPC-Dokumentation verwendet werden:

Begriff	Erklärung
Client-Stub	Veralteter Begriff, siehe *Client-Interface-Objekt*.
EntireX Broker Stub	Schnittstelle zwischen der Natural RPC-Laufzeit und der EntireX Broker-Transportschicht, die umgewandelten („**marshalled**“) Daten zwischen Client und Server austauscht.
Client-Interface-Objekt	Nimmt die CALLNAT-Anforderungen auf der Client-Seite entgegen, wandelt das Format der übergebenen Parameter um („**marshalling**“), überträgt die Daten durch die Natural RPC-Laufzeit und die Transportschicht zum Remote-Server, wandelt das Ergebnis zurück („**unmarshalling**“) und gibt es an den Aufrufer zurück. Der Interface-Objekt-Stub ist das lokale Subprogramm, über das das Server-Subprogramm aufgerufen wird. Das Client-Interface-Objekt hat den gleichen Namen und enthält die gleichen Parameter wie das entsprechende Server-Subprogramm.
Impersonation	Impersonation setzt voraus, dass der Zugriff auf das Betriebssystem, auf dem ein Natural RPC Server läuft, durch ein SAF-konformes externes Security-System kontrolliert wird. Die Benutzerauthentifizierung wird von diesem externen Security-System durchgeführt. Impersonation bedeutet, dass nach erfolgreicher Authentifizierung und Feststellung der Identität des Benutzers alle nachfolgenden Berechtigungsprüfungen auf der Grundlage dieser Identität durchgeführt werden. Dazu gehören auch Berechtigungsprüfungen für den Zugriff auf externe Ressourcen (z. B. Datenbanken oder Arbeitsdateien). Nach erfolgreicher Authentifizierung kann der Benutzer „seine Identität nicht ändern“, d. h. er kann keine andere Benutzerkennung verwenden. Siehe *Impersonation* in *Verwendung von Security*.
Interface-Objekt	In früheren Versionen von EntireX wurde der Begriff „Stub“ auch für anwendungsabhängige, von der Workbench generierte Codestücke zum Absetzen und Empfangen von Remote Procedure Calls verwendet. Diese Objekte werden jetzt als „Interface-Objekte“ bezeichnet.
Knotenname	Der Name des Knotens, an den der Remote-CALLNAT gesendet wird. Bei der Kommunikation über den EntireX Broker ist der Knotenname beispielsweise der Name des EntireX Brokers, wie er in der EntireX Broker-Attributdatei im Feld BROKER-ID definiert ist.
Marshalling	Beim **Marshalling** werden Schnittstellendaten in RPC-Pakete verpackt, die über Prozess- und Netzwerkgrenzen hinweg übertragen werden. Dazu gehört die Konvertierung von Daten von ihrem eigentlichen Datentyp in eine interne Darstellung. Die Umwandlung aus einer internen Darstellung in ihren eigentlichen Datentyp wird als **Unmarshalling** bezeichnet.

NATCLTGS	Der Name des Natural-Subprogramms, das mit der SYSRPC Utility generiert wurde, um das Diensteverzeichnis (Service Directory) zu implementieren (siehe unten).
RPC-Parameter	Alle Parameter, die zur Steuerung eines Natural RPC zur Verfügung stehen, sind in der Natural *Parameter-Referenz*-Dokumentation ausführlich beschrieben. Siehe Abschnitt *Profilparameter*. Die RPC-Parameter sind im Parameter-Makro NTRPC enthalten (statische Definition) oder werden mit dem Profilparameter RPC definiert (dynamische Definition). Siehe *RPC - Remote Procedure Call-Einstellungen* und *NTRPC-Macro-Syntax* (in der Natural *Parameter-Referenz*-Dokumentation).
RPC-Server	Ein RPC-Server ist entweder ein Natural-Server oder ein EntireX RPC-Server.
Service Directory	Das Diensteverzeichnis (Service Directory) enthält Informationen über die Services, d.h. Dienste (Subprogramme), die ein Server anbietet. Es kann lokal auf jedem Client-Knoten verfügbar sein oder sich auf einem Remote Directory Server befinden, auf den unter Windows, UNIX und OpenVMS durch den Profilparameter RDS und auf Großrechnern durch den entsprechenden Schlüsselwort-Unterparameter RDS des Profilparameters RPC oder des Parameter-Makros NTRPC verwiesen wird.
Servername	Der Name des Servers, auf dem der CALLNAT ausgeführt werden soll. Bei der Kommunikation über EntireX Broker ist der Servername der Name, der in der EntireX Broker-Attributdatei im Feld SERVER definiert ist.
Server-Task	Eine Natural-Task, die Services, d.h. Dienste (Subprogramme), anbietet. Dies ist typischerweise eine Batch-Task oder eine asynchrone Task. Sie wird durch einen Servernamen identifiziert.

3 Voraussetzungen und vorbereitende Informationen

Dieses Kapitel liefert einen Überblick über die allgemeinen Voraussetzungen und eine kurze Beschreibung der Funktionalitäten, die in Natural für die Implementierung einer Natural Remote Procedure Call-Umgebung (RPC-Umgebung) zur Verfügung stehen.

Beteiligte Produkte

Wenn die RPC-Umgebung auf verschiedenen Plattformen implementiert werden soll, werden die jeweils aktuellen Versionen von Natural for Mainframes, Windows, UNIX oder OpenVMS benötigt. Darüber hinaus sind die folgenden erforderlichen oder optionalen Produkte, Unterprodukte und Funktionalitäten für den Einsatz in einer Natural RPC-Umgebung verfügbar:

Produkt	Zweck
EntireX Communicator (EntireX Broker)	Der EntireX Broker des Software AG-Produkts EntireX Communicator stellt in der Regel die Middleware-Schicht zwischen dem Natural-Client und einem Natural-Server dar. Er verwendet das ACI-Protokoll und umfasst die entsprechenden Client/Server-Stubs.
Entire Net-Work	Dieses Software AG-Produkt wird benötigt, wenn die von EntireX Broker verwendete Transportmethode Entire Net-Work ist. Dies ist die bevorzugte Transportmethode.
TCP/IP	Erforderlich, wenn die vom EntireX Broker verwendete Transportmethode TCP/IP ist. Siehe auch ***TCP/IP als Transportmethode verwenden*** in *Einrichten einer Natural RPC-Umgebung*.
EntireX Developer's Kit	Dieses Kit ist im Produkt EntireX Communicator enthalten und wird für die Unterstützung von Nicht-Natural-Programmiersprachen benötigt.
Directory Services	Ein Remote Directory Server (RDS) ermöglicht es Ihnen, Directory-Definitionen an einem Ort zu definieren, so dass seine Dienste von allen Clients in einer RPC-Umgebung genutzt werden können.
Natural Security	Optional. Dieses Zusatzprodukt der Software AG wird auf der Server-Seite benötigt, um Natural RPC Server und -Dienste zu schützen, wenn die Security auf dem Client aktiv ist und auch umgekehrt.
EntireX RPC	Optional. Natural RPC unterstützt den EntireX RPC für 3GL vollständig. EntireX RPC ist im Produkt EntireX Communicator enthalten.
EntireX Security	Optional. Natural RPC unterstützt EntireX Security sowohl auf der Client- als auch auf der Server-Seite vollständig. EntireX Security ist im Produkt EntireX Broker enthalten.

Hinweise, wie Sie Informationen zu den unterstützten Software AG-Produktversionen erhalten, finden Sie unter *Verfügbare und unterstützte Software AG-Produktversionen* in der *Freigabemitteilung (Release Notes)*.

Informationen zu anderen Produkten, die in einer Natural RPC-basierten Umgebung eingesetzt werden können, finden Sie in der entsprechenden Produktdokumentation.

Verwendete Natural-Statements

Die folgenden Natural-Statements werden bei der Erstellung einer Natural RPC-Umgebung verwendet:

- `CALLNAT`
- `DEFINE DATA PARAMETER`
- `DEFINE DATA CONTEXT`
- `OPEN CONVERSATION`
- `CLOSE CONVERSATION`

Im Abschnitt ***Grenzen und Einschränkungen*** finden Sie in den Abschnitten ***Reaktionen der Natural-Statements*** und ***Hinweise zu Natural-Statements auf dem Server*** Informationen darüber, was Sie über ein abweichendes Verhalten dieser Statements wissen sollten, wenn sie in einer Natural RPC-Umgebung verwendet werden.

Natural-Utilities zur Verwendung beim Natural RPC

Die folgenden Natural-Utilities werden bei der Erstellung und Pflege einer Natural RPC-Umgebung verwendet:

- `SYSRPC`

  Diese Utility wird für die Pflege von Remote-Procedure-Call-Umgebungen verwendet.

- `SYSEXT`

  Diese Utility dient zum Auffinden und Testen von Natural-Anwendungsprogrammierschnittstellen (APIs, siehe unten), die in der aktuellen System Library `SYSEXT` enthalten sind.

- `SYSPARM`

  Auf Großrechnern dient diese Utility zur Erstellung und Pflege eines Satzes von Natural-Profilparametern, der unter einem Profilnamen gespeichert wird.

- Configuration Utility

Unter Windows, UNIX und OpenVMS wird diese Utility verwendet, um globale und lokale Konfigurationsdateien zu ändern und Parameterdateien zu erstellen oder zu ändern.

Anwendungsprogrammierschnittstellen zur Verwendung beim Natural RPC

Natural Anwendungsprogrammierschnittstellen (APIs) können verwendet werden, um Informationen abzurufen oder zu ändern oder Dienste zu nutzen, die nicht über Natural-Statements zugänglich sind. Die für die Verwendung beim Natural RPC vorgesehenen Anwendungsprogrammschnittstellen befinden sich in der Natural Library `SYSEXT`.

Weitere Informationen siehe Kapitel ***APIs zur Verwendung beim Natural RPC***.

Abbildung (Mapping) von Software AG IDL auf Natural

Dieser Abschnitt beschreibt die spezifische Abbildung (Mapping) von Software AG IDL-Datentypen, -Gruppen, -Arrays und -Strukturen auf die Programmiersprache Natural. Bitte beachten Sie auch die Anmerkungen und Hinweise zu den IDL-Datentypen, die für alle Sprachbindungen gelten und in der Software AG IDL-Datei (in der *EntireX*-Dokumentation) zu finden sind.

Abbilden der IDL-Datentypen der Software AG auf Natural-Datenformate

In der folgenden Tabelle werden die folgenden Metasymbole und informellen Begriffe für die IDL verwendet.

- Die Metasymbole `[` und `]` umschließen optionale lexikalische Einheiten.
- Der informelle Begriff *`number`* (oder in manchen Fällen *`number.number`*) ist eine Folge von numerischen Zeichen, zum Beispiel 123.

Software AG IDL	Beschreibung	Natural-Datenformat	Anmerkung
A*number*	Alphanumerisch	A*number*	
AV	Alphanumerisch, variable Länge	A DYNAMIC	
AV*number*	Alphanumerisch, variable Länge mit maximaler Länge	A DYNAMIC	
B*number*	Binär	B*number*	
BV	Binär, variable Länge	B DYNAMIC	
BV*number*	Binär, variable Länge mit maximaler Länge	B DYNAMIC	
D	Datum	D	3,5
F4	Fließkomma (klein)	F4	2
F8	Fließkomma (groß)	F8	2
I1	Ganzzahl (klein)	I1	
I2	Ganzzahl (mittel)	I2	
I4	Ganzzahl (groß)	I4	
K*number*	Kanji	A*number*	1
KV	Kanji, variable Länge	A DYNAMIC	1
KV*number*	Kanji, variable Länge mit maximaler Länge	A DYNAMIC	1
L	Logisch	L	
N*number*[.*number*]	Ungepackte Dezimalzahl	N*number*[.*number*]	
NU*number*[.*number*]	Ungepackte Dezimalzahl ohne Vorzeichen	N*number*[.*number*]	
P*number*[.*number*]	Gepackte Dezimalzahl	P*number*[.*number*]	
PU*number*[.*number*]	Gepackte Dezimalzahl ohne Vorzeichen	P*number*[.*number*]	
T	Zeit	T	3,4
U*number*	Unicode	U*number*	
UV	Unicode, variable Länge	U DYNAMIC	
UV*number*	Unicode, variable Länge mit maximaler Länge	U DYNAMIC	

Siehe auch Hinweise und Einschränkungen zu den IDL-Datentypen der Software AG (in der *EntireX*-Dokumentation), die für alle Sprachbindungen gelten.

Anmerkungen:

1. Der Datentyp K ist ein RPC-spezifisches Datenformat, das nicht Teil der Natural-Sprache ist.
2. Bei der Verwendung von Fließkomma-Datentypen kann es zu Rundungsfehlern kommen, so dass die Werte von Sendern und Empfängern leicht voneinander abweichen können. Dies gilt insbesondere, wenn Client und Server unterschiedliche Darstellungen für Fließkommadaten verwenden (IEEE, HFP).
3. Anzahl der Tage AD (anno domini, nach Christi Geburt). Der gültige Bereich reicht von 1.1.0001 bis zum 28.11.2737. Die Zuordnung (Mapping) der Zahl zum Datum im gesamten Bereich ab

1.1.0001 erfolgt nach dem Julianischen und Gregorianischen Kalender unter Berücksichtigung der folgenden Regeln:

a. Jahre, die geradzahlig durch 4 teilbar sind, sind Schaltjahre.

b. Jahre, die geradzahlig durch 100 teilbar sind, sind keine Schaltjahre, es sei denn, Regel 3 (siehe unten) ist zutreffend.

c. Jahre, die geradzahlig durch 400 teilbar sind, sind Schaltjahre.

d. Vor dem Jahr 1582 n. Chr. wird die Regel 1 des Julianischen Kalenders angewendet. Nach dem Jahr 1582 n. Chr. werden die Regeln 1, 2 und 3 des gregorianischen Kalenders angewandt.

In der folgenden Tabelle finden Sie die Beziehung zwischen der gepackten Zahl und einem realen Datum:

Datum / Datumsbereich	Wert / Wertebereich
1.1.0000	0 (spezieller Wert - kein Datum)
nicht definierte Daten	1 - 364 (nicht verwenden)
1.1.0001	365
1.1.1970	719527 (Beginn der C-Zeitfunktionen)
28.11.2737	999999 (maximales Datum)

4. Zählung von Zehntelsekunden AD (anno domini, nach Christi Geburt). Der gültige Bereich reicht von 1.1.0001 00:00:00.0 bis 16.11.3168 9:46:39 plus 0,9 Sekunden. In der folgenden Tabelle finden Sie das Verhältnis zwischen der gepackten Zahl und einer realen Zeit:

Zeit / Zeitbereich	Wert / Wertebereich
1.1.0000 00:00:00.0	0 (spezieller Wert - kein Datum)
nicht definierte Zeiten	1 - 315359999
1.1.0001 00:00:00.0	315360000
1.1.1970 00:00:00.0	621671328000 (Beginn der C-Zeitfunktionen)

5. Die Beziehung zwischen der gepackten Zahl eines Datums- und Zeit-Datentyps ist wie folgt:

   Zehntel einer Sekunde pro Tag = 24*60*60*10 = 864000

Zahl der Zeit	= Zahl des Datums	* 864000
315360000	= 365	* 864000 1.1.0001 00:00:00.0
621671328000	= 719527	* 864000 1.1.1970 00:00:00.0
Zahl des Datums	= Zahl der Zeit	/ 864000
365	= 315360000	/ 864000 1.1.0001
719527	= 621671328000	/ 864000 1.1.1970

Abbilden von Bibliotheksname und Alias

Der Bibliotheksname, wie er in der IDL-Datei angegeben ist, wird von Natural nicht unterstützt. Standardmäßig sendet ein Natural-Client den Library-Namen `SYSTEM` an den Server. Um einen anderen Library-Namen als `SYSTEM` von einem Client an einen Server zu senden, sind für den Client die folgenden Schritte erforderlich:

Um einen anderen Library-Namen als `SYSTEM` von einem Client an einen Server zu senden:

1. Aktivieren Sie auf dem Client die Logon-Option.
2. Rufen Sie die Anwendungsprogrammierschnittstelle `USR4008N` auf, um den Namen der Library anzugeben, andernfalls wird der Name der aktuellen Library gesendet.

Die Länge des Library-Namens ist auf 8 Zeichen begrenzt. Die Länge des Library-Namens ist auf 8 Zeichen begrenzt.

Abbilden von Programmname und Alias

Der Programmname wird von einem Client an den Server gesendet. Sonderzeichen werden nicht ersetzt. Der Programm-Aliasname wird nicht an den Server gesendet.

Das generierte Natural-Interface-Objekt hat denselben Namen.

Im RPC-Server wird der gesendete IDL-Programmname zum Auffinden des Natural-Subprogramms verwendet.

Die Länge des Programmnamens ist auf 8 Zeichen begrenzt.

Abbilden von Parameternamen

Die Parameternamen, die in der Parameter-Daten-Definition der IDL-Datei angegeben sind, werden im generierten Natural-Interface-Objekt durch künstliche Namen ersetzt.

Siehe *parameter-data-definition* im Abschnitt *Software AG IDL Grammar* in der *EntireX*-Dokumentation.

Abbilden von Fixed und Unbounded Arrays

- Feste Arrays innerhalb der IDL-Datei werden auf feste Natural Fixed Arrays abgebildet. Die untere Grenze wird auf 1 gesetzt und die obere Grenze wird auf die in der IDL-Datei angegebene obere Grenze gesetzt.

  Informationen über die Syntax zur Beschreibung Fixed Arrays in der IDL-Datei siehe *array-definition* und *fixed-bound-array-index* (im Abschnitt *Software AG IDL Grammar* in der *EntireX*-Dokumentation).
- Unbounded Arrays innerhalb der IDL-Datei werden auf Natural X-Arrays abgebildet. Die untere Grenze ist immer fest und auf 1 gesetzt.

  Siehe die *array-definition* (im Abschnitt Software AG IDL Grammatik in der EntireX-Dokumentation) für die Syntax von nichtbegrenzten Arrays innerhalb der IDL-Datei und siehe *unbounded-array-index*.

  Anmerkung: Natural Variable Arrays (Natural Notation `(../1:V)`) können auf der Natural RPC Server-Seite anstelle von Natural Fixed Arrays oder X-Arrays verwendet werden. Ein RPC-Client kann entweder ein IDL Fixed Array oder ein IDL Unbounded Array an einen Natural RPC Server mit einem solchen Natural Variable Array übergeben. Im RPC-Server kann das Variable Array nicht in der Größe verändert werden. Das bedeutet, dass die Anzahl der Array-Ereignisse nicht verändert werden kann und der Natural RPC Server immer die gleiche Anzahl an Ausprägungen zurückgibt.

Abbilden von Gruppen und periodischen Gruppen

Gruppen in der IDL-Datei werden auf Natural-Gruppen abgebildet. Die Syntax für die Beschreibung von Gruppen in der IDL-Datei finden Sie unter *group-parameter-definition* (im Abschnitt *Software AG IDL Grammar* in der *EntireX*-Dokumentation).

Abbilden von Strukturen

Strukturen innerhalb der IDL-Datei werden auf Natural-Gruppen abgebildet. Die Syntax für die Beschreibung von Strukturen in der IDL-Datei finden Sie unter *structure-definition* (im Abschnitt *Software AG IDL Grammar* in der *EntireX*-Dokumentation).

Abbilden der Richtungsattribute IN, OUT, INOUT

Die IDL-Syntax erlaubt es, Parameter als IN-Parameter, OUT-Parameter oder INOUT-Parameter zu definieren (INOUT ist der Standard, wenn nichts angegeben wird). Diese Richtungsangabe wird von Natural wie folgt wiedergegeben:

- Parameter mit dem `OUT`-Attribut werden vom RPC-Client an den RPC-Server gesendet. Sie werden immer mit der Call-by-Reference-Methode bereitgestellt.
- Parameter mit dem `IN`-Attribut werden vom RPC-Server an den RPC-Client gesendet. Sie werden immer mit der Call-by-Reference-Methode bereitgestellt.
- Parameter mit dem `INOUT`-Attribut werden vom RPC-Client an den RPC-Server und dann zurück an den RPC-Client gesendet.
- Nur die Richtungsinformation der Top-Level-Felder (Level 1) ist relevant. Gruppenfelder erben immer die Spezifikation von ihrem übergeordneten Feld. Eine andere Spezifikation wird ignoriert.

Siehe *attribute-list* (im Abschnitt *Software AG IDL Grammar* in der *EntireX*-Dokumentation) für die Syntax zur Beschreibung von Attributen innerhalb der IDL-Datei und siehe *direction-attribute*.

Anmerkung: Wenn Sie in der Natural-Utility `SYSRPC` ein Interface-Objekt-Layout definieren, ist die Bedeutung der Richtungsattribute `IN` und `OUT` gegenüber der IDL vertauscht:

- `IN` in `SYSTRPC` ist `OUT` in IDL
- `OUT` in `SYSTRPC` ist `IN` in IDL

Abbilden des ALIGNED-Attributs

Das `ALIGNED`-Attribut ist für die Programmiersprache Natural nicht relevant. Ein Natural-Client kann jedoch das `ALIGNED`-Attribut an einen RPC-Server senden, wo es eventuell benötigt wird. Dazu benötigen Sie ein Natural-Interface-Objekt, das aus einer IDL-Datei generiert wurde.

Siehe *attribute-list* (im Abschnitt *Software AG IDL Grammar* in der *EntireX*-Dokumentation) für die Syntax der Attribute in der IDL-Datei und siehe *aligned-attribute*.

Aufrufen von Servern als Prozeduren oder Functions

Die IDL-Syntax erlaubt nur die Definition von Prozeduren. Sie kennt nicht das Konzept der Function. Eine Function ist eine Prozedur, die neben den Parametern auch einen Wert zurückgibt. Prozeduren und Functions sind zwischen Client und Server transparent. Das bedeutet, dass ein Client, der eine Function verwendet, einen Server aufrufen kann, der als Prozedur implementiert ist, und umgekehrt.

Client- und Server-Seite

Der Natural RPC unterstützt keine Functions.

4 Grenzen und Einschränkungen

Dieses Kapitel informiert Sie über einige Grenzen und Einschränkungen, die Sie bei der Verwendung des Natural RPC (Remote Procedure Call) beachten sollten.

Übertragung von Benutzerkontext

Mit Ausnahme der Benutzerkennung wird kein Benutzerkontext an die Server-Sitzung übertragen, z. B:

- Alle Parameter der Client-Sitzung bleiben unverändert und haben keinen Einfluss auf die Ausführung auf der Server-Seite.
- Offene Transaktionen auf der Client-Seite können nicht vom Server geschlossen werden und umgekehrt.
- Die Behandlung von Client-Reports und die Verarbeitung von Arbeitsdateien können nicht auf der Serverseite fortgesetzt werden und umgekehrt.
- Die Behandlung des Natural-Stacks kann ebenfalls nicht fortgesetzt werden.

Übertragung von Systemvariablen

Außer `*USER` können keine Systemvariablen von der Client- zur Server-Seite übertragen werden.

Anwendungsunabhängige Variablen

In einem RPC-Server werden anwendungsunabhängige Variablen (Application Independent Variables, AIVs) nicht implizit freigegeben, sondern bleiben über RPC-Anforderungen hinweg aktiv, da verschiedene Clients Zugriff auf dieselben Variablen des RPC-Servers haben können. Das heißt, sie müssen explizit mit dem Statement `RELEASE VARIABLES` freigegeben werden.

Wenn Sie AIVs am Ende einer RPC-Anforderung freigeben wollen, können Sie dazu den Natural RPC User-Exit `NATRPC03` verwenden.

Behandlung von Parametern in Fehlersituationen

Die Behandlung von Parametern in Fehlersituationen ist unterschiedlich:

- Tritt ein Fehler bei der lokalen Ausführung auf, sind alle bisher durchgeführten Parameteränderungen wirksam, da die Parameter per „Call by Reference“ übergeben werden.
- Tritt hingegen ein Fehler bei der Remote-Ausführung auf, bleiben alle Parameter unverändert.

Variable Arrays in Subprogrammen

Wenn der Parameterdatenbereich (PDA) des Subprogramms eine variable Anzahl von Ausprägungen enthält (`1:V`-Notation), sollten Sie dieses Subprogramm nicht über ein Interface-Objekt aufrufen. Da ein Interface-Objekt nur Array-Definitionen mit einer festen Anzahl von Ausprägungen unterstützt, können Sie die Anzahl der Ausprägungen nicht von Aufruf zu Aufruf variieren.

Wenn Sie ein Interface-Objekt verwenden müssen, weil Sie z. B. einen EntireX RPC-Server mit demselben Programm aufrufen wollen, sollten Sie ein X-Array auf der Natural Client-Seite verwenden. Mit X-Arrays ist es möglich, die Anzahl der Ausprägungen von Aufruf zu Aufruf zu variieren, auch wenn ein Interface-Objekt verwendet wird. In diesem Fall wird das X-Array auf der Client-Seite an das (feste) variable Array auf der Server-Seite übergeben. Das variable Array ist fest, weil das Server-Programm von Aufruf zu Aufruf eine unterschiedliche Anzahl von Ausprägungen erhalten kann, aber die Anzahl der Ausprägungen nicht ändern kann.

X-Arrays

X-Arrays werden in der Parameterliste einer entfernten `CALLNAT`-Statement-Ausführung unterstützt. Der Server kann die Anzahl der Ausprägungen erhöhen oder verringern.

Einschränkungen

- Im Falle eines mehrdimensionalen Arrays müssen alle Dimensionen des Arrays erweiterbar sein.
- Die untere Grenze (Lower Bound) darf nicht erweiterbar sein, d.h. es sind nur erweiterbare obere Grenzen (Upper Bounds) erlaubt.
- Wenn Sie ein X-Gruppen-Array verwenden möchten, das ein Array mit konstanten Grenzen enthält, oder ein Gruppen-Array, das ein X-Array enthält, müssen Sie ein Interface-Objekt verwenden. Bei der Generierung des Interface-Objekts müssen Sie die Gruppenstruktur auf dem Bildschirm **Interface Object Generation** definieren.

Beispiele:

```
01 X-Group-Array (/1:*)
02 Array (A10/1:10)
*
01 Group-Array (/1:10)
02 X-Array (A10/1:*)
```

Gruppen und Interface-Objekte

Wenn Gruppen-Arrays oder X-Gruppen-Arrays in der Parameterliste einer entfernten `CALLNAT`-Statement-Ausführung vorhanden sind und ein Interface-Objekt verwendet wird, gelten die folgenden Einschränkungen.

Einschränkungen

- Sie dürfen die Session-Parametereinstellungen `AD=O` oder `AD=A` (Attributdefinition) nicht im `CALLNAT`-Statement nicht verwenden.
- Gruppen-Arrays und X-Gruppen-Arrays, die von einem Client-Natural-Objekt an ein Interface-Objekt übergeben werden, müssen zusammenhängend sein. Es wird daher dringend empfohlen, immer ein vollständiges Array an das Interface-Objekt zu übergeben, indem für alle Dimensionen die Stern-Notation (*) verwendet wird. Außerdem wird dringend empfohlen, im Client-Natural-Programm, im Interface-Objekt und im Server-Programm identische Datendefinitionen zu verwenden.

Gruppen-Arrays auf der RPC Server-Seite

Das Speicherlayout von Gruppen-Arrays im `DEFINE DATA PARAMETER`-Bereich von Subprogrammen auf der RPC Server-Seite ist hinsichtlich der Syntax nicht unbedingt identisch.

Redefinieren Sie keine Felder innerhalb eines Gruppen-Arrays und übergeben Sie das Gruppen-Array nicht an ein 3GL-Programm. Wenn Sie dies tun müssen, kopieren Sie das Gruppen-Array in ein Gruppen-Array mit dem gleichen Layout im Bereich `DEFINE DATA LOCAL` und verwenden Sie dieses lokale Gruppen-Array im Aufruf des 3GL-Programms.

Nicht unterstützte Natural-Datenformate

Die Natural-Datenformate C (Attribute Control) und Handle sind in der Parameterliste einer Remote CALLNAT-Statement-Ausführung nicht erlaubt.

EntireX RPC Server

Wenn Sie einen EntireX RPC-Server mit einem entfernten `CALLNAT`-Statement aufrufen möchten, wird dringend empfohlen, ein Interface-Objekt zu verwenden. Es gibt zwei Möglichkeiten, ein solches Interface-Objekt zu erzeugen:

- Generieren Sie das Interface-Objekt aus der EntireX IDL-Datei (IDL = Interface Definition Language). Dies wird nur von Natural Studio (nur Windows) oder der EntireX Workbench (nur Linux und Windows) unterstützt.
- Definieren Sie die Parameter der IDL-Definition des Subprogramms, das Sie auf einem EntireX RPC-Server aufrufen möchten, manuell im Bildschirm **Interface Object Generation** der Utility `SYSRPC`. Wenn die IDL (Interface Definition Language) eine Gruppenstruktur enthält, müssen Sie die gleiche Gruppenstruktur auf dem Bildschirm **Interface Object Generation** definieren.

VSAM verwenden

Wenn Sie auf einen VSAM-Datensatz in einer Subtasking-Umgebung zugreifen oder wenn Sie einen VSAM-Datensatz regionsübergreifend gemeinsam nutzen, müssen Sie die erforderlichen Freigabeoptionen berücksichtigen. Beispielsweise müssen Sie möglicherweise regionsübergreifende `SHAREOPTIONS 4` anstelle von `SHAREOPTIONS 2` festlegen, um eine Pufferinvalidierung zu erzwingen, wenn Datensätze in einem Adressraum in einem VSAM-Datensatz eingefügt werden und derselbe VSAM-Datensatz in einem anderen Adressraum gelesen wird. Andernfalls werden die erst kürzlich eingefügten Datensätze möglicherweise nicht gefunden.

Sie sollten auch die Verwendung von Record Level Sharing (RLS) in Betracht ziehen.

Weitere Informationen finden Sie in der entsprechenden VSAM-Dokumentation von IBM.

Reaktionen der Natural-Statements

Einige Natural Statements können unterschiedlich reagieren, wenn sie in einem Natural RPC-Kontext verwendet werden, zum Beispiel:

Statement	Beschreibung
`OPEN CONVERSATION` `CLOSE CONVERSATION`	Wenn dieses Statement auf einem Server ausgeführt wird, hat es keinen Einfluss auf die Client-Sitzung. Wenn der Server selbst als Client für einen anderen Server (als Agent) fungiert, wirken sich diese Statements nur auf die Konversationen auf dem zweiten Server aus.
`PASSW`	Die Passwort-Einstellung bleibt nur auf der Server-Seite aktiv, auch bei späteren Ausführungen durch andere Benutzer.

`SET CONTROL`
`SET GLOBALS`
`SET KEY`
`SET TIME`
`SET WINDOW`
	Es werden keine Einstellungen an den Aufrufer zurückgegeben.

`STACK`	Alle Stack-Daten werden nach der Ausführung freigegeben.
`STOP` `TERMINATE`	Mit diesen Statements wird die Client-Sitzung *nicht* beendet. Informationen zum Beenden eines Natural RPC Servers finden Sie unter ***Beenden eines Natural RPC Servers***.

Hinweise zu Natural-Statements auf dem Server

Die Verwendung der folgenden Statements bei einem Natural RPC ist theoretisch möglich, wird aber nicht empfohlen, da sie unerwünschte Auswirkungen haben:

Statement	Beschreibung
`TERMINATE`	Die Verwendung dieses Statements führt zur Beendigung des Servers, unabhängig von eventuell noch offenen Konversationen.

`FETCH`
`RUN`
`STOP`
	Die Verwendung dieser Statements führt dazu, dass der `CALLNAT`-Kontext verloren geht.
	
	Bei einem `FETCH`-, `RUN`- oder `STOP`-Statement stellt der Server fest, dass er seinen `CALLNAT`-Kontext verloren hat und gibt eine entsprechende Natural-Fehlermeldung an den Client zurück. Zu diesem Zeitpunkt ist das Statement jedoch schon vom Server ausgeführt worden.
	
	Ausnahme: Dies gilt nicht bei `FETCH RETURN`.

`INPUT`	Eingabewerte sind unvorhersehbar, wenn die Eingabedaten aus einer Datei (und nicht vom Stack) gelesen werden.

5 Einrichten einer Natural RPC-Umgebung

Um eine Natural RPC-Umgebung einzurichten und in Betrieb zu nehmen, müssen Sie die unten beschriebenen Schritte für alle Client-Naturals und Server-Naturals durchführen und die plattformspezifischen Hinweise und Aspekte lesen.

Einen Natural-Client einrichten

Wenn nicht anders vermerkt, gilt diese Anleitung für alle Umgebungen.

Um eine Natural RPC-Umgebung einzurichten, müssen Sie die folgenden Schritte ausführen:

- Den Namen des Servers definieren
- Client-Interface-Objekt generieren
- RPC-Client-spezifische Natural-Parameter setzen

Den Namen des Servers definieren

Benutzen Sie die Funktion **Service Directory Maintenance** der `SYSRPC` Utility, um den Namen des Servers zu definieren, der für jedes remote auszuführende `CALLNAT`-Statement verwendet werden soll.

Einzelheiten und Beispiel-Bildschirme finden Sie unter *Funktion Service Directory Maintenance aufrufen* (in der *Debugger und Dienstprogramme*-Dokumentation).

Das generierte Directory-Subprogramm `NATCLTGS` muss der Natural-Client-Anwendung zur Verfügung gestellt werden. Wenn Sie `NATCLTGS` nicht in Ihrer Client Library generiert haben, verschieben Sie `NATCLTGS` in diese Library oder in eine der Steplibs.

Wahlweise können Sie eine der folgenden Möglichkeiten zur Serverauswahl verwenden:

- **Einen Standard-Server adressieren**
  Siehe ***Festlegen einer Standard-Serveradresse innerhalb einer Natural-Sitzung*** oder Schlüsselwort-Subparameter `DFS` des Profilparameters `RPC` oder des Parameter-Makros `NTRPC`.
- **Einen Remote Directory Server verwenden**
  Siehe **Verwendung eines Remote Directory Servers** oder Schlüsselwort-Subparameter `RDS` des Profilparameters `RPC` oder des Parameter-Makros `NTRPC`.

Client-Interface-Objekt generieren

Dieser Schritt gilt nur, wenn Sie nicht mit automatischer Natural RPC-Ausführung arbeiten wollen oder können (siehe *Betrieb einer Natural RPC-Umgebung* , ***Mit automatischer Natural RPC-Ausführung arbeiten***).

Verwenden Sie für jedes remote auszuführende `CALLNAT`-Statement die Funktion **Interface Object Generation** der `SYSRPC` Utility (siehe ***Interface-Objekte erstellen***).

Beachten Sie, dass das generierte Interface-Objekt der Natural-Client-Umgebung zur Verfügung gestellt werden muss. Wenn Sie das **Interface-Objekt** nicht in Ihrer Client-Library generieren, verschieben Sie das Interface-Objekt in diese Library oder in eine der Steplib Libraries.

RPC-Client-spezifische Natural-Parameter setzen

Setzen Sie die Schlüsselwort-Subparameter des Profilparameters `RPC` oder des Parameter-Makros `NTRPC`, die für die Client-spezifische Behandlung von Remote Procedure Calls relevant sind.

Zwingend erforderliche Parameter:

Schlüsselwort-Subparameter	Funktion
`MAXBUFF`	Maximale Puffergröße (nur bei automatischer RPC-Ausführung).
`RPCSIZE`	Größe des von Natural RPC verwendeten Puffers (nur für Mainframe-Clients).

Optionale Parameter:

Schlüsselwort-Subparameter	Funktion
`ACIVERS`	Dieser Parameter ist obsolet und wird ignoriert. Die höchste ACI-Version wird durch den RPC-Nukleus und EntireX ausgehandelt.
`AUTORPC`	Automatische Natural RPC-Ausführung.
`COMPR`	Einstellen der RPC-Pufferkomprimierung. Siehe auch *Betrieb einer Natural RPC-Umgebung*, ***Komprimierung verwenden***.
`CPRPC`	Codepage-Namen definieren.
`DFS`	Standard-Serveradresse des RPC-Clients festlegen.
`RDS`	Remote Directory Server definieren.
`RPCSDIR`	Geben Sie den Namen der Natural Library an, in der sich das Service Directory befindet (nur für Großrechner-, UNIX- und OpenVMS-Server).
`TIMEOUT`	Wartezeit auf Antwort des RPC-Servers.
`TRYALT`	Alternative Serveradresse versuchen.

Die folgenden Hinweise gelten für die Verwendung des EntireX-Brokers.

Anmerkungen:

1. Die mit dem Schlüsselwort-Subparameter `DFS` angegebenen Namen müssen einen aktiven EntireX Broker identifizieren und einer Serverdefinition in der EntireX Broker-Attributdatei entsprechen (siehe *Einrichten einer EntireX Broker-Umgebung*).
2. Die mit dem Schlüsselwort-Subparameter `TIMEOUT` angegebene Wartezeit wird verwendet, um das Feld `WAIT` des EntireX Broker ACI zu setzen. Wenn `TIMEOUT` auf Null gesetzt wird, wird `WAIT=YES` gesetzt und der Client wartet auf die `CLIENT-NONACT`-Zeit. Ist die Wartezeit abgelaufen, wird der Remote Procedure Call mit einer entsprechenden Fehlermeldung beendet. Die Verwendung von `TIMEOUT` ermöglicht es Ihnen, die Vorteile des Transport-Timeout-Mechanismus zu nutzen, der von den EntireX Broker-Stubs bereitgestellt wird.

Einen Natural-Server einrichten

Ein Natural-Server ist eine Natural-Task (Server-Task), die Natural-Subprogramme (Services, d.h. Dienste) ausführen kann. Diese Natural-Task ist in der Regel eine asynchrone oder Hintergrund-Task (detached process). Der EntireX-Broker und der Client identifizieren ihn über einen Knotennamen (*nodename*) und einen Servernamen (*servername*).

Um einen Natural-Server einzurichten, führen Sie die im Folgenden beschriebenen Schritte aus:

- RPC-Server-spezifische Natural-Parameter setzen
- Verwendung des Kommando-Modus in der Server-Sitzung sicherstellen
- Eindeutige Verwendung der Adabas ETID sicherstellen
- Eine Natural-Server starten

RPC-Server-spezifische Natural-Parameter setzen

Setzen Sie die plattformabhängigen Natural-Parameter, die für die allgemeine und serverspezifische Behandlung von Remote Procedure Calls für das Server-Natural relevant sind.

Für Großrechner-Server:

1. Erstellen Sie ein RPC-spezifisches Natural-Parametermodul.
2. Setzen Sie die Schlüsselwort-Subparameter des Profilparameters `RPC` oder des Parameter-Makros `NTRPC` (siehe Tabelle unten) wie gewünscht.

Für Windows-, UNIX- oder OpenVMS-Server:

1. Erstellen Sie eine RPC-spezifische Natural-Parameterdatei.
2. Setzen Sie die Natural-Profilparameter (siehe Tabelle unten) wie gewünscht.

Zwingend erforderliche Parameter:

Schlüsselwort-Subparameter	Funktion
MAXBUFF	Maximale Puffergröße.
RPCSIZE	Größe des von Natural RPC verwendeten Puffers (nur für Großrechner-Server)
SERVER	Natural-Sitzung als RPC-Server-Sitzung starten.
SRVNAME	Name des RPC-Servers, siehe *Hinweis für EntireX Broker* unten.
SRVNODE	Name des Knotens, siehe *Hinweis für EntireX Broker* unten.

Optionale Parameter:

Schlüsselwort-Subparameter	Funktion
ACIVERS	Dieser Parameter ist obsolet und wird ignoriert. Die höchste ACI-Version wird durch den RPC-Nukleus und EntireX ausgehandelt.
CPRPC	Name der Codepage festlegen.
LOGONRQ	Anmeldung für RPC-Server-Anforderung erforderlich.
NTASKS	Mindest- und Höchstwert der Anzahl der Server-Replikate (nur bei Großrechner-Servern).
SRVCMIT	Zeit, zu der ein Natural RPC Server eine RPC-Konversation oder eine nicht konversationelle RPC-Anfrage automatisch festschreibt (Commit).
SRVRTRY	Anzahl der Versuche eines RPC-Servers, sich mit einem nicht aktiven EntireX-Broker zu verbinden/neu zu verbinden (REGISTER), und die Wartezeit zwischen zwei aufeinanderfolgenden Versuchen.
SRVTERM	Server-Beendigungsereignis.
SRVUSER	Benutzerkennung für die RPC-Server-Registrierung.
SRVWAIT	Wartezeit des RPC-Servers auf eine Client-Anforderung.
TRACE	Definition der zu verfolgenden Komponenten.
TRANSP	Server-Transportprotokoll (nicht mehr erforderlich).

Die folgenden Hinweise gelten für die Verwendung des EntireX-Brokers.

Anmerkungen:

1. Der mit dem Schlüsselwort-Subparameter SRVNODE angegebene Name muss einen aktiven EntireX Broker identifizieren und der mit dem Schlüsselwort-Subparameter SRVNAME angegebene Name muss einer Serverdefinition in der EntireX Broker-Attributdatei entsprechen, siehe *Einrichten einer EntireX Broker-Umgebung*.
2. Die mit dem Schlüsselwort-Subparameter SRVWAIT angegebene Wartezeit wird verwendet, um das Feld WAIT des EntireX Broker ACI zu setzen. Wenn SRVWAIT nicht angegeben oder auf Null gesetzt ist, wird WAIT=YES gesetzt, und der Server wartet auf die SERVER-NONACT-Zeit. Ist die Wartezeit verstrichen, wird eine entsprechende Meldung in die RPC-Server-Trace-Datei

geschrieben, und der RPC-Server wartet weiter auf die nächste Client-Anforderung. Die Verwendung des SRVWAIT-Parameters ermöglicht es Ihnen, den Transport-Timeout-Mechanismus zu nutzen, der von den EntireX Broker-Stubs bereitgestellt wird.

3. Wenn Sie die Anzahl der Versuche eines RPC-Servers, eine Verbindung zu einem EntireX Broker herzustellen oder wiederherzustellen, auf einen Wert größer als Null gesetzt haben, wird der RPC-Server nicht mehr sofort beendet, wenn der EntireX Broker heruntergefahren wird (z. B. für ein IPL oder einen längeren Wartungszeitraum). In diesem Fall müssen Sie den RPC-Server explizit beenden, bevor Sie den EntireX Broker auf eine der folgenden Arten herunterfahren: Verwenden Sie das Kommando **Terminate Service** (TS) der Funktion **Server Command Execution** der SYSRPC Utility, oder verwenden Sie die Anwendungsprogrammierschnittstelle USR2075N oder USR8208N, oder verwenden Sie den **System Management Hub for EntireX**.

Verwendung des Kommando-Modus in der Server-Sitzung sicherstellen

Um sicherzustellen, dass Ihre Natural-Server-Sitzung in den Kommando-Modus wechselt:

- Deaktivieren Sie den Natural-Menü-Modus, indem Sie den Natural-Profilparameter MENU auf =OFF setzen (gilt nur bei Großrechner-Servern).

Vermeiden Sie, ...

- ein Programm auf den Natural-Stack zu legen, das nie beendet wird.
- ein STARTUP-Programm zu verwenden, das nie beendet wird.
- den NEXT-Modus in Natural Security für Ihre Server Library zu deaktivieren.

Eindeutige Verwendung der Adabas ETID sicherstellen

Stellen Sie sicher, dass die von der Natural-Server-Sitzung verwendete Adabas ETID innerhalb eines bestimmten Adabas-Nukleus singulär ist.

Eine Natural-Server starten

Um einen Natural Server zu starten, gehen Sie wie im Abschnitt *Starten eines Natural RPC Servers* beschrieben vor.

Dieser Server wartet dann auf remote CALLNAT-Anforderungen von einem Client.

Anmerkung zu Natural im Batch-Modus auf z/OS oder auf z/VSE:

Informationen zu Servern, die den Schlüsselwort-Subparameter NTASKS des Profilparameters RPC oder des Parameter-Makros NTRPC verwenden, finden Sie unter *Besondere Aspekte bei Großrechner-Natural RPC Servern mit Replicaten*.

Einrichten eines EntireX Broker-Zugangs

Um eine EntireX Broker-Schnittstelle einzurichten, führen Sie die unten beschriebenen Schritte durch:

- Zugriff auf den EntireX Broker Stub einrichten
- API-Version
- TCP/IP als Transportmethode verwenden

Zugriff auf den EntireX Broker Stub einrichten

Ermöglichen Sie für Ihre Natural-Umgebung den Zugang zum EntireX Broker Stub. Dieser Schritt hängt von der verwendeten Plattform ab.

- Zugang zum EntireX Broker Stub auf dem Großrechner bereitstellen
- Zugriff auf den EntireX Broker Stub auf UNIX und OpenVMS bereitstellen
- Zugriff auf den EntireX Broker-Stub unter Windows bereitstellen

Zugang zum EntireX Broker Stub auf dem Großrechner bereitstellen

Verlinken Sie den EntireX Broker Stub `NATETB23` mit Ihrem Natural oder geben Sie den Profilparameter `RCA=BROKER` an, damit `NATETB23` dynamisch zur Laufzeit geladen wird.

In den folgenden Fällen kann `NATETB23` nicht verwendet werden und Sie müssen einen anderen EntireX Broker-Stub verwenden:

- Auf z/VSE müssen Sie stattdessen `BKIMB` oder `BKIMC` verwenden.
- Wenn Sie das TCP/IP-Protokoll unter BS2000 verwenden wollen, müssen Sie stattdessen `BKIMBTIA` verwenden.
- Wenn Sie Impersonation im z/OS-Batch-Modus verwenden wollen, müssen Sie stattdessen `BROKER` verwenden.
- Wenn Sie Impersonation unter CICS verwenden wollen, müssen Sie stattdessen CICSETB verwenden.

Um `BKIMBTIA`, `BROKER` oder `CICSETB` dynamisch zur Laufzeit zu laden, geben Sie `RCA=BROKER RCALIAS=(BROKER,`*`stubname`*`)` an.

Weitere Einzelheiten finden Sie in der Dokumentation zum EntireX Communicator.

Anmerkung: Bitte prüfen Sie die Voraussetzungen für den Einsatz von `CICSETB` hinsichtlich der erforderlichen PPT-Einträge und der Adabas-Link-Routine.

Zugriff auf den EntireX Broker Stub auf UNIX und OpenVMS bereitstellen

Der EntireX Broker Stub wird im Zuge der EntireX-Installation automatisch zur Verfügung gestellt.

Zugriff auf den EntireX Broker-Stub unter Windows bereitstellen

Der EntireX-Broker-Stub wird im Zuge der EntireX-Installation automatisch zur Verfügung gestellt.

API-Version

Die mögliche ACI-Version wird zwischen dem RPC-Nukleus und EntireX ausgehandelt und auf den höchstmöglichen Wert gesetzt. Diese Version von Natural schränkt die ACI-Version 13 ein, auch wenn EntireX eine höhere ACI-Version unterstützt.

TCP/IP als Transportmethode verwenden

Wenn TCP/IP als Transportmethode verwendet wird und Sie den Serverknoten über einen Hostnamen adressieren, haben Sie folgende Möglichkeiten:

- Definieren Sie den Serverknoten im **Hosts and Services Directory** Ihrer TCP/IP-Installation.
- Verwenden Sie ein **Domain Name System** (DNS) für die Auflösung des Domänennamens.

Einrichten einer EntireX Broker-Umgebung

Fügen Sie in der EntireX Broker-Attributdatei Folgendes hinzu:

1. Für jeden Natural RPC Server muss eine Dienstdefinition wie folgt angegeben werden:

   `CLASS=RPC, SERVICE=CALLNAT, SERVER=`*`servername`*.

2. Wenn Sie die Konvertierungsdienste verwenden wollen, setzen Sie `CONVERSION=`*`userexit`*. In diesem Fall müssen Sie den Schlüsselwort-Subparameter `CPRPC` des Profilparameters `RPC` oder des Parameter-Makros `NTRPC` entsprechend einstellen.

   Wenn Sie **Reliable RPC** verwenden möchten, sind zusätzliche Einstellungen für jeden Natural RPC Server erforderlich, der Reliable RPC unterstützen soll:

- Das EntireX Broker-Attribut `MAX-UOWS` muss auf einen Wert größer Null gesetzt werden.
- Das EntireX Broker-Attribut `DEFERRED` (aufgeschoben) muss auf `YES` gesetzt werden, wenn der Client in der Lage sein soll, Reliable RPC-Nachrichten an einen RPC-Server zu senden, der dem EntireX Broker bekannt ist, aber noch nicht gestartet wurde.
- Das EntireX Broker-Attribut `STORE` muss auf `BROKER` gesetzt werden, wenn eine Wiederherstellung von Reliable RPC-Nachrichten nach einem Systemausfall möglich sein soll. Außerdem muss der persistente Speicher von EntireX Broker aktiviert sein.

- Die Lebensdauer der Reliable RPC Nachricht selbst (EntireX Broker Attribut `UWTIME`) und die Lebensdauer ihres Status (EntireX Broker Attribut `UWSTAT-LIFETIME`) müssen an Ihre Bedürfnisse angepasst werden.

6 Starten eines Natural RPC Servers

In diesem Kapitel wird beschrieben, wie Sie einen Natural RPC Server auf den verschiedenen Plattformen starten.

Vorbereitungen vor dem Start eines Natural RPC Servers

Jede Art von Natural-Sitzung kann als Natural RPC Server verwendet werden, aber normalerweise ist ein Natural-Server eine Natural-Sitzung, die als asynchrone Task oder als Hintergrund-Task gestartet wird.

Auf Großrechnern:

Um einen Server zu starten, haben Sie folgende Möglichkeiten:

- Erstellen Sie ein RPC-Server-spezifisches Natural-Parametermodul.

  Eine Liste der relevanten Parameter finden Sie im Abschnitt *Einrichten einer Natural RPC-Umgebung*, *RPC-Server-spezifischen Natural-Parameter setzen*.

  Dieses Parametermodul wird mit Ihrem Natural verlinkt.
- Alternativ dazu können Sie die RPC-Server-spezifischen Natural-Profilparameter auch dynamisch angeben.

  The RPC server-specific Natural profile parameters may also be specified in a Natural profile created with the `SYSPARM` utility. Natural would then be started with

  Die RPC-Server-spezifischen Natural-Profilparameter können auch in einem Natural-Profil angegeben werden, das mit der `SYSPARM` Utility erstellt wurde. Natural würde dann gestartet mit `PROFILE=serverprofile`,

  wobei `serverprofile` wobei

Unter Windows, UNIX oder OpenVMS:

Um einen Server zu starten, haben Sie die folgenden Möglichkeiten:

- Erstellen Sie eine RPC-Server-spezifische Natural-Parameterdatei.

  Eine Liste der relevanten Parameter finden Sie im Abschnitt *Einrichten einer Natural RPC-Umgebung*, *RPC-Server-spezifischen Natural-Parameter setzen*. Da dringend empfohlen wird, einen Natural RPC Server mit dem Natural-Profilparameter `BATCHMODE` zu starten, sollten Sie zusätzlich die Natural-Profilparameter `CMSYNIN`, `CMOBJIN` und `CMPRINT` angeben.

  Natural wird dann gestartet mit

  `PARM=serverparm`

wobei *serverparm* der Name der Parameterdatei ist.

- Alternativ können Sie die RPC-Server-spezifischen Natural-Profilparameter auch dynamisch angeben.

Wie ein Natural-Server gestartet wird, hängt von der Umgebung ab und wird in den entsprechenden Abschnitten weiter unten beschrieben. Der Einfachheit halber werden hier nur die zwingend erforderlichen RPC-Server-spezifischen Profilparameter aufgeführt. Die optionalen RPC-Server-spezifischen Profilparameter können entsprechend angegeben werden.

Starten eines Natural RPC Servers in einer Online-Großrechner-Umgebung (alle TP-Monitore)

Um einen Natural-Server in einer Online-Großrechner-Umgebung zu starten:

- Geben Sie in Ihrer TP-Monitor-Umgebung das folgende Kommando ein:

```
<natural> PROFILE=serverprofile
```

Oder:

```
<natural> RPC=(SERVER=ON,SRVNAME=servername,SRVNODE=nodename,RPCSIZE=n,MAXBUFF=n)
```

Dabei ist *<natural>* der Name, mit dem Sie Ihr Natural starten (Transaktionscode, Transaktionskennung, umgebungsabhängiger Nukleusname), und *serverprofile* ist der Name des Natural-Profils.

Starten eines Natural RPC Servers in einer Online-Großrechner-Umgebung (nur CICS)

In einer CICS-Umgebung werden Sie einen Natural RPC Server normalerweise im asynchronen Modus starten. Zusätzlich zu den oben beschriebenen Optionen für alle TP-Monitoren haben Sie für diesen Zweck die folgenden Möglichkeiten:

- Sie können das Natural-Programm `STARTSRV` in der Library `SYSRPC` verwenden, um einen Natural-Server im asynchronen Modus zu starten (siehe **unten**).
- Sie können einen Natural RPC Server im asynchronen Modus während des Starts von CICS starten (siehe **unten**).

Beim Starten des asynchronen Natural RPC Servers wird empfohlen, zusätzlich die Natural-Profilparameter TTYPE, INTENS, SENDER mit folgenden Einstellungen anzugeben:

```
TTYPE=ASYL,INTENS=1,SENDER=CSSL
```

Dies bewirkt, dass jede Ausgabe in das primäre Ausgabeziel im Zeilenmodus und nicht im 3270-Modus geschrieben wird. Anstelle von CSSL können Sie auch jedes andere CICS-Ausgabeziel angeben.

Starten eines Natural RPC Servers im asynchronen Modus mit STARTSRV

STARTSRV ist ein Beispiel-Frontend für RPCSSRV, das die asynchrone Natural-Sitzung startet.

Standardmäßig wird das asynchrone Natural mit der gleichen Transaktionskennung in der gleichen Library wie die aktuelle Sitzung gestartet.

Wenn Natural Security (NSC) verwendet wird, wird auch die Benutzerkennung der aktuellen Natural-Sitzung weitergegeben. Sie können die Eingabe an Ihre Anforderungen anpassen.

Beachten Sie, dass einige Natural-Profilparameter von RPCSSRV implizit hinzugefügt werden. Dies gilt insbesondere für den Schlüsselwort-Subparameter RPCSIZE des Profilparameters RPC. RPCSIZE ist standardmäßig auf MAXBUFF+4 eingestellt, wobei MAXBUFF der Wert ist, der im Masken-Feld Receiving buffer eingetragen ist. Sie können den Standardwert für RPCSIZE überschreiben, indem Sie RPC=(RPCSIZE=*n*) in das Masken-Feld Session parameter eintragen.

Wenn Sie beim Starten der Natural-Sitzung ein Natural-Programm ausführen wollen, können Sie das Masken-Feld User Stack verwenden. Der Inhalt von User Stack wird auf dem Natural STACK abgelegt und ausgeführt, bevor der Natural-Server aktiviert wird.

Um die beteiligten Natural-Profilparameter anzuzeigen, geben Sie *SHOW* in das Feld Transaction ID ein. STARTSRV zeigt Ihnen alle dynamischen Profilparameter an, die von RPCSSRV verwendet werden, um die asynchrone Natural-Sitzung zu starten.

Starten einer Natural RPC Server-Sitzung im asynchronen Modus während des Starts von CICS

Um eine Natural RPC Server-Sitzung im asynchronen Modus während des Starts von CICS zu starten, gehen Sie wie folgt vor:

Verwenden Sie das PLTPI, um ein Programm zu starten, das mit EXEC CICS START eine Natural-Sitzung mit allen erforderlichen RPC-spezifischen Natural-Profilparametern startet. Sie können das PLTPI-Beispielprogramm XNCIFRNP anpassen, das in der Natural CICS Source Library enthalten ist.

Bitte beachten Sie, dass die Natural-Sitzung unter der CICS-Standard-Benutzerkennung gestartet wird, die standardmäßig CICSUSER lautet. Diese Benutzerkennung wird den Natural-Systemvariablen *INIT-USER und *USER zugewiesen. Wenn Ihre Natural-Sitzung unter Natural Security läuft, müssen Sie daher eventuell ein Natural LOGON-Kommando auf den Natural STACK legen.

Starten eines Natural RPC Servers in einer Großrechner-Online-Umgebung (nur Com-plete)

In einer Com-plete-Umgebung werden Sie einen Natural RPC Server normalerweise im asynchronen Modus starten. Zusätzlich zu den oben beschriebenen Optionen für alle TP-Monitore haben Sie zu diesem Zweck die folgenden Möglichkeiten:

- Sie können das Natural-Programm `STARTSRV` in der Library `SYSRPC` verwenden, um einen Natural-Server im asynchronen Modus zu starten (siehe **unten**).
- Sie können einen Natural RPC Server im asynchronen Modus während des Starts von Com-plete starten (siehe **unten**).

Starten eines Natural RPC Servers im asynchronen Modus mit STARTSRV

STARTSRV ist ein Beispiel-Frontend für `RPCSSRV`, das die asynchrone Natural-Sitzung startet.

Standardmäßig wird das asynchrone Natural mit demselben Natural-Namen in derselben Library wie die aktuelle Sitzung gestartet.

Wenn Natural Security (NSC) verwendet wird, wird auch die Benutzerkennung der aktuellen Natural-Sitzung weitergegeben. Sie können die Eingabe an Ihre Anforderungen anpassen.

Beachten Sie, dass einige Natural-Profilparameter von `RPCSSRV` implizit hinzugefügt werden. Dies gilt insbesondere für den Schlüsselwort-Subparameter `RPCSIZE` des Profilparameters `RPC`. `RPCSIZE` ist standardmäßig auf `MAXBUFF+4` eingestellt, wobei `MAXBUFF` der im Masken-Feld `Receiving buffer` eingetragene Wert ist. Sie können den Standardwert für `RPCSIZE` überschreiben, indem Sie `RPC=(RPCSIZE=`*`n`*`)` in das Masken-Feld `Session parameter` eintragen.

Wenn Sie beim Starten der Natural-Sitzung ein Natural-Programm ausführen wollen, können Sie das Masken-Feld `User Stack` verwenden. Der Inhalt vom `User Stack` wird auf dem Natural `STACK` abgelegt und ausgeführt, bevor der Natural-Server aktiviert wird.

Um die beteiligten Natural-Profilparameter anzuzeigen, geben Sie `*SHOW*` in das Feld `Transaction ID` ein. `STARTSRV` zeigt Ihnen alle dynamischen Profilparameter an, die von `RPCSSRV` verwendet werden, um die asynchrone Natural-Sitzung zu starten.

Starten einer Natural RPC Server-Sitzung im asynchronen Modus während des Starts von Com-plete

Um eine Natural RPC Server-Sitzung im asynchronen Modus während des Starts von Com-plete zu starten, gehen Sie wie folgt vor:

Verwenden Sie die Startoption (sysparms) `STARTUPPGM`, um eine Natural-Sitzung mit allen erforderlichen RPC-spezifischen Natural-Profilparametern zu starten. Sie können die benötigten RPC-

spezifischen Parameter entweder einzeln angeben oder sie über ein Natural-Profil wie folgt festlegen:

```
STARTUPPGM='<natural> PROFILE=<serverprofile>'
```

Beachten Sie, dass die Natural-Sitzung unter der Benutzerkennung gestartet wird, unter der auch Com-plete gestartet wird. Diese Benutzerkennung ist den Natural-Systemvariablen `*INIT-USER` und `*USER` zugewiesen. Wenn Ihre Natural-Sitzung unter Natural Security läuft, müssen Sie daher eventuell ein Natural `LOGON`-Kommando auf den Natural `STACK` legen.

Starten eines Batch-Servers in einer Großrechner-Umgebung

Ein Batch-Server ist eine Standard-Natural-Batch-Sitzung, die mit den RPC-Parametern gestartet wird, die im Abschnitt *Einrichten einer Natural RPC-Umgebung*, *RPC-Server-spezifischen Natural-Parameter setzen* beschrieben sind.

Die folgenden Themen werden behandelt:

- Starten eines Batch-Servers unter z/OS
- Starten eines Batch-Servers unter z/VSE
- Starten eines Batch-Servers unter BS2000

Anmerkung: Ein Beispiel für eine JCL, die die Trace-Funktion verwendet, finden Sie unter *Betrieb einer Natural RPC-Umgebung* , *Verwendung der Server Trace Facility*.

Starten eines Batch-Servers unter z/OS

Beispiel-JCL für z/OS

```
//NATRPC   JOB  CLASS=K,MSGCLASS=X
//         EXEC PGM=NATOS,REGION=8M                <== Note 1
//STEPLIB  DD   DISP=SHR,DSN=SAG.NAT.LOAD
//         DD   DISP=SHR,DSN=SAG.EXX.LOAD
//         DD   DISP=SHR,DSN=SAG.ADA.LOAD          <== Note 2
//         DD   DISP=SHR,DSN=DB2_load_library      <== Note 3
//CMPRMIN  DD   *
IM=D,MADIO=0,MT=0,OBJIN=R,AUTO=OFF,MAXCL=0,ID=',',INTENS=1,
RPC=(SERVER=ON,SRVNAME=servername,SRVNODE=nodename)
RPC=(RPCSIZE=m,MAXBUFF=n),
STACK=(LOGON serverlibrary,userID,password)
/*
//CEEOPTS  DD *                                    <== Note 4
POSIX(ON)
/*
//SYSUDUMP DD SYSOUT=X
```

```
//CMPRINT  DD SYSOUT=X
/*
```

Anmerkungen:

1. Wenn der RPC-Server mit Subtasks läuft, werden mit der `EXEC PARM`-Anweisung Natural-Parameter nur an die Haupt-Task übergeben. Verwenden Sie stattdessen die `CMPRMIN`-Anweisung, um Parameter gleichzeitig an die Haupt-Task und die Subtasks zu übergeben.
2. Gilt nur, wenn die Adabas-Link-Routine `ADAUSER` oder der Natural-Profilparameter `ADANAME` verwendet wird.
3. Gilt nur für DB2-Benutzer.
4. Gilt nur, wenn SSL verwendet wird.

Beispiel-JCL für eine gestartete Task

Ein Beispiel für eine JCL für eine gestartete Task finden Sie in der Installations-Dokumentation von Natural for Mainframes; siehe *Installing Natural on z/OS*.

Ausführen eines Batch-Servers mit Replikaten

Sie können einen Batch-Server auch mit Replikaten laufen lassen, indem Sie den Schlüsselwort-Subparameter `NTASKS` des Profilparameters `RPC` oder des Parameter-Makros `NTRPC` auf einen Wert größer als `1` setzen.

Replikate werden als zusätzliche Server-Tasks an eine Natural-Haupt-Task angehängt. Sie ermöglichen es Ihnen, mehrere identische Server in derselben Region zu starten.

Starten eines Batch-Servers unter z/VSE

Beispiel-JCL für z/VSE

```
// LIBDEF PHASE,SEARCH=(SAGLIB.NATvrs,SAGLIB.EXXvrs,SAGLIB.ADAvrs),TEMP
// ASSGN SYS000,READER
// ASSGN SYSLST,FEE
// EXEC NATVSE,SIZE=AUTO,PARM='SYSRDR'
IM=D,MADIO=0,MT=0,OBJIN=R,AUTO=OFF,MAXCL=0,ID=',',INTENS=1,
RPC=(SERVER=ON,SRVNAME=servername,SRVNODE=nodename)
RPC=(RPCSIZE=m,MAXBUFF=n),
STACK=(LOGON serverlibrary,userID,password)
/*
```

wobei *vrs* für die jeweilige Produktversion steht. Siehe auch *Version* im *Glossar*.

Einen Batch-Server mit Replikaten ausführen

Sie können einen Batch-Server auch mit Replikaten laufen lassen, indem Sie den Schlüsselwort-Subparameter `NTASKS` des Profilparameters `RPC` oder des Parameter-Makros `NTRPC` auf einen Wert größer als `1` setzen.

Replikate werden als zusätzliche Server-Tasks an eine Natural-Haupt-Task angebunden. Sie ermöglichen es Ihnen, mehrere identische Server in derselben Region zu starten.

Starten eines Batch-Servers unter BS2000

Beispiel-JCL für BS2000

```
/.NATRPC      LOGON
/             SYSFILE   SYSOUT=output-file
/             SYSFILE   SYSDTA=(SYSCMD)
/             SYSFILE   SYSIPT=(SYSCMD)
/             STEP
/             SETSW      ON=2
/             EXEC NATBS2
IM=D,MADIO=0,MT=0,OBJIN=R,AUTO=OFF,MAXCL=0,ID=',',INTENS=1,
RPC=(SERVER=ON,SRVNAME=servername,SRVNODE=nodename)
RPC=(RPCSIZE=m,MAXBUFF=n),
STACK=(LOGON serverlibrary,userID,password)
/             EOF
```

Starten eines Natural RPC Servers in einer Windows-Umgebung

Um einen Natural RPC Servers unter Windows zu starten:

1 Erstellen Sie ein Shortcut-Verknüpfung zu Natural.

2 Geben Sie die Eigenschaften der Verknüpfung ein.

3 Editieren Sie im Textfeld **Target** den Natural-Pfad und fügen Sie an:

```
"<Path>\natural.exe" batchmode parm=serverparm
```

Dabei ist *serverparm* der Name der Parameterdatei.

Oder:

```
"<Path>\natural.exe" batchmode
server=on ↩
srvname=servername srvnode=nodename maxbuff=n cmsynin=cmsynin cmobjin=cmobjin cmprint=cmprint
```

Starten eines Natural RPC Servers in einer UNIX-Umgebung

Um einen Natural RPC Server unter UNIX zu starten:

- Geben Sie das folgende Kommando ein:

```
natural batchmode parm=serverparm &
```

Dabei ist *serverparm* der Name der Parameterdatei.

Oder:

```
natural batchmode
server=on ↩
srvname=servername srvnode=nodename maxbuff=n cmsynin=cmsynin cmobjin=cmobjin cmprint=cmprint
 &
```

Starten eines Natural RPC Servers in an OpenVMS Environment

Um einen Natural RPC Server unter OpenVMS zu starten:

1 Geben Sie in der DCL-Kommando-Prozedur *myserver*.com die folgenden Kommandos ein:

```
$ DEFINE NATOUTPUT NLA0:
$ NAT batchmode parm=serverparm
```

2 Übergeben Sie dann *myserver*.com an eine Batch-Warteschlange:

```
$ SUBMIT myserver.com
```

Besondere Aspekte bei Großrechner-Natural RPC Servern mit Replicaten

Dieser Abschnitt gilt für Großrechner-Natural-Server unter z/OS und z/VSE.

Natural RPC Batch Server mit NTASKS >1

Die Haupt-Task und alle Replikate laufen in der gleichen z/OS-Region oder z/VSE-Partition.

1. Use the reentrant batch link routine ADALNKR instead of ADALNK.

   Verwenden Sie die eintrittsinvariante (reentrant) Batch Link Routine ADALNKR anstelle von ADALNK.

   Wenn Sie ADAUSER verwenden wollen, dürfen Sie ADAUSER nicht mit Ihrem Frontend verlinken, da ADAUSER nicht eintrittsinvariant (reentrant) ist (siehe Punkt 5). Verwenden Sie stattdessen den Natural-Profilparameter ADANAME und setzen Sie ADANAME=ADAUSER. Dadurch wird Natural veranlasst, ADAUSER zur Laufzeit dynamisch zu laden.

   Hinweis für z/VSE: Wenn Sie ADAUSER verwenden, müssen Sie ADALNKR in ADALNK umbenennen.
2. Im Natural-Parametermodul:
   - Setzen Sie den Schlüsselwort-Subparameter NTASKS=*n* des Profilparameters RPC oder des Parametermakros NTRPC, wobei *n* die Anzahl der zu startenden parallelen Server (< 100), einschließlich der Haupt-Task, ist.

     Hinweis für z/VSE: Die Anzahl der Subtasks ist durch das Betriebssystem begrenzt. Fragen Sie Ihren Systemadministrator.
   - Verwenden Sie den Natural-Profilparameter ETID, um die Adabas-Benutzerkennung als Leerzeichen anzugeben. Dies ist notwendig, um einen NAT3048-Fehler (ETID nicht eindeutig im Adabas-Nukleus) beim Starten der Subtask zu vermeiden.
3. Bei Verwendung von dynamischen Natural-Profilparametern:

   Verwenden Sie den dynamischen Parameter-Dataset CMPRMIN, um die dynamischen Natural-Profilparameter an Natural zu übergeben. Verwenden Sie *nicht* die PARM-Karte oder den primären Kommando-Eingabe-Dataset CMSYNIN.
4. Bei Verwendung eines lokalen Bufferpools (nur z/OS):

   Jede Subtask ordnet ihren eigenen lokalen Bufferpool zu, es sei denn, Sie geben einen gemeinsam genutzten lokalen Bufferpool an. Siehe Subparameter LBPNAME des Profilparameters OSP oder des Parametermakros NTOSP (in der *Parameter-Referenz*-Dokumentation.

5. Im Natural Frontend-Link-Job (nur z/OS):

   Verlinken Sie das Frontend eintrittsinvariant (reentrant), indem Sie die Option `RENT` des Linkage-Editors verwenden.

   Wenn das Frontend nicht mittels der Option `RENT` verlinkt wäre, würde nur die Haupt-Task die Kommunikation mit dem EntireX Broker starten. Alle Subtasks würden von z/OS in einen `WAIT`-Status gesetzt, bis die Haupt-Task beendet worden wäre. Wenn Sie den RPC-Server später beenden würden, würde der Adressraum blockieren und müsste abgebrochen werden.
6. Make sure that any other modules that are additionally linked to the Natural nucleus are reentrant. Any dynamically loaded programs must also be reentrant.

   Stellen Sie sicher, dass alle anderen Module, die zusätzlich mit dem Natural-Nukleus verlinkt sind, eintrittsinvariant (reentrant) sind. Dynamisch geladene Programme müssen ebenfalls eintrittsinvariant sein.

   Hinweis für z/OS: Wenn Sie ein Modul nicht reentrant machen können, verlinken Sie das Modul als nicht wiederverwendbar, d.h. Sie sollten die Link-Option `RENT` oder `REUS` *nicht* angeben. Damit wird sichergestellt, dass jede Subtask ihre eigene Kopie erhält.

Ausführen eines Batch-Servers mit Replikaten

Eine Beispiel-JCL finden Sie unter *Verwendung der Server Trace Facility*.

Starten eines Natural RPC Servers über das RPC-Server-Frontend (nur bei z/OS Batch-Modus)

Im z/OS-Batch-Modus kann ein Natural RPC Server alternativ über das RPC-Server-Frontend gestartet werden. Dieser Ansatz ist bei Impersonation erforderlich und in anderen Fällen optional.

Wenn Sie das RPC-Server-Frontend ohne Impersonation verwenden, wird empfohlen, den Schlüsselwort-Subparameter `NTASKS` des Profilparameters `RPC` oder des Parametermakros `NTRPC` auf einen Wert größer als `1` zu setzen, da sonst kein nutzbringender Effekt entsteht. Die Überlegungen im Abschnitt *Besondere Aspekte bei Großrechner-Natural RPC Servers mit Replikaten* gelten auch, wenn Sie das RPC-Server-Frontend verwenden.

Das RPC-Server-Frontend verwendet die Natural-Server-Funktionalität; siehe *Natural as a Server under z/OS* (in der Natural-*Operations*-Dokumentation). Es zeichnet sich durch die folgenden Eigenschaften aus:

- Das Natural RPC Server-Frontend startet eine Reihe von Natural RPC Server-Sitzungen, die durch den Schlüsselwort-Subparameter `NTASKS` des Profilparameters `RPC` oder das Parameter-Makro `NTRPC` festgelegt sind.

- Alle Natural RPC Server-Sitzungen werden mit denselben Natural-Profilparametereinstellungen ausgeführt.

  Die Natural-Profilparametereinstellungen werden aus dem Natural-Parametermodul übernommen und können durch dynamisch angegebene Profilparameter, die mit der Transaktionskennung übergeben werden, überschrieben werden. Die dynamisch angegebenen Profilparameter müssen auf den Startup-Parameter folgen.
- Wenn alle Natural RPC Server-Sitzungen gerade von Clients verwendet werden (Ausführung einer Client-Anforderung oder Warten auf die nächste Anforderung innerhalb einer Konversation) und wenn der Schlüsselwort-Subparameter `NTASKS` des Profilparameters `RPC` oder des Parameter-Makros `NTRPC` auf einen Wert größer als `1` gesetzt ist, werden zusätzliche Natural RPC Server-Sitzungen gestartet. Diese Natural RPC Server-Sitzungen werden beim ersten EntireX Broker-Timeout automatisch beendet, vorausgesetzt, es gibt mindestens eine weitere Natural RPC Server-Sitzung, die nicht von einem Client verwendet wird. Wenn alle anderen Natural RPC Server-Sitzungen gerade von Clients genutzt werden, bleibt die zusätzliche RPC-Server-Sitzung bis zum nächsten EntireX Broker-Timeout bestehen. Dadurch wird sichergestellt, dass immer ein Natural RPC Server zur Verfügung steht, um eine neue Client-Anforderung zu bearbeiten.
- Die Natural RPC Server-Sitzungen werden in einer Thread-Umgebung ausgeführt, die der von Natural-Sitzungen ähnelt, die in einem TP-Monitor-System ausgeführt werden.
- **Mit Impersonation:**
  Am Ende eines nicht-konversationellen `CALLNAT` und am Ende einer Konversation werden alle Datenbanksitzungen und alle Arbeitsdateien geschlossen. Dadurch wird sichergestellt, dass die nächste Client-Anfrage die Datenbank und die Arbeitsdateien mit der eigenen Benutzerkennung öffnet.
- **Ohne Impersonation:**
  Nach dem ersten EntireX Broker Timeout werden alle Datenbanksitzungen und alle Arbeitsdateien geschlossen. Dadurch wird sichergestellt, dass keine Ressourcen während der Wartezeiten blockiert werden.

Startparameter:

Die erforderlichen Startparameter werden im Parameter `PARM=` der `EXEC`-Anweisung in der JCL übergeben. Diese Parameter sind:

- Der Name des Natural z/OS-Batch-Nukleus,
- die Größe eines Speicher-Threads,
- das optionale Schlüsselwort `UCTRAN`.

  `UCTRAN` gibt an, dass alle Meldungen des RPC-Server-Frontends in Großbuchstaben umgewandelt werden.

Diese Parameter müssen durch Kommas getrennt sein und ohne führende oder nachfolgende Leerzeichen eingegeben werden:

PARM='*Natural-z/OS-batch-nucleus,size-of-thread*[,,UCTRAN]'

Aus Kompatibilitätsgründen müssen `UCTRAN` zwei aufeinanderfolgende Kommas vorangestellt werden.

Siehe auch die ***Beispiel-JCL*** unten.

Hinweise zur Ausführung:

- Der Natural z/OS Batch-Nukleus wird dynamisch geladen.

  Die Load Library, die den z/OS-Batch-Nukleus enthält, muss in der Steplib-Verkettung verfügbar sein.
- Der EntireX-Broker-Stub `NATETB23` darf nicht verwendet werden.

  Auf den EntireX Broker wird außerhalb des Natural-Kontextes zugegriffen. Daher müssen Sie den EntireX-Broker-Stub `BROKER` verwenden, siehe ***Zugang zum EntireX Broker Stub auf dem Großrechner bereitstellen*** .

Nur bei Impersonation:
Wenn die **Impersonation**-Funktion verwendet wird, muss das RPC-Server-Frontend von einer Authorized Program Facility (APF) Library ausgeführt werden. Es wird empfohlen, das RPC-Server-Frontend von einer APF-autorisierten `LINKLIST` Library aus auszuführen. Dadurch entfällt die Notwendigkeit, die gesamte Steplib-Verkettung APF-autorisiert bereitzustellen.

Nur bei Natural Security:
Wenn das Natural RPC Server-Frontend mit dem Profilparameter `AUTO=OFFAUTO=OFF` gestartet wird, müssen Sie ein Natural-Systemkommando `LOGON` mit Library-Kennung, Benutzerkennung und Passwort auf dem Natural-Stack angeben:

```
STACK=(LOGON library-id;user-id;password)
```

Beispiel-JCL:

```
//NATRPC   JOB  CLASS=K,MSGCLASS=X
//         EXEC PGM=RPC-FRONT,REGION=8M
//         PARM='Natural-z/OS-interface-module,1000'
//STEPLIB  DD   DISP=SHR,DSN=SAG.NAT.LOAD
//         DD   DISP=SHR,DSN=SAG.EXX.LOAD
//         DD   DISP=SHR,DSN=SAG.ADA.LOAD         <== Note 1
//         DD   DISP=SHR,DSN=DB2_load_library     <== Note 2
//CMPRMIN  DD   *
IM=D,MADIO=0,MT=0,OBJIN=R,AUTO=OFF,MAXCL=0,ID=',',INTENS=1,
PRINT=((10),AM=STD)
RPC=(SERVER=ON,SRVNAME=servername,SRVNODE=nodename,NTASKS=3)
RPC=(RPCSIZE=m,MAXBUFF=n,TRACE=2),
RCA=BROKER,RCALIAS=(BROKER,BROKER)
STACK=(LOGON serverlibrary,userID,password)
```

```
/*
//CEEOPTS  DD *                                    <== Note 3
POSIX(ON)
*
//SYSUDUMP DD SYSOUT=X
//CMPRT10  DD SYSOUT=X
//CMPRT101 DD SYSOUT=X
//CMPRT102 DD SYSOUT=X
//CMPRT199 DD SYSOUT=X
//CMPRINT  DD SYSOUT=X
//CMPRINT1 DD SYSOUT=X
//CMPRINT2 DD SYSOUT=X
//CMPRIN99 DD SYSOUT=X
/*
```

Anmerkungen:

1. Gilt nur, wenn die Adabas-Link-Routine `ADAUSER` oder der Natural-Profilparameter `ADANAME` verwendet wird.
2. Gilt nur für DB2-Benutzer.
3. Gilt nur, wenn SSL verwendet wird.

Starten eines Natural RPC Servers über das RPC Server Frontend (nur CICS)

In CICS kann ein Natural RPC Server alternativ auch über das RPC-Server-Frontend gestartet werden. Diese Vorgehensweise ist erforderlich, wenn Impersonation verwendet wird, und ist in anderen Fällen optional.

Wenn Sie das RPC-Server-Frontend ohne Impersonation verwenden, wird empfohlen, den Schlüsselwort-Subparameter `NTASKS` des Profilparameters `RPC` oder des Parameter-Makros `NTRPC` auf einen Wert größer als `1` zu setzen. Andernfalls ergibt sich kein nutzbringender Effekt.

Das RPC-Server-Frontend verwendet die Natural Server-Funktionalität. Es zeichnet sich durch die folgenden Merkmale aus:

- Das Natural RPC Server-Frontend wird über die Transaktionskennung gestartet, die im Schritt *Customize CICS* für das Natural RPC Server-Frontend definiert wurde. Siehe *Installing the Natural CICS Interface on z/OS*.

  Die Transaktionskennung können Sie entweder an einem Terminal eingeben oder Sie können das Natural-Programm `STARTSFE` in der Library `SYSRPC` verwenden, um das Natural RPC Server-Frontend im asynchronen Modus zu starten.

  Das Natural RPC Server-Frontend benötigt den Namen des Natural CICS-Interface-Nukleus als Startparameter. Dieser Startparameter wird mit der Transaktionskennung übergeben.

- Das Natural RPC Server-Frontend startet eine Anzahl von Natural RPC Server-Sitzungen, die durch den Schlüsselwort-Subparameter `NTASKS` des Profilparameters `RPC` oder des Parameter-Makros `NTRPC` festgelegt wird.
- Alle Natural RPC Server-Sitzungen werden mit denselben Natural-Profilparametereinstellungen ausgeführt.

  Die Natural-Profilparametereinstellungen werden aus dem Natural-Parametermodul übernommen und können durch dynamisch angegebene Profilparameter, die mit der Transaktionskennung übergeben werden, überschrieben werden. Die dynamisch angegebenen Profilparameter müssen auf den Startparameter folgen.
- Wenn alle Natural RPC Server-Sitzungen gerade von Clients verwendet werden (Ausführung einer Client-Anforderung oder Warten auf die nächste Anforderung innerhalb einer Konversation) und wenn der Schlüsselwort-Subparameter `NTASKS` des Profilparameters `RPC` oder des Parameter-Makros `NTRPC` auf einen Wert größer als `1` gesetzt ist, werden zusätzliche Natural RPC Server-Sitzungen gestartet. Diese Natural RPC Server-Sitzungen werden beim ersten EntireX Broker-Timeout automatisch beendet, vorausgesetzt, es gibt mindestens eine weitere Natural RPC Server-Sitzung, die nicht von einem Client verwendet wird. Wenn alle anderen Natural RPC Server-Sitzungen von Clients genutzt werden, bleibt die zusätzliche RPC-Server-Sitzung bis zum nächsten EntireX Broker Timeout bestehen. Damit wird sichergestellt, dass immer ein Natural RPC Server zur Verfügung steht, um eine neue Client-Anforderung zu bearbeiten.
- Die Natural RPC Server-Sitzungen werden in einer Thread-Umgebung ausgeführt, die der von Natural-Sitzungen ähnelt, die in einem TP-Monitor-System ausgeführt werden.
- Alle inaktiven Natural RPC Server-Sitzungen (Sitzungen, die auf eine Client-Anfrage warten) werden mit Hilfe der Natural Roll Facilities under CICS ausgelagert, die für den verwendeten Natural CICS-Schnittstellenkern definiert sind.
- **Mit Impersonation:**
  Zu Beginn eines nicht-konversationellen `CALLNAT` und zu Beginn einer Konversation wird mit der Option `USERID()` des Kommandos `EXEC CICS START TRANSID()` eine neue CICS-Worker-Task unter der Benutzerkennung des Clients gestartet. Die Client-Anforderung wird von Natural in dieser Worker-Task ausgeführt. Während die Client-Anforderung ausgeführt wird, wartet die Natural RPC Server-Sitzung auf die Beendigung der Worker-Task.

  Am Ende eines nicht-konversationellen `CALLNAT` und am Ende einer Konversation wird die Worker-Task beendet, alle Datenbanken werden geschlossen und alle CICS-Ressourcen werden freigegeben. Dadurch wird sichergestellt, dass die nächste Client-Anfrage die Datenbank öffnet und auf die CICS-Ressourcen mit ihrer eigenen Benutzerkennung zugreift.
- **Ohne Impersonation**
  Die Client-Anforderung wird von der Natural RPC Server-Sitzung selbst ausgeführt.

  Nach dem ersten EntireX Broker Timeout werden alle Datenbanken geschlossen und alle CICS-Ressourcen freigegeben. Dadurch wird sichergestellt, dass keine Ressourcen während der Wartezeiten blockiert werden.

Startparameter:

Die erforderlichen Startparameter werden mit der Transaktionskennung übergeben. Diese Parameter sind:

- Der Name des Natural CICS Interface-Nukleus <*ncistart*>.
- Das optionale Schlüsselwort UCTRAN.

  UCTRAN gibt an, dass alle Meldungen des RPC-Server-Frontends in Großbuchstaben umgewandelt werden.
- Eine optionale Natural-Profilparameter-Zeichenkette.

Beispiel für den Start an einem Terminal:

```
<natural> <ncistart>[,UCTRAN]
RPC=(SERVER=ON,SRVNAME=servername,SRVNODE=nodename,RPCSIZE=n, MAXBUFF=n)
RCA=BROKER,RCALIAS=(BROKER,CICSETB)
```

Dabei ist <*natural*> die Transaktionskennung, mit der Sie Ihr Natural RPC Server-Frontend starten, und <*ncistart*> der Name Ihres Natural CICS-Interface-Nukleus.

Hinweise zur Ausführung:

- Wenn der Natural Roll Server erforderlich ist (NCMDIR-Parameter ROLLSRV ist auf YES gesetzt), müssen Sie einen Natural Roll Server für die verwendete Subsystemkennung (wie durch den Natural-Profilparameter SUBSID definiert) starten, bevor das Natural RPC Server-Frontend gestartet wird.
- Die Transaktionskennung des RPC-Server-Frontends wird verwendet, um die RPC-Server-Umgebung zu identifizieren.

  Starten Sie nicht mehr als ein Natural RPC Server-Frontend mit derselben Transaktionskennung.
- Das ausführbare NCI-Modul wird dynamisch geladen.

  Die Load Library, die das ausführbare NCI-Modul enthält, muss in der DFHRPL-Verkettung verfügbar sein.
- Der EntireX-Broker-Stub NATETB23 darf nicht verwendet werden.

  Auf den EntireX-Broker wird außerhalb des Natural-Kontextes zugegriffen. Daher müssen Sie den EntireX-Broker-Stub CICSETB verwenden, siehe ***Zugang zum EntireX Broker Stub auf dem Großrechner bereitstellen*** .

Nur bei Impersonation:
Wenn die Impersonation-Funktion verwendet wird, startet das RPC-Server-Frontend Worker-Tasks. Stellen Sie sicher, dass die Einstellung des CICS-System-Initialisierungsparameters MXT in Ihrer CICS-Installation hoch genug ist.

Nur bei Natural Security:
Wenn das Natural RPC Server-Frontend mit dem Profilparameter `AUTO` gestartet wird, müssen Sie auf den Natural-Stack ein Natural-Systemkommando `LOGON` mit Library-Kennung, Benutzerkennung und Passwort legen:

```
STACK=(LOGON library-id;user-id;password).
```

7 Beenden eines Natural RPC Servers

In diesem Kapitel wird beschrieben, wie Sie einen Natural RPC Server beenden können. Dies kann entweder durch das Beenden eines einzelnen RPC-Server-Replikats oder des EntireX-Broker-Dienstes, der die RPC-Verbindung unterstützt, erfolgen. Es gibt mehrere Vorgehensweisen.

SYSRPC Utility benutzen

Diese Methode zum Beenden eines Natural RPC Servers kann entweder bei einem einzelnen RPC-Server-Replikat oder dem EntireX Broker-Dienst angewendet werden:

- Mit SYSRPC ein einzelnes RPC-Server-Replikat beenden
- Mit SYSRPC einen EntireX-Broker-Dienst beenden

Mit SYSRPC ein einzelnes RPC-Server-Replikat beenden

Benutzen Sie das Kommando `TE` (Terminate RPC Server) in der `SYSRPC` Utility, wie unter *RPC-Server beenden* in der *SYSRPC Utility*-Dokumentation beschrieben.

Anmerkung: Ein Natural RPC Server kann nur beendet werden, wenn der Server zurzeit weder ein entferntes `CALLNAT` ausführt noch auf die nächste `CALLNAT`-Anforderung in einer Konversation wartet.

Mit SYSRPC einen EntireX-Broker-Dienst beenden

Verwenden Sie das Kommando `TS` (Terminate EntireX Broker Service) in der `SYSRPC` Utility, wie unter *RPC-Server beenden* in der *SYSRPC Utility*-Dokumentation beschrieben.

EntireX System Management Hub verwenden

Wählen Sie zum Abmelden die Schaltfläche **Deregister** in der Teilbaumstruktur **Server** des EntireX System Management Hub. Sie können auch die Schaltfläche **Deregister** in der Teilbaumstruktur **Service** des EntireX System Management Hub verwenden.

Anwendungsprogrammierschnittstelle USR2073N verwenden

Die Anwendungsprogrammierschnittstelle (API) USR2073N ermöglicht es Ihnen, einen Natural RPC Server anzupingen oder zu beenden.

Die Schnittstelle sendet ein PING- oder TERMINATE-Kommando an einen RPC-Server, der durch den Knotennamen und den Servernamen angegeben wird. Die zurückgegebene Nachricht enthält die folgenden Informationen:

- die Version des laufenden Servers (PING) bzw.
- die Bestätigung der Beendigung (TERMINATE) oder
- eine Fehlermeldung.

Um USR2073N zu benutzen:

1 Kopieren Sie das Subprogramm USR2073N aus der Library SYSEXT in die Library SYSTEM oder in die Steplib Library oder in eine beliebige Anwendung in der Serverumgebung.

2 Geben Sie mit einem DEFINE DATA-Statement im Structured Mode oder einem RESET-Statement im Reporting Mode die folgenden Parameter an:

Parameter	Format	I/O	Beschreibung	
FUNCTION	A10	I	PING	Fragt den RPC-Server nach seiner Version.
			TERMINATE	Leitet die Beendigung der Natural RPC Server-Task ein.
SRVNODE	A192	I	Knotenname. Siehe Schlüsselwort-Subparameter SRVNODE des Profilparameters RPC oder des Parameter-Makros NTRPC.	
SRVNAME	A32	I	Name des Servers. Siehe Schlüsselwort-Subparameter SRVNAME des Profilparameters RPC oder des Parameter-Makros NTRPC.	
LOGON	A1	I	Das Logon-Flag Y (ja) zeigt an, dass Anmeldedaten an den Natural RPC Server übermittelt werden müssen.	
USER-ID PASSWORD	A8	I	Wenn LOGON=Y, dann sind Benutzerkennung und Passwort die Anmeldedaten für Natural Security. Wenn der Client nicht unter Natural Security (NSC) läuft und das Logon-Flag gesetzt ist, müssen die Benutzerkennung und das Passwort angegeben werden, es sei denn, die Daten sind über die Anwendungsprogrammierschnittstelle USR1071N eingegeben worden. Siehe *Verwendung von Security*, ***Natural RPC mit Natural Security verwenden***.	

Parameter	Format	I/O	Beschreibung	
MSG	A79	O	Zurückgegebene Nachricht.	
RC	I2	O	Rückgabecode. Mögliche Werte:	
			0	MSG enthält eine normale Nachricht vom RPC-Server oder EntireX Broker.
			1	MSG enthält eine Fehlermeldung vom RPC-Server oder EntireX Broker.
			2	MSG enthält eine Fehlermeldung vom Interface.
			3	Natural Security-Fehler.

3 Bevor Sie die API aufrufen, füllen Sie die oben aufgeführten **I**nput-Variablen.

Anwendungsprogrammierschnittstelle USR2075N verwenden

Über die Anwendungsprogrammierschnittstelle (API) USR2075N können Sie einen EntireX-Brokerdienst aus Ihrer Anwendung heraus zu beenden.

Mit dem Kommando TERMINATE-SERVICE verwendet die Schnittstelle den **Command and Information Service** von EntireX, um die Aufgabe zu erfüllen. Zunächst sendet sie eine Anmeldeanforderung an den EntireX Broker. Dann erhält sie eine Liste aller Server, die nach Serverklasse, Servername und Diensttyp spezifiziert ist. Schließlich sendet sie an jeden Server eine Abschaltanforderung. Eine Meldung zeigt an, wie viele Server beendet wurden.

Mit dem Kommando PING wird der Identifikationsstring eines Servers zurückgegeben.

Um USR2075N zu verwenden:

1 Kopieren Sie das Subprogramm USR2075N aus der Library SYSEXT in die Library SYSTEM oder in die Steplib Library oder in eine beliebige Anwendung in der Serverumgebung.

2 Geben Sie mit einem DEFINE DATA-Statement im Structured Mode oder einem RESET-Statement im Reporting Mode die folgenden Parameter an:

Parameter	Format	I/O	Beschreibung	
FUNCTION	A17	I	Geben Sie TERMINATE-SERVICE an. Dies sendet eine Abschaltanforderung an jeden mit den Parametern SRVCLASS, SRVNAME und SERVICE (Diensttyp) angegebenen Server, siehe unten. Oder: Geben Sie PING an. Dadurch wird der Identifikationsstring eines Servers zurückgegeben.	
SRVNODE	A192	I	Geben Sie den Knotennamen an. Siehe Schlüsselwort-Subparameter SRVNODE des Profilparameters RPC oder des Parameter-Makros NTRPC.	
SRVCLASS	A32	I	Geben Sie die Serverklasse an. Für Natural RPC Server ist dies RPC.	
SRVNAME	A32	I	Name des Servers. Siehe Schlüsselwort-Subparameter SRVNAME des Profilparameters RPC oder des Parameter-Makros NTRPC.	
SERVICE	A32	I	Geben Sie die Art des Dienstes an. Bei Natural RPC Servern ist dies CALLNAT.	
IMMEDIATE	L		TRUE	Sofortige Beendigung aller Konversationen für den angegebenen Server.
			FALSE	Bestehende Konversationen dürfen normal beendet werden; neue Konversationen werden nicht akzeptiert.
USER-ID	A32	I	Geben Sie die Benutzerkennung für die Anmeldung beim EntireX Broker an.	
PASSWORD	A32	I	Wenn die EntireX Broker Security aktiv ist: Geben Sie das Passwort für den EntireX Broker an. Wenn die Security von EntireX Broker aktiv ist, müssen Sie ein Passwort angeben. Wenn die Security von EntireX Broker die Trusted User ID-Technik verwendet (nur auf Großrechnern), können Sie anstelle eines Benutzerpasswords das reservierte Passwort *TRUSTED verwenden.	
MSG	A (dynamic)	O	Zurückgegebene Nachricht.	
RC	I2	O	Rückgabecode. Mögliche Werte:	
			0	MSG enthält eine normale Nachricht vom EntireX Broker.
			1	MSG enthält eine Fehlermeldung vom EntireX-Broker..
			3	MSG enthält eine Fehlermeldung von Natural Security auf der Client-Seite.

3 Bevor Sie die API aufrufen, geben Sie die oben aufgeführten Input-Variablen ein.

Anwendungsprogrammierschnittstelle USR8220N verwenden

Die Anwendungsprogrammierschnittstelle (API) USR8208N ist eine erweiterte Version der API USR2075N mit den folgenden neuen Funktionen:

- Mit dem Kommando TERMINATE-SERVICE verwendet die Schnittstelle den **Command and Information Service** von EntireX, um die Aufgabe zu erfüllen. Das Kommando SHUTDOWN-SERVICE wird an den EntireX Broker gesendet, wenn dessen Version 5 oder höher ist. Es werden Versionsinformationen über den EntireX-Broker und den Befehls- und Informationsdienst selbst zurückgeliefert. Ansonsten wird die gleiche Methode wie in USR2075N verwendet.
- Mit dem Kommando INFO-SERVICE werden zusätzliche Informationen über den/die angegebenen Dienst(e) vom EntireX Broker zurückgegeben, wie SERVER-CLASS, SERVER-NAME, SERVICE, TRANS, CONV-NONACT usw. Eine ausführliche Beschreibung finden Sie in der *webMethods EntireX V9.0.1* Dokumentation unter *Components and Features of EntireX* > *ACI Programming* > *Broker CIS Data Structures* > *Information Reply Structures* > *SERVICE-OBJECT (Struct INFO_SV)*.

Um USR8208N zu verwenden:

- Siehe die ähnliche Anweisung ***Um USR2075 zu verwenden***.

Anwendungsprogrammierschnittstelle USR8220N verwenden

Die Anwendungsprogrammierschnittstelle (API) USR8220N löst die Beendigung des Natural RPC Servers aus. Nach dem Aufruf von USR8220N können Sie eine Bestätigungsmeldung an den RPC-Client senden. Diese API ist nützlich, um den Natural RPC Server bei schweren Fehlern explizit zu beenden.

Um USR8220N zu verwenden:

1 Kopieren Sie das Subprogramm USR8220N aus der Library SYSEXT in die Library SYSTEM oder in die Steplib Library oder in eine beliebige Anwendung in der Natural RPC Server-Umgebung.

2 Erstellen Sie ein Natural RPC Server-Subprogramm, das die API USR8220N aufruft und eine optionale Bestätigungsmeldung an das RPC-Client-Programm sendet.

Server-Beendigung bei Verwendung eines Attach Manager

Der Schlüsselwort-Subparameter SRVTERM des Profilparameters RPC oder des Parameter-Makros NTRPC beeinflusst das Beendigungsverhalten eines Natural RPC Servers. Standardmäßig wird ein Server nie beendet (SRVTERM=NEVER), es sei denn, eine der zuvor beschriebenen Beendigungsmethoden wird angewendet.

Wenn Sie einen Attach Manager verwenden, um einen Natural RPC Server auf Anforderung dynamisch zu starten, sollten Sie SRVTERM auf TIMEOUT setzen. Mit dieser Parametereinstellung wird ein Natural RPC Server automatisch beendet, wenn die Wartezeit für die nächste Client-Anforderung außerhalb einer RPC-Konversation überschritten wird.

User Exit NATRPC99

Dieser Exit wird aufgerufen, nachdem sich der Natural RPC Server vom Serverknoten per Deregister und Logoff abgemeldet hat.

- Wird kein Programm NATRPC99 gefunden, dann wird der Server sofort wie normalerweise beendet.
- Wird das Programm NATRPC99 gefunden, läuft der Server als normale Natural-Sitzung weiter.

NATRPC99 wird mit einem FETCH-Statement ohne Parameter aufgerufen, d.h. es werden keine Daten auf den Natural Stack gelegt, bevor NATRPC99 aufgerufen wurde.

Sie können in NATRPC99 beliebiges Coding hinzufügen, auch Statements zur Übergabesteuerung (FETCH, CALLNAT, PERFORM) und Statements zur Beendigung des Programms (STOP, ESCAPETERMINATE).

Wenn NATRPC99 mit einem RETURN- oder STOP-Statement beendet wird, kehrt Natural zum NEXT-Prompt zurück. Wenn der NEXT-Prompt in der verwendeten Umgebung nicht unterstützt wird (Profilparameter CM=OFF, asynchrone Natural-Sitzung usw.), bricht die Sitzung ab. Andernfalls versucht die Sitzung, den nächsten Befehl aus der primären Eingabedatei bzw. dem primären Dataset für Natural-Befehle und INPUT-Daten zu lesen.

Wichtige Hinweise:

1. NATRPC99 muss ein Natural-Programm sein.
2. NATRPC99 muss sich in der Library SYSTEM in der Systemdatei FUSER befinden. Die Steplib-Verkettung der Library, bei der der Server gerade angemeldet ist, wird nicht ausgewertet, um NATRPC99 zu finden.

3. Natural-Objekte, die von NATRPC99 aufgerufen werden (FETCH, CALLNAT, PERFORM), müssen sich entweder in der Library befinden, bei der der Server angemeldet ist, oder in einer seiner Steplibs (einschließlich SYSTEM in der Systemdatei FUSER).
4. NATRPC99 wird für alle Arten von normalen Natural RPC Server-Beendigungen aufgerufen. Er wird nicht bei Abbrüchen (Abends) aufgerufen.

8 Betrieb einer Natural RPC-Umgebung

In diesem Abschnitt werden überwiegend Aufgaben beschrieben, die für den Betrieb einer Natural RPC-Umgebung relevant sind.

Einige dieser Aufgaben werden mit der Utility SYSRPC ausgeführt. Weitere Informationen siehe Dokumentation zur *SYSRPC Utility*.

RPC-Server-Adressen festlegen

Jeder Remote-CALLNAT-Anforderung muss ein Server zugewiesen werden (identifiziert durch *servername* und *nodename*), bei dem das CALLNAT-Statement ausgeführt werden soll. Daher müssen alle Subprogramme, auf die remote zugegriffen werden soll, wie folgt definiert werden:

- in einem lokalen Service Directory auf der Client-Seite,
- oder in einem Remote Directory, auf das über einen Remote Directory Server zugegriffen wird,
- oder über die Standard-Server-Adressierung mit dem Schlüsselwort-Subparameter DFS des Profilparameters RPC oder des Parameter-Makros NTRPC,
- oder innerhalb der Client-Anwendung selbst über die Standard-Serveradressierung.

Zusätzlich zu den oben genannten Methoden können Sie auch alternative Server angeben.

Nachfolgend finden Sie Informationen zu:

Lokale Verzeichniseinträge verwenden

Alle Daten des lokalen Service Directory eines Clients werden in dem Subprogramm NATCLTGS gespeichert. Zur Ausführungszeit wird dieses Subprogramm verwendet, um auf den Ziel-Server aufzusuchen. Folglich muss NATCLTGS in der Client-Anwendung oder in einer der für die Anwendung definierten Natural Steplibs vorhanden sein.

Wenn NATCLTGS nicht in eine Steplib generiert wurde oder sich auf einem anderen Rechner befindet, verwenden Sie die entsprechende Natural-Utility (SYSMAIN oder den Natural Object Handler), um NATCLTGS in eine der für die Anwendung definierten Steplibs zu verschieben.

Wenn Sie ein NATCLTGS-Subprogramm zur gemeinsamen Nutzung verwenden, müssen Sie es allen Client-Umgebungen zur Verfügung stellen, z.B. indem Sie es in die Library SYSTEM kopieren, oder, wenn eine einzelne Kopie für einen Client verwendet wird, muss es für diesen Client mit der Funktion Service Directory Maintenance der SYSRPC Utility gepflegt werden.

Zur Definition und Bearbeitung von RPC-Service-Einträgen siehe Kapitel *Service Directory Maintenance* in der *SYSRPC Utility*-Dokumentation.

Remote Directory-Einträge benutzen

Ein Remote Directory enthält Diensteinträge (Service Entries), die mehreren Natural-Clients zur Verfügung gestellt werden können. Die Natural-Clients können diese Diensteinträge von Remote Directory-Servern abrufen. Informationen über den Zweck und die Installation von Remote-Directory-Servern finden Sie unter ***Verwendung eines Remote Directory Servers***.

Standard-Serveradresse beim Start von Natural Operations festlegen

Anstatt einen Server über ein lokales oder entferntes Service Directory zu adressieren, können Sie mit dem Schlüsselwort-Subparameter `DFS` des Profilparameters `RPC` oder des Parameter-Makros `NTRPC`, wie in Ihrer Natural-*Parameter-Referenz*-Dokumentation beschrieben, einen Standard-Server voreinstellen. Diese Serveradresse wird verwendet, wenn das Subprogramm weder im lokalen noch im entfernten Service Directory gefunden werden kann.

Die `DFS`-Einstellung bestimmt den Standard-Server für die gesamte Sitzung oder bis sie dynamisch überschrieben wird.

Wenn keine `DFS`-Einstellung vorhanden ist und die Serveradresse eines bestimmten Remote Procedure Call nicht im Service Directory gefunden werden konnte, wird eine Natural-Fehlermeldung zurückgegeben.

Eine in einer Client-Anwendung definierte Standard-Serveradresse bleibt auch dann aktiv, wenn Sie sich bei einer anderen Library anmelden oder wenn ein Natural-Fehler auftritt.

Festlegen einer Standard-Serveradresse innerhalb einer Natural-Sitzung

Die Client-Anwendung selbst kann zur Laufzeit dynamisch eine Standard-Serveradresse angeben. Dazu gibt es in Natural die Anwendungsprogrammierschnittstelle `USR2007N`. Über diese Schnittstelle können Sie eine Standard-Serveradresse festlegen, die immer dann verwendet werden soll, wenn ein entferntes Programm nicht über das Diensteverzeichnis (Service Directory) angesprochen werden kann.

Um `USR2007N` zu vernwenden:

1. Kopieren Sie das Subprogramm `USR2007N` aus der Library `SYSEXT` in die Library `SYSTEM` oder in die Steplib Library oder in eine beliebige Anwendung in der Serverumgebung.
2. Geben Sie mit dem `DEFINE DATA`-Statement im Structured Mode oder dem `RESET`-Statement im Reporting Mode die folgenden Parameter an:

Parameter	Format	Beschreibung	
function	A1	Funktion. Mögliche Werte:	
		P (Put)	Legt fest, dass die Serveradresse (bestehend aus den Parametern *nodename* und *servername*, siehe unten) die Standardadresse für alle nachfolgenden Remote-Prozeduraufrufe ist, die nicht im Diensteverzeichnis (Service Directory) definiert sind. Um eine Standard-Serveradresse zu entfernen, geben Sie ein Leerzeichen für *nodename* und *servername* an.
		G (Get)	Ruft die aktuelle Standard-Server-Adresse ab, die mit der Funktion P festgelegt wurde.
nodename	A192	Gibt den Namen des zu adressierenden Serverknotens an bzw. zurück. **Anmerkung:** Aus Kompatibilitätsgründen wird nodename mit der Option BY VALUE oder BY VALUE RESULT definiert (siehe Abschnitt parameter-data-definition in der Beschreibung des DEFINE DATA-Statement), um bestehende Aufrufer zu unterstützen, die ein A8- oder A32-Feld für *nodename* übergeben. Das in der Library SYSEXT enthaltene Beispiel USR2007P unterstützt bis zu 32 Zeichen.	
servername	A32	Gibt den Namen des Servers, der angesprochen werden soll, an oder zurück. **Anmerkung:** Aus Kompatibilitätsgründen wird servername mit der Option BY VALUE oder BY VALUE RESULT definiert (siehe Abschnitt parameter-data-definition in der Beschreibung des DEFINE DATA-Statement), um bestehende Aufrufer zu unterstützen, die ein A8-Feld für *servername* übergeben.	
logon	A1	Gibt die Logon-Option zurück, siehe ***Logon-Option benutzen***.	
protocol	A1	Gibt das Transportprotokoll an bzw. zurück. Gültiger Wert: B (=EntireX Broker).	
noservdir	A1	Gibt die Service Directory-Option an bzw. zurück, siehe Schlüsselwort-Subparameter DFS des Profilparameters RPC oder Parametermakro NTRPC.	
		Y	Das Service Directory muss nicht vorhanden sein.
		N	Das Service Directory muss vorhanden sein.

3 Geben Sie im aufrufenden Programm auf der Client-Seite das folgende Statement an:

```
CALLNAT 'USR2007N' function nodename servername logon protocol [noservdir]
```

Anmerkung: Das Natural-Subprogramm NATCLTPS in der Library SYSRPC wird nur noch aus Kompatibilitätsgründen gepflegt.

Alternativen Server benutzen

Um Verbindungsfehlschläge zu vermeiden, können Sie mehrere alternative Server für ein Remote-CALLNAT definieren. Wenn Sie solche alternativen Server angeben, geht Natural wie folgt vor:

- Der Client unternimmt einen ersten Versuch, die Verbindung herzustellen.
- Wenn dieser Versuch fehlschlägt, wird anstelle einer Fehlermeldung ein zweiter Versuch unternommen, diesmal jedoch nicht bei demselben Server. Stattdessen wird das Diensteverzeichnis (Service Directory) ab dem aktuellen Eintrag erneut durchsucht, um herauszufinden, ob ein anderer Server verfügbar ist, der den gewünschten Dienst anbietet.
- Wird ein zweiter Eintrag gefunden, versucht Natural, die Verbindung zu diesem Server herzustellen. Wenn der Remote Procedure Call erfolgreich durchgeführt wird, läuft die Client-Anwendung weiter. Der Benutzer merkt nicht, ob die Verbindung zum ersten Server oder zum alternativen Server zum Ergebnis geführt hat.
- Wenn kein weiterer Eintrag gefunden wird oder die Verbindung zu alternativen Servern fehlschlägt, gibt Natural eine entsprechende Fehlermeldung aus.

Um die Verwendung eines alternativen Servers zu ermöglichen:

1. Definieren Sie im Diensteverzeichnis (Service Directory) mehr als einen Server für denselben Dienst.
2. Setzen Sie den Schlüsselwort-Subparameter TRYALT des Profilparameters RPC oder des Parameter-Makros NTRPC auf ON, um die Benutzung eines alternativen Servers zu ermöglichen.

Dieser Parameter kann auch dynamisch für die aktuelle Sitzung mit der **Parameter Maintenance**-Funktion gesetzt werden (siehe *SYSRPC Utility*-Dokumentation).

Interface-Objekte und automatische RPC-Ausführung

Interface-Objekte werden nicht mehr benötigt, wenn die automatische Natural RPC-Ausführung verwendet wird, wie unter *Mit automatischer Natural RPC-Ausführung arbeiten* weiter unten beschrieben.

Die Generierung von Interface-Objekten bietet jedoch den Vorteil, dass die remote ausgeführten CALLNAT(s) kontrolliert werden können und die Fehlerdiagnose erleichtert wird. Sollte ein Remote-Aufruf aufgrund eines falschen CALLNAT-Namens fehlschlagen, hilft die ausgegebene

Natural-Fehlermeldung dabei, die Problemursache sofort zu identifizieren. Ohne ein Interface-Objekt erhalten Sie bei einem fehlerhaften CALLNAT möglicherweise Folgefehler von der Transportschicht oder dem Natural-Server zurück.

Wenn Sie ein Remote-CALLNAT-Statement verwenden wollen, um ein Subprogramm auf einem EntireX-RPC-Server auszuführen, empfehlen wir Ihnen dringend, den Schlüsselwort-Subparameter AUTORPC auf OFF zu setzen und ein Interface-Objekt zu verwenden. Wenn die IDL (Interface Definition Language) des Subprogramms, das Sie auf einem EntireX RPC-Server aufrufen möchten, eine Gruppenstruktur enthält, müssen Sie dieselbe Gruppenstruktur während der Generierung des Interface-Objekts in der SYSRPC-Funktion **Interface Object Generation** definieren oder das Interface-Objekt aus der EntireX IDL-Datei (nur Windows) generieren oder die EntireX Workbench (nur Linux und Windows) verwenden.

Nachfolgend finden Sie Informationen zu:

- Interface-Objekte erstellen
- Mit automatischer Natural RPC-Ausführung arbeiten

Interface-Objekte erstellen

Mit der SYSRPC-Funktion **Interface Object Generation** können Sie Natural Interface-Objekte generieren, die verwendet werden, um das aufrufende Programm des Clients mit einem Subprogramm auf einem Server zu verbinden. Das Interface-Objekt besteht aus einem Parameterdatenbereich (PDA) und der Server-Aufruflogik. Siehe *Interface-Objekte generieren - Allgemeine Aspekte* in der *SYSRPC Utility*-Dokumentation.

Der PDA enthält dieselben Parameter, die in dem CALLNAT-Statement des aufrufenden Programms verwendet werden, und muss in der SYSRPC-Funktion **Interface Object Generation** definiert werden. Wenn bereits ein kompiliertes Natural-Subprogramm mit demselben Namen existiert, wird der von diesem Subprogramm verwendete PDA als Vorbelegung im Bildschirm verwendet. Die Logik für den Serveraufruf wird von der **Interface Object Generation**-Funktion automatisch generiert, nachdem der PDA definiert wurde.

Zur Ausführungszeit muss das Natural-Anwendungsprogramm, das das CALLNAT-Statement und das Interface-Objekt enthält, auf der Client-Seite vorhanden sein. Das Natural-Anwendungsunterprogramm muss auf der Server-Seite vorhanden sein. Das Subprogramm des Schnittstellenobjekts und das Server-Subprogramm müssen denselben Namen haben.

Mit automatischer Natural RPC-Ausführung arbeiten

Dieser Abschnitt gilt nur, wenn Sie keinen EntireX-RPC-Server aufrufen möchten.

Sie müssen keine Natural RPC Interface-Objekte generieren, sondern können mit automatischer Natural RPC-Ausführung arbeiten (d.h. ohne Verwendung von Natural Interface-Objekten). Um mit automatischer Natural RPC-Ausführung zu arbeiten, setzen Sie den Schlüsselwort-Subparameter `AUTORPC` des Profilparameters `RPC` oder des Parameter-Makros `NTRPC` wie folgt:

```
AUTORPC=ON
```

In diesem Fall können Sie bei Ihren Vorbereitungen für die RPC-Nutzung auf die Generierung des Client-Interface-Objekts verzichten. Wenn die automatische Natural RPC-Ausführung aktiviert ist (`AUTORPC=ON`), verhält sich Natural wie folgt:

- Wenn ein Subprogramm lokal nicht gefunden werden kann, versucht Natural, es remote auszuführen (ein Interface-Objekt ist nicht erforderlich),
- der Parameterdatenbereich wird dann dynamisch zur Laufzeit generiert.

Da es Interface-Objekte nur für Client-Programme gibt, hat diese Funktion keine Auswirkungen auf das `CALLNAT`-Programm auf dem Server.

Wenn der Schlüsselwort-Subparameter `AUTORPC` des Profilparameters `RPC` oder des Parameter-Makros `NTRPC` auf `ON` gesetzt ist und ein Natural-Interface-Objekt existiert, wird es trotzdem verwendet.

RPC-Profilparametern während einer Natural-Sitzung abfragen/ändern

Schlüsselwort-Subparameter mit der Utility SYSRPC ändern

Sie können die `SYSRPC` Utility-Funktion **Parameter Maintenance** benutzen, um dynamisch (innerhalb einer Sitzung) einige Schlüsselwort-Subparameter des Profilparameters `RPC` oder des Parameter-Makros `NTRPC`, die im Natural-Parametermodul für die aktuelle Sitzung festgelegt wurden, zu ändern.

Vorsicht: Diese Änderungen bleiben so lange erhalten, wie die Benutzersitzung aktiv ist. Sie gehen verloren, wenn die Sitzung beendet wird.

TIMEOUT-Wert mit API USR2076N abfragen/setzen

Sie können die API USR2076N verwenden, um dynamisch (innerhalb einer Sitzung) den Wert des Schlüsselwort-Subparameters TIMEOUT des Profilparameters RPC oder des Parameter-Makros NTRPC, die im Natural-Parametermodul für die aktuelle Sitzung festgelegt wurden, abzufragen (Funktion: G=Get) oder zu ändern (Funktion: S=Set).

TIMEOUT=*value* gibt die Anzahl der Sekunden an, die der Client auf eine RPC-Server-Antwort warten soll. Wenn diese Zeit überschritten wird, wird der Remote Procedure-Aufruf mit einer entsprechenden Fehlermeldung abgebrochen.

Vorsicht: Diese Änderung bleibt so lange erhalten, wie die Benutzersitzung aktiv ist. Sie geht verloren, wenn die Sitzung beendet wird.

Weitere Informationen siehe Beschreibung im Text-Objekt USR2076T in der Library SYSEXT.

Server-Kommandos ausführen

Aktive Server, die im Dienste-Verzeichnis (Service Directory) definiert wurden (siehe ***RPC-Server-Adressen festlegen***), können mit der SYSRPC Utility-Funktion zur Ausführung von Serverkommandos (**Server Command Execution**) gesteuert werden. Siehe *Server-Kommandos ausführen* in der *SYSRPC Utility*-Dokumentation.

Anmeldung bei einer Server Library

Die Server Library, in der das CALLNAT ausgeführt wird, hängt von der RPC **Logon Option** auf der Client-Seite und einer Reihe von Parametern auf der Server-Seite ab.

Die folgende Tabelle zeigt, welches die relevanten Parameter sind und wie sie die Library-Einstellung beeinflussen:

	Client		**Server**				
	1	2	3	4	5	6	7
	*library-id	RPC LOGON Flag für Server- Eintrag gesetzt?	LOGONRQ set?	Server gestartet mit STACK=	NSC oder natives Natural?	NSC: RPC Logon-Option im Library-Profil	Server *library-id
1	Lib1	nein	nein	logon lib1	Kein Einfluss	N/--	Lib1
2	Lib1	nein	nein	logon lib2	Kein Einfluss	N/--	Lib2

3	Lib1	nein	ja	(Client LOGON flag = NO) *und* (LOGONRQ=YES) ist nicht möglich.			
4	Lib1	ja	Kein Einfluss	Kein Einfluss	NSC	AUTO	Lib1
5	Lib1	ja	Kein Einfluss	Kein Einfluss	NSC	N	Lib1
6	Lib1	ja	Kein Einfluss	Kein Einfluss	Natives Natural	--	Lib1

Erläuterung der Tabellenspalten:

1. Die Library-Kennung (library-id) der Client-Anwendung, von der das CALLNATCALLNAT initiiert wird.
2. Der Wert des RPC LOGON-Flags. Kann für einen ganzen Knoten oder einen Server gesetzt werden.

   Das Flag kann gesetzt werden

   - mit der SYSRPC Utility-Funktion **Service Directory Maintenance**
   - oder mit dem Schlüsselwort-Subparameter DFS des Profilparameters RPC oder des Parameter-Makros NTRPC
   - oder mit der Anwendungsprogrammierschnittstelle USR2007N.
3. Der Schlüsselwort-Subparameter LOGONRQ des Profilparameters RPC oder des Parameter-Makros NTRPC kann beim Start des Servers gesetzt werden.
4. Die Library-Kennung (library-id), auf die der Server beim Start positioniert wird.
5. Läuft der Server unter Natural Security (NSC) (siehe ***Natural RPC mit Natural Security verwenden***) oder nicht?
6. Die Einstellung der Logon-Option in den NSC *Library Profile Items* (*Session options* > *Natural RPC Restrictions*) der NSC-Serveranwendung. Wenn die NSC-Logon-Option auf A (AUTO) gesetzt ist, werden nur Library-Kennung und Benutzerkennung übernommen. Steht sie auf N (Standard), werden die Parameter Library, Benutzerkennung und Passwort ausgewertet.
7. Die Library auf dem Server, in der das CALLNAT-Programm schließlich ausgeführt wird.

Logon-Option benutzen

Die Logon-Option legt fest, in welcher Library das Remote-Subprogramm ausgeführt werden soll. Siehe auch ***Anmeldung bei einer Server Library***.

Anmerkung: Wenn Sie die Logon-Option nicht benutzen, wird das CALLNAT bei der Library ausgeführt, bei der der Server gerade angemeldet ist. Diese Serveranmeldung wird mit dem Natural-Profilparameter STACK=(LOGON *library*) definiert. Der Server sucht in der Library nach den auszuführenden CALLNAT-Statements (und allen zugehörigen Steplibs, die für die Library definiert sind).

Einer Client-Anwendung kann die Ausführung eines Subprogramms in einer anderen Library ermöglicht werden, indem die Logon-Option für dieses Subprogramm gesetzt wird. Dies veranlasst den Client, den Namen seiner aktuellen Library zusammen mit der Logon-Option an den Server zu übergeben. Der Server meldet sich daraufhin bei dieser Library an, sucht in ihr nach dem gewünschten Subprogramm und führt es aus, wenn es gefunden wird. Danach meldet er sich bei der vorherigen Library ab.

Bei einer anderen Library anmelden

Wenn sich der Server bei einer anderen Library als der aktuellen Library des Client anmelden soll, muss der Client die Anwendungsprogrammierschnittstelle USR4008N aufrufen, bevor das Remote-CALLNAT ausgeführt wird. Mit USR4008N gibt der Client einen alternativen Namen einer Library an, bei der sich der Server anmelden soll. Der Name dieser Library wird für alle nachfolgenden Aufrufe von Remote-Subprogrammen verwendet, für die die Logon-Option gilt. Wird für den Library-Namen ein Leerzeichen angegeben, dann wird wieder der Name der aktuellen Client Library verwendet.

Um USR4008N zu verwenden:

1 Kopieren Sie das Subprogramm USR4008N aus der Library SYSEXT in die Library SYSTEM oder in die Steplib Library oder in eine beliebige Anwendung in der Serverumgebung.

2 Geben Sie mit dem DEFINE DATA-Statement die folgenden Parameter an:

Parameter	I/O	Format	Beschreibung	
P-FUNC	I	A01	Funktionscode. Mögliche Werte:	
			P (Put)	Angabe einer neuen Library für die Remote-CALLNAT-Ausführung.
			G (Get)	Abrufen der zuvor angegebenen Library für die Remote-CALLNAT-Ausführung.
P-LIB	I	A8	Library auf dem Server für die Remote-CALLNAT-Ausführung.	

3 Geben Sie im aufrufenden Programm auf der Client-Seite das folgende Statement an:

```
CALLNAT 'USR4008N' P-FUNC P-LIB
```

Anmerkung: Das aufrufende Programm muss ausgeführt werden, bevor der Natural RPC-Client einen Remote-CALLNAT aufruft.

Erforderliche Einstellungen auf der Client-Seite

Um die Logon-Option zu setzen, können Sie entweder die von SYSRPC-Funktion Service Directory Maintenance verwenden (siehe den entsprechenden Abschnitt in der *SYSRPC Utility*-Dokumentation) oder - bei Verwendung eines Standard-Servers - den Schlüsselwort-Subparameter DFS des Profilparameters RPC oder des Parameter-Makros NTRPC oder die Anwendungsprogrammierschnittstelle USR2007N.

Erforderliche Einstellungen auf der Serverseite

Auf der Serverseite ist keine Einstellung erforderlich.

Komprimierung verwenden

Die Komprimierungstypen können sein: 0, 1 oder 2. Interface-Objekte, die mit COMPR=1 oder 2 generiert werden, können zur Reduzierung der Datenübertragungsrate beitragen.

Komprimierungstyp	Beschreibung
COMPR=0	Alle CALLNAT-Parameterwerte werden an den Server gesendet und von diesem zurückgegeben, d.h. es wird keine Kompression durchgeführt.
COMPR=1	Parameter vom Typ M werden an den Server gesendet und von diesem zurückgegeben, während Parameter vom Typ O nur in den Sendepuffer übertragen werden. Parameter vom Typ A werden nur in den Antwortpuffer aufgenommen. Der Antwortpuffer enthält nicht die Formatbeschreibung. Dies ist die Standardeinstellung.
COMPR=2	Wie bei COMP=1, nur dass die Antwortnachricht des Servers noch die Formatbeschreibung der CALLNAT-Parameter enthält. Dies kann nützlich sein, wenn Sie bestimmte Optionen für die Datenkonvertierung durch den EntireX Broker verwenden möchten (weitere Informationen finden Sie in der Beschreibung der *Translation Services* in der *EntireX Broker*-Dokumentation).

Secure Socket Layer verwenden

Der Natural RPC unterstützt Secure Socket Layer (SSL) für die TCP/IP-Kommunikation mit dem EntireX Broker.

Damit der EntireX Broker erkennt, dass die TCP/IP-Kommunikation SSL verwenden soll, müssen Sie eine der folgenden Methoden verwenden:

- Fügen Sie die Zeichenfolge :SSL an den Knotennamen an. Wenn dem Knotennamen bereits die Zeichenfolge :TCP vorangestellt wurde, muss :TCP durch :SSL ersetzt werden.

- Stellen Sie dem Knotennamen die Zeichenfolge `//SSL` voran:

Beispiel:

```
SRVNODE='157.189.160.95:1971:SSL'
```

Bevor Sie über SSL auf einen EntireX Broker zugreifen, müssen Sie zunächst die Anwendungsprogrammierschnittstelle `USR2035N` aufrufen, um die erforderliche SSL-Parameterzeichenfolge zu setzen.

Um `USR2035N` zu verwenden:

1. Kopieren Sie das Subprogramm `USR2035N` aus der Library `SYSEXT` in die Library `SYSTEM` oder in eine Steplib Library oder in eine beliebige Anwendung in der Serverumgebung.
2. Geben Sie mit dem `DEFINE DATA`-Statement die folgenden Parameter an:

Parameter	I/O	Format	Beschreibung
FUNCTION	I	A01	Funktionscode. Mögliche Werte:

			`P` (Put) — Geben Sie einen neuen SSL-Parameter-String an.
			
			Der SSL-Parameter-String wird intern gespeichert und jedes Mal an EntireX übergeben, wenn ein EntireX Broker, der SSL-Kommunikation verwendet, zum ersten Mal angesprochen wird. Sie können verschiedene SSL-Parameter-Strings für mehrere EntireX Broker-Verbindungen verwenden, indem Sie die Anwendungsprogrammierschnittstelle `USR2035N` jedes Mal aufrufen, bevor Sie das erste Mal auf den EntireX Broker zugreifen.
			
			Beispiel:
			
			`FUNCTION := 'P'`
			`SSLPARMS := ↩`
			`'TRUST_STORE=FILE://DDN:CACERT&VERIFY_SERVER=N'`
			`CALLNAT 'USR2035N' USING FUNCTION SSLPARMS ↩`
			
			Um SSL-Parameter im Falle eines Natural RPC Servers zu setzen, legen Sie den Namen des aufrufenden Programms beim Start des Servers auf den Natural-Stack.
			
			Beispiel:

Parameter	I/O	Format	Beschreibung	
				`STACK=(LOGON server-library;set-SSL-parms)` Dabei ist *set-SSL-parms* ein Natural-Programm, das die Anwendungsprogrammierschnittstelle USR2035N aufruft, um den SSL-Parameter-String zu setzen.
			G (Get)	Abrufen der zuvor angegebenen SSL-Parameterzeichenfolge. Der zuvor gesetzte SSL-Parameterstring wird an den Aufrufer zurückgegeben. Weitere Informationen über den SSL-Parameter-String finden Sie in der *EntireX*-Dokumentation.
SSLPARMS	I	A128	SSL-Parameter-String, wie vom EntireX-Broker benötigt.	

3 Geben Sie im aufrufenden Programm auf der Client-Seite das folgende Statement an:

```
CALLNAT 'USR2035N' FUNCTION SSLPARMS
```

Den Status einer RPC-Sitzung überwachen

Folgende Themen werden behandelt:

- Programm RPCERR verwenden
- Subprogramm RPCINFO verwenden
- Server-Trace-Funktion benutzen
- Trace-Datei definieren

Programm RPCERR verwenden

Sie können das Programm RPCERR über die Kommandozeile ausführen oder es mit einem FETCH-Statement aus einem Natural-Programm heraus aufrufen. RPCERR zeigt die folgenden Informationen an:

- Die letzte Natural-Fehlernummer und -meldung, wenn sie mit RPC zusammenhängt.
- Die letzte EntireX-Broker-Meldung im Zusammenhang mit diesem Fehler.
- The last EntireX RPC server error message if the Natural error error number is related to the EntireX RPC server error

  Die letzte EntireX RPC-Server-Fehlermeldung, wenn die Natural-Fehlernummer mit dem EntireX RPC-Server-Fehler zusammenhängt.

Zusätzlich können der Knoten- und Servername des letzten EntireX Broker-Aufrufs abgerufen werden.

Beispiel für eine RPC-Fehleranzeige: RPCERROR

```
Natural error number: NAT6972
Natural error text  :
Directory error on Client, reason 3.

RPC error information:
No additional information available.

Server Node:                                          Library: SYSTEM
Server Name:                                          Program: NATCLT3
                                                      Line No: 1010        ↩
```

Subprogramm RPCINFO verwenden

Sie können das Subprogramm `RPCINFO` in Ihrem Anwendungsprogramm verwenden, um Informationen über den Zustand der aktuellen RPC-Sitzung abzurufen. Dies ermöglicht Ihnen auch eine angemessenere Fehlerbehandlung, indem Sie auf eine bestimmte Fehlerklasse reagieren.

Das Subprogramm `RPCINFO` ist in der Library `SYSTEM` enthalten und kann von jeder Benutzeranwendung aufgerufen werden.

Ein Beispielprogramm `TESTINFO` ist in der Library `SYSRPC` zusammen mit dem Parameterdatenbereich `RPCINFOL` für den Aufruf von `RPCINFO` enthalten.

Beispiel:

```
DEFINE DATA LOCAL USING RPCINFOL
  LOCAL
  1 PARM   (A1)
  1 TEXT   (A80)
  1 REDEFINE TEXT
    2 CLASS  (A4)
    2 REASON (A4)
END-DEFINE
...
OPEN CONVERSATION USING SUBPROGRAM 'APPLSUB1'
  CALLNAT 'APPLSUB1' PARM
CLOSE CONVERSATION *CONVID
...
ON ERROR
   CALLNAT 'RPCINFO' SERVER-PARMS CLIENT-PARMS
   ASSIGN TEXT=C-ERROR-TEXT
   DISPLAY CLASS REASON
END-ERROR
...
END
```

Parameter von RPC Info

RPCINFO hat die folgenden Parameter, die im Parameterdatenbereich RPCINFOL bereitgestellt werden:

Parameter	Format	Beschreibung
SERVER-PARMS		Enthält Informationen über die Natural-Sitzung, wenn Sie als Server fungiert. Die SERVER-PARMS gelten nur, wenn Sie RPCINFO remote auf einem RPC-Server ausführen.
S-BIKE	A1	Verwendetes Transportprotokoll. Möglicher Wert:
		B EntireX Broker
S-NODE	A32	Der Knotenname des Servers.
S-NAME	A32	Der Name des Servers.
S-ERROR-TEXT	A80	Enthält den von der Transportschicht zurückgegebenen Nachrichtentext.
S-CON-ID	I4	Aktuelle Kennung der Konversation. Beachten Sie, dass dies die physische Kennung von EntireX Broker ist, nicht die logische Natural-Kennung. Dieser Parameter enthält immer einen Wert, da EntireX Broker Kennungen sowohl für konversationelle als auch für nicht-konversationelle Anrufe generiert. Wenn die physische Konversationskennung entweder nicht numerisch oder größer als I4 ist, wird ein -1 zurückgegeben.
S-CON-OPEN	L	Gibt an, ob eine offene Konversation besteht. Dieser Parameter enthält den Wert TRUE, wenn eine Konversation offen ist, andernfalls enthält er FALSE.
CLIENT-PARMS		Enthält Informationen über die Natural-Sitzung, wenn diese als Client fungiert.
C-BIKE	A1	Verwendetes Transportprotokoll. Möglicher Wert:
		B EntireX Broker
C-NODE	A32	Der Knotenname des zuvor angesprochenen Servers.
C-NAME	A32	Der Name des zuvor angesprochenen Servers.
C-ERROR-TEXT	A80	A80 Enthält den von der Transportschicht zurückgegebenen Nachrichtentext.
C-CON-ID	I4	Konversationskennung des letzten Serveraufrufs. Beachten Sie, dass dies die physische Kennung von EntireX Broker ist, nicht die logische Natural-Kennung. Wenn keine Konversation offen ist, ist der Wert dieses Parameters kleiner oder gleich 0. Wenn die physische

Parameter	Format	Beschreibung
		Konversationskennung entweder nicht numerisch oder größer als I4 ist, wird ein `-1` zurückgegeben.
`C-CON-OPEN`	L	Gibt an, ob eine offene Konversation besteht. Dieser Parameter enthält den Wert `TRUE`, wenn eine Konversation offen ist, andernfalls enthält er `FALSE`.
`C-ENTIREX-RPC-ERROR-MESSAGE`	A	Enthält den von einem EntireX-RPC-Server zurückgegebenen Nachrichtentext.

Server-Trace-Funktion benutzen

Natural RPC enthält eine Trace-Funktion, mit der Sie Serveraktivitäten überwachen und mögliche Fehlersituationen aufspüren können.

Server-Trace-Funktion aktivieren/deaktivieren

Um die Server-Trace-Funktion zu aktivieren/deaktivieren:

- Starten Sie den Server mit der Option

  ```
  TRACE=n
  ```

  Der ganzzahlige Wert *n* steht für den gewünschten Trace-Level, d.h. für die Detailtiefe, mit der Ihr Server protokolliert werden soll.

  Die folgenden Werte sind möglich:

Wert	Trace Level
0	Es wird kein Trace durchgeführt (Standardeinstellung).
1	Alle Client-Anforderungen und die entsprechenden Server-Rückmeldungen werden aufgezeichnet und dokumentiert.
2	Alle Client-Anforderungen und die entsprechenden Server-Rückmeldungen werden aufgezeichnet und dokumentiert. Zusätzlich werden alle RPC-Daten in die Trace-Datei geschrieben.

  Die RPC-Trace-Funktion schreibt die Trace-Daten in den Natural Report Nummer 10.

  Im Falle eines Konvertierungsfehlers, der mit der Natural-Fehlernummer NAT6974 und den Ursachencodes 2 und 3 gemeldet wird, wird die Position der fehlerhaften Daten im Puffer angegeben.

Unterstützung von TS=ON für RPC Server Trace

Die folgenden Informationen gelten nur für Großrechner-Umgebungen:

Alle Meldungen im Natural RPC Server Trace werden in Großbuchstaben umgewandelt, wenn `TS=ON` in der Natural RPC Server-Sitzung angegeben ist. Die Trace der Daten vom/zum Client wird durch `TS=ON` nicht beeinflusst und bleibt unverändert.

Trace-Datei definieren

Die Definition der Trace-Datei hängt von der Umgebung ab:

- Trace-Dateien für Großrechner-Umgebungen - Allgemeine Informationen
- Trace-Datei im z/OS-Batch-Modus
- Trace-Datei unter CICS
- Trace-Datei im z/VSE Batch-Modus
- Trace-Datei im BS2000-Batch-Modus
- Trace-Dateien für UNIX- und OpenVMS-Umgebungen
- Trace-Dateien für Windows

Trace-Dateien für Großrechner-Umgebungen - Allgemeine Informationen

Definieren Sie auf dem Großrechner die für Ihre Umgebung geeignete Trace-Datei, siehe auch `NTPRINT`-Makro (in der *Parameter-Referenz*-Dokumentation).

Trace-Datei im z/OS-Batch-Modus

a) Bei Ausführung eines Servers als einzelne Task

Weisen Sie im Server-Startjob dem zusätzlichen Natural-Report `CMPRT10` einen z/OS-Dataset zu.

Beispiel:

```
//NATRPC   JOB  CLASS=K,MSGCLASS=X
//NATSTEP  EXEC PGM=NATOS
//STEPLIB  DD   DISP=SHR,DSN=SAG.NAT.LOAD
//         DD   DISP=SHR,DSN=SAG.EXX.LOAD
//CMPRMIN  DD   *
IM=D,MADIO=0,MT=0,OBJIN=R,AUTO=OFF,MAXCL=0,ID=',',INTENS=1,
PRINT=((10),AM=STD)
/*
//SYSUDUMP DD   SYSOUT=X
//CMPRT10  DD   SYSOUT=X
//CMPRINT  DD   SYSOUT=X
/*
```

b) Bei Ausführung eines Servers mit Replikaten

1. Setzen Sie den Schlüsselwort-Subparameter `NTASKS` des Profilparameters `RPC` oder des Parameter-Makros `NTRPC` auf einen Wert größer als `1`.
2. Weisen Sie `CMPRMIN` einem Dataset mit `DISP=SHR` oder einen Stern (`*`) zu.

3. Da jede Task auf ein separates CMPRINT-Dataset schreibt, definieren Sie die folgenden DD-Kartennamen:

CMPRINT	für die Haupt-Task
CMPRINT1 bis CMPRINT9	für die ersten neun Subtasks
CMPRIN10 bis CMPRIN*nn*	für die nächsten zweistelligen Nummern der Subtasks, *nn*=NTASKS-1.

4. Wenn der Schlüsselwort-Subparameter TRACE des Profilparameters RPC oder des Parameter-Makros NTRPC gesetzt ist, schreibt die Trace-Funktion auf Drucker 10.

   Sie müssen die folgenden DD-Kartennamen definieren:

CMPRT10	für die Haupt-Task
CMPRT101 bis CMPRT1*nn*	für alle Subtasks, *nn*=NTASKS-1

Beispiel:

```
//NATRPC   JOB  CLASS=K,MSGCLASS=X
//NATSTEP  EXEC PGM=NATOS,REGION=8M
//steplib  DD   DISP=SHR,DSN=SAG.NAT.LOAD
//         DD   DISP=SHR,DSN=SAG.EXX.LOAD
//CMPRMIN  DD   *
IM=D,MADIO=0,MT=0,OBJIN=R,AUTO=OFF,MAXCL=0,ID=',',INTENS=1,
PRINT=((10),AM=STD)
/*
//SYSUDUMP DD   SYSOUT=X
//CMPRT10  DD   SYSOUT=X
//CMPRT101 DD   SYSOUT=X
//CMPRT102 DD   SYSOUT=X
//CMPRT103 DD   SYSOUT=X
//CMPRINT  DD   SYSOUT=X
//CMPRINT1 DD   SYSOUT=X
//CMPRINT2 DD   SYSOUT=X
//CMPRINT3 DD   SYSOUT=X
/*
```

Trace-Datei unter CICS

Weisen Sie unter CICS die Druckdatei 10 einer extra-partitionierten CICS-Warteschlange für transiente Daten zu.

Beispiele:

Dynamische Natural-Profildefinition:

```
PRINT=((10),AM=CICS,DEST=RPCT,TYPE=TD)
```

CICS-Definition:

```
RPCTRAC DFHDCT TYPE=SDSCI,                        X
               BLKSIZE=136,                       X
               BUFNO=1,                           X
               DSCNAME=RPCTRACE,                  X
               RECFORM=VARUNB,                    X
               RECSIZE=132,                       X
               TYPEFLE=OUTPUT
        SPACE
RPCT    DFHDCT TYPE=EXTRA,                        X
               DSCNAME=RPCTRACE,                  X
               DESTID=RPCT,                       X
               OPEN=INITIAL
```

CICS Startup JCL:

```
RPCTRACE DD    SYSOUT=*
```

Trace-Datei im z/VSE Batch-Modus

Im z/VSE-Batch-Modus weisen Sie der Druckernummer 10 eine Trace-Datei zu.

Beispiel:

```
// LIBDEF PHASE,SEARCH=(SAGLIB.NATvrs,SAGLIB.ETBvrs),TEMP
// ASSGN SYS000,READER
// ASSGN SYSLST,FEE
// ASSGN SYS050,FEF
// EXEC NATVSE,SIZE=AUTO,PARM='SYSRDR'
IM=D,MADIO=0,MT=0,OBJIN=R,AUTO=OFF,MAXCL=0,ID=',',INTENS=1,
PRINT=((10),AM=STD,SYSNR=50)
/*
```

wobei *vrs* für die jeweilige Produktversion steht.

Trace-Datei im BS2000-Batch-Modus

Weisen Sie im BS2000-Batch-Modus eine Trace-Datei der Druckernummer 10 zu.

Beispiel:

```
/.NATRPC      LOGON
/             SYSFILE   SYSOUT=output-file
/             SYSFILE   SYSDTA=(SYSCMD)
/             SYSFILE   SYSIPT=(SYSCMD)
/             FILE trace-file,LINK=P10,OPEN=EXTEND       */server trace file
/             STEP
/             SETSW      ON=2
/             EXEC NATBS2
MADIO=0,IM=D,ID=',',PRINT=((10),AM=STD)
```

Trace-Dateien für UNIX- und OpenVMS-Umgebungen

Es wird empfohlen, für jeden Server einen anderen Dateinamen (d. h. eine andere NATPARM-Parameterdatei) zu verwenden, damit Sie jeden Server einzeln protokollieren können. Die Trace-Datei wird in der NATPARM-Parameterdatei des Natural-Servers definiert:

1. **Report-Zuweisungen**
   Weisen Sie Ihrem Report Nummer 10 das logische Gerät (Device) LPT10 zu.
2. **Device-Parameter- Zuweisungen**
   Anstatt eine physische Druckerspezifikation für LPT10 zu wählen, geben Sie einen Dateinamen an, der den Namen Ihrer Trace-Datei darstellt.

Beispiel für UNIX:

```
/bin/sh -c cat>>/filename
```

wobei *filename* den Namen der Trace-Datei darstellt.

Beispiel für OpenVMS:

```
nattmp:filename
```

wobei *filename* den Namen der Trace-Datei darstellt.

Trace-Dateien für Windows

Es wird empfohlen, für jeden Server einen anderen Dateinamen (d.h. eine andere NATPARM-Parameterdatei) zu verwenden, damit Sie sie einzeln protokollieren können. Die Trace-Datei wird in der NATPARM-Parameterdatei des Natural-Servers definiert (siehe *Device/Report Assignments* in *Configuration Utility* in der *Natural for Windows*-Dokumentation):

1. **Reports**

   Weisen Sie Ihrer Reportnummer 10 das logische Gerät (Device) LPT10 zu.

2. **Devices**

   Anstatt eine physische Druckerspezifikation für LPT10 zu wählen, geben Sie einen Dateinamen an, der dem Namen Ihrer Trace-Datei entspricht. Standardmäßig werden alte Trace-Dateien gelöscht, wenn eine neue Datei mit demselben Namen angelegt wird.

   Wenn Sie das neue Protokoll an das bestehende anhängen wollen, geben Sie an:

   `>>filename`

Laufzeiteinstellungen eines Servers abrufen

Die Natural-Anwendungsprogrammierschnittstelle (API) `USR4010N` ermöglicht es Ihnen, die Laufzeit-Einstellungen eines Servers abzurufen:

- die Systemdatei-Zuweisungen für `FUSER`, `FNAT` und `FSEC`,
- die Steplib-Kette.

Um `USR4010N` zu verwenden:

1 Server-Umgebung:

Kopieren Sie das Subprogramm `USR4010N` aus der Library `SYSEXT` in die Library `SYSTEM` oder in die Steplib Library oder in eine beliebige Anwendung in der Serverumgebung.

2 Client-Umgebung:

Geben Sie in einem `DEFINE DATA`-Statement die folgenden Parameter als Redefinition von `USR4010-PARM` (A300) an:

Parameter	Format	Beschreibung
`FUSER-DBID`	N5	Datenbankkennung der Systemdatei `FUSER`.
`FUSER-FNR`	N5	Dateinummer der Systemdatei `FUSER`.
`FNAT-DBID`	N5	Datenbankkennung der Systemdatei `FNAT`.
`FNAT-FNR`	N5	Dateinummer der Systemdatei `FNAT`.
`FSEC-DBID`	N5	Datenbankkennung der Systemdatei `FSEC`.
`FSEC-FNR`	N5	Dateinummer der Systemdatei `FSEC`.
`STEP-NAME`	A8/15	Name der Steplib.
`STEP-DBID`	N5/15	Datenbankkennung der Steplib.
`STEP-FNR`	N5/15	Dateinummer der Steplib.

3 Geben Sie im aufrufenden Programm auf der Client-Seite das folgende Statement an:

```
CALLNAT 'USR4010N' USR4010-PARM
```

Siehe auch *Syntax-Beschreibung* des CALLNAT-Statement.

4 Wenn der Schlüsselwort-Subparameter AUTORPC des Profilparameters RPC oder des Parameter-Makros NTRPC auf OFF gesetzt ist, kopieren Sie die Schnittstelle USR4010X unter dem Namen USR4010N in die Client-Umgebung.

Wenn der Schlüsselwort-Subparameter AUTORPC auf ON gesetzt ist, darf die API in der Client-Umgebung nicht verfügbar sein, um zu verhindern, dass die Schnittstelle lokal aufgerufen wird.

Beim Aufruf von USR4010N werden die Werte der oben angegebenen Parameter in der Feldgruppe USR4010-PARM ausgegeben.

Parameter für EntireX setzen/abfragen

Mit der Anwendungsprogrammierschnittstelle (API) USR4009N können Sie den/die EntireX-Parameter, die aktuell vom Natural RPC unterstützt werden, setzen oder abrufen:

- Komprimierungsstufe (Compression Level)

Um USR4009N zu verwenden:

1 Kopieren Sie das Subprogramm USR4009N aus der Library SYSEXT in die Library SYSTEM oder in die Steplib Library oder in eine beliebige Anwendung in der Serverumgebung.

2 Geben Sie mit einem DEFINE DATA-Statement die folgenden Parameter an:

Parameter	Format	I/O	Beschreibung	
FUNCTION	A01	I	Funktion. Mögliche Werte:	
			G (Get)	Die bereits eingestellten Werte für die EntireX-Parameter werden zurückgegeben. Wenn in der Natural-Sitzung noch kein PUT aufgerufen wurde, sind alle Werte Null oder leer.
			P (Put)	Die für die EntireX-Parameter angegebenen Werte werden gespeichert und bei allen nachfolgenden Aufrufen von EntireX verwendet.
ENVIRONMENT	A01	I	Umgebung. Mögliche Werte:	
			S	Server
			C	Client

Parameter	Format	I/O	Beschreibung	
			B	Beide
COMPRESSLEVEL	A01	I/O	Komprimierungsstufe.	
RESERVED	I01	I/O	Reserviert für künftige Verwendung.	
ACIVERS	B02	O	Verwendete API-Version.	
RC	B01	O	Rückgabecode, sofern nicht gleich Null. Enthält die API-Version, die zum Setzen des angeforderten Parameters erforderlich ist:	
			0	Funktion erfolgreich.
			7	Komprimierungsstufe erfordert API-Version 7.

3 Die Schnittstelle kann auf zwei Arten aufgerufen werden:

1. Aus einem Programm heraus:

```
CALLNAT 'USR4009N' FUNCTION ENVIRONMENT
                   COMPRESSLEVEL
                   RESERVED
                   ACIVERS RC
```

2. Von der Kommando-Eingabeaufforderung aus oder durch Verwendung des Statements STACK mit Werten für die oben genannten Parameter.

   Beispiele:

```
USR4009P P,C,COMPRESSLEVEL=6
USR4009P P,C,6
```

Im Kommando-Modus können Sie die Notation *keyword=value* verwenden, um nur eine Teilmenge der EntireX-Parameter zu setzen. Die Werte für Parameter, die nicht referenziert werden, bleiben unverändert.

Anmerkungen:

1. Die Anforderung wird abgelehnt und es werden keine Werte gespeichert, wenn die von der aktuellen Natural-Sitzung verwendete API-Version nicht hoch genug ist, um den angeforderten EntireX-Parameter zu unterstützen. In diesem Fall enthält der Parameter RC die erforderliche API-Version.
2. Die EntireX-Parameter werden nur vom Natural RPC beachtet.

Message ID und Correlation ID von EntireX verwalten

Mit der Anwendungsprogrammierschnittstelle (API) USR8225N können Sie die EntireX-Parameter MESSAGE_ID und CORRELATION_ID in einer Natural RPC Client- oder Server-Umgebung verwalten.

Um USR8225N zu verwenden:

1 Kopieren Sie das Subprogramm USR8225N aus der Library SYSEXT in die Library SYSTEM oder in die Steplib Library oder in eine beliebige Anwendung in der Serverumgebung.

2 Geben Sie mit einem DEFINE DATA-Statement die folgenden Parameter an:

Parameter	Format	I/O	Beschreibung
FUNCTION	A08	I	Funktionscode: ■ GENERATE ■ RETRIEVE
ENVIRONMENT	A06	I	Umgebungseinstellung: ■ CLIENT ■ SERVER
MESSAGE_ID	A64	O	Nachrichtenkennung.
CORRELATION_ID	A64	O	Korrelationskennung.
RC	N04	O	Rückgabewert: ■ 0 - Keine Fehler ■ 1 - Ungültiger Funktionscode ■ 2 - Ungültige Umgebung ■ 3 - Von Natural RPC nicht unterstützt

Behandlung von Fehlern

Behandlung von Remote-Fehlern

Jeder Natural-Fehler auf der Server-Seite wird wie folgt an den Client zurückgegeben:

- Der Natural RPC überträgt die entsprechende Fehlernummer in die Systemvariable `*ERROR-NR`.
- Natural reagiert so, als ob der Fehler lokal aufgetreten wäre.

Anmerkung: Wenn der Schlüsselwort-Subparameter `AUTORPC` des Profilparameters `RPC` oder des Parameter-Makros `NTRPC` auf `ON` gesetzt ist und ein Subprogramm in der lokalen Umgebung nicht gefunden werden kann, interpretiert Natural dies als einen Remote Procedure-Aufruf. Es versucht dann, dieses Subprogramm im Diensteverzeichnis (Service Directory) zu finden. Wenn es dort nicht gefunden wird, wird ein NAT6972-Fehler ausgegeben. Infolgedessen wird kein NAT0082-Fehler ausgegeben, wenn ein Subprogramm nicht gefunden werden kann.

Siehe auch *Programm RPCERR verwenden*.

Vermeidung der Fehlermeldung NAT3009 vom Serverprogramm

Wenn ein Server-Anwendungsprogramm über einen längeren Zeitraum keinen Datenbankaufruf tätigt, kann der nächste Datenbankaufruf eine Fehlermeldung NAT3009 zurückgeben.

Um dieses Problem zu vermeiden, gibt es den optionalen User-Exit `NATRPC39`. Dieser Exit wird in den folgenden Fällen aufgerufen:

1. Nach einem Ping-Kommando.
2. Bei Kommunikation über den EntireX Broker: jedes Mal, wenn die `SERVER-NONACT`-Zeit überschritten wird.

Um den User Exit `NATRPC39` zu aktivieren:

1 Kopieren Sie den Beispiel-Exit `NATRPC39` aus der Library `SYSRPC` in die Library `SYSTEM` in der Systemdatei `FUSER`.

Die Steplib-Verkettung der Library, bei der der Server gerade angemeldet ist, wird nicht ausgewertet.

2 Passen Sie die Datenbankkennung, die dem Feld ACB-RSP zugeordnet ist, an Ihre Bedürfnisse an.

3 Fügen Sie Ihrem NATRPC39 zusätzliche CALL 'CMADA'-Statements hinzu, die auf weitere Datenbankkennungen verweisen, wenn mehr als eine Adabas-Datenbank beteiligt ist.

User Exit NATRPC01

Der User-Exit NATRPC01 wird aufgerufen, wenn ein Natural-Fehler aufgetreten ist, und zwar nachdem der Fehler von der Natural RPC-Laufzeit behandelt wurde und unmittelbar bevor die Rückmeldung an den Client zurückgesendet wird. Das heißt, der Exit wird an der gleichen logischen Stelle wie eine Fehlertransaktion aufgerufen, nämlich am Ende der Natural-Fehlerbehandlung, nachdem alle ON ERROR-Statement-Blöcke abgearbeitet worden sind.

Im Gegensatz zu einer Fehlertransaktion wird dieser Exit mit einem CALLNAT-Statement aufgerufen und muss daher ein Subprogramm sein, das zu seinem Aufrufer zurückmelden muss.

Die Schnittstelle zu diesem Exit ist ähnlich wie die Schnittstelle einer Fehlertransaktion. Darüber hinaus kann der Exit bis zu 10 Zeilen an Informationen zurückgeben, die von der Natural RPC-Laufzeit protokolliert werden. Nur Zeilen, die mit einem Nicht-Leerzeichen beginnen, werden aufgezeichnet.

Ein Beispiel-User Exit NATRPC01 befindet sich in der Library SYSRPC.

Wichtige Hinweise:

1. NATRPC01 muss sich in der Library SYSTEM in der Systemdatei FUSER befinden. Die Steplib-Verkettung der Library, bei der der Server gerade angemeldet ist, wird *nicht* ausgewertet.
2. The DEFINE DATA PARAMETER statement block must not be changed.

Benutzer-Exits vor und nach der Ausführung von Diensten

Um Administratoren mehr Kontrolle über die Ausführung von Diensten (Remote CALLNATs) zu geben, werden auf der Natural RPC Server-Seite zwei optionale User Exits aufgerufen.

User Exit	Zweck
NATRPC02	Der optionale Before-Service-Exit NATRPC02 wird unmittelbar vor der Ausführung des Dienstes aufgerufen. Zu diesem Zeitpunkt hat die Anforderung bereits alle Sicherheitsprüfungen durchlaufen und die Daten sind **„unmarshalled“**.
NATRPC03	Der optionale After-Service-Exit NATRPC03 wird unmittelbar nach der erfolgreichen Rückkehr von dem Dienst aufgerufen. Zu diesem Zeitpunkt sind die Daten noch nicht **„marshalled“**. Der Exit wird nicht aufgerufen, wenn ein unbehandelter Fehler aufgetreten ist

Diese Exits sind unabhängig voneinander und können separat verwendet werden.

Für beide Exits gelten die folgenden Regeln:

- Der Exit muss sich in der Library SYSTEM in der Systemdatei FUSER befinden.

  Wird der Exit beim Starten des Natural RPC Servers gefunden, wird eine Meldung in den Natural RPC Server Trace geschrieben, um die Aktivierung des Exits anzuzeigen. Der Exit wird anschließend ohne Vorbedingungen aufgerufen. Wird der Exit während der Lebensdauer der Server-Sitzung entfernt, tritt ein permanenter NAT0082-Fehler auf.

  Wird der Exit beim Starten des Natural RPC Servers nicht gefunden, wird er während der Lebensdauer der Serversitzung nie aufgerufen. Der Exit kann nicht dynamisch aktiviert werden.
- Der Exit muss vom Anwender als Subprogramm implementiert werden. Der Exit wird mit einer einzigen dynamischen Variable als Parameter aufgerufen. Der Inhalt der dynamischen Variablen ist der acht Zeichen lange Name des Remote-Subprogramms.

  Die Verwendung der dynamischen Variable ermöglicht es, künftige Erweiterungen der übergebenen Informationen zu implementieren, ohne dass es zu Problemen mit bestehenden, vom Anwender geschriebenen Exits kommt.
- Der Exit wird auch innerhalb einer Konversation aufgerufen.
- Der Natural RPC Server fängt unbehandelte Fehler im Exit nicht ab. Tritt ein unbehandelter Fehler im Exit auf, wird der Fehler an den Client weitergegeben.

Die Exits können für Auditing- oder Tracing-Zwecke verwendet werden. NATRPC02 kann auch für zusätzliche Sicherheitsüberprüfungen verwendet werden.

Beispiel für NATRPC02:

```
DEFINE DATA PARAMETER
1 SUBPROGRAM (A8) BY VALUE
END-DEFINE
IF *USER <> 'DBA' AND SUBPROGRAM = 'PRIVATE'
  *ERROR-NR := 999
END-IF
END
```

9 Verwendung des Natural RPC im konversationellen Modus

Eine Konversation eröffnen

Um eine Konversation zu eröffnen:

1 Geben Sie auf der Client-Seite ein `OPEN CONVERSATION`-Statement an.

2 Geben Sie in dem `OPEN CONVERSATION`-Statement eine Liste von Diensten (Services, die in Form von Subprogrammen realisiert sind) als Teilnehmer dieser Konversation an.

Das `OPEN CONVERSATION`-Statement weist der Systemvariablen `*CONVID` ein eindeutige Konversationskennung zu.

Es kann mehr als eine Konversation parallel geöffnet sein. Wenn Subprogramme sich gegenseitig stören, sind die Anwendungsprogramme dafür zuständig, die verschiedenen Konversationen zu verwalten, indem sie die entsprechende Systemvariable `*CONVID` setzen, die vom `CALLNAT`-Statement ausgewertet wird.

- Ist das Subprogramm Teil der aktuellen Konversation (durch `*CONVID` referenziert), wird es in der Server-Task ausgeführt, die ausschließlich für diese Konversation reserviert ist.
- Ist es nicht Teilnehmer der aktuellen Konversation, wird es in einer anderen Server-Task ausgeführt. Dies gilt auch für unterschiedliche Konversationen.

Eine Konversation kann auf jeder Programmebene geöffnet werden und `CALLNAT`-Statements innerhalb dieser Konversation können auf jeder anderen Programmebene darunter oder darüber ausgeführt werden.

Es ist möglich, eine Client-Konversation innerhalb eines entfernt (remote) auf einem Server ausgeführten `CALLNAT` zu öffnen, so dass der Server als Agent fungiert. Da der Client nur seine eigenen Konversationen kontrolliert und nicht die des Servers, muss der Anwendungsprogrammierer sicherstellen, dass die Konversation auf dem Server ordnungsgemäß beendet wird, bevor der Haupt-Client geschlossen wird.

Zusätzliche Einschränkungen

Der konversationelle RPC kann noch lokal getestet werden. Um das Verhalten identisch zu halten, wenn Sie einen konversationellen `CALLNAT` entfernt (remote) oder lokal ausführen, gelten die folgenden zusätzlichen Einschränkungen:

- Ein `CLOSE CONVERSATION` ist nicht möglich innerhalb eines Objekts, das gerade als Teilnehmer an dieser Konversation läuft. Dies entspricht der Einschränkung, dass es nicht möglich ist, eine Konversation aus einem entfernt (remote) laufenden Programm heraus zu beenden.
- Es ist nicht möglich, ein konversationelles `CALLNAT`, das Mitglied der Konversation ist, aus einem anderen (oder demselben) Teilnehmer an dieser Konversation heraus auszuführen. Dies ent-

spricht der Einschränkung, dass es nicht möglich ist, ein konversationelles CALLNAT, das Mitglied der Konversation des Clients ist, von einem Server-Subprogramm aus auszuführen.

- Es wird nicht empfohlen, eine Konversation aus einem Subprogramm einer anderen Konversation heraus zu eröffnen.

Eine Konversation beenden

Um eine Konversation zu beenden:

- Geben Sie ein CLOSE CONVERSATION-Statement auf der Client-Seite an.

Damit kann der Client eine bestimmte Konversation oder alle Konversationen schließen. Alle Kontextvariablen der beendeten Konversation werden dann freigegeben und die Server-Task ist wieder für einen anderen Client verfügbar.

Wenn Sie Natural beenden, schließen Sie damit implizit alle Konversationen.

Wenn ein Server ein CLOSE CONVERSATION-Anforderung erhält, gibt er ein CLOSE CONVERSATION ALL-Statement aus, so dass alle Konversationen, die der Server (als Agent) geöffnet haben könnte, ebenfalls geschlossen werden.

Eine Konversation mit implizitem BACKOUT TRANSACTION (Rollback) schließen

Wenn ein CLOSE CONVERSATION-Statement ausgeführt wird, wird standardmäßig die Rollback-Option zusammen mit dem CLOSE CONVERSATION-Statement an den Server gesendet. Dies führt am Ende der Konversationsverarbeitung zu einem impliziten BACKOUT TRANSACTION auf der Server-Seite.

Eine Konversation mit implizitem END TRANSACTION (Commit) beenden

Sie können die in der Library SYSEXT verfügbare Anwendungsprogrammierschnittstelle USR2032N verwenden, um ein implizites END TRANSACTION auf der Serverseite auszulösen.

Die Anwendungsprogrammierschnittstelle muss vor der Ausführung des nächsten CLOSE CONVERSATION-Statement aufgerufen werden. Dies hat zur Folge, dass die Commit-Option zusammen mit dem CLOSE CONVERSATION-Statement an den Server gesendet wird und der Server am Ende der Konversationsverarbeitung ein END TRANSACTION-Statement ausführt.

Die Commit-Option gilt für das nächste CLOSE CONVERSATION-Statement, das von der Client-Anwendung ausgeführt wird. Nachdem die Konversation(en) beendet wurde(n), wird wieder die Standardoption verwendet. Das bedeutet, dass das folgenden CLOSE CONVERSATION-Statement wieder zu einem BACKOUT TRANSACTION-Statement führt.

Einen Konversationskontext definieren

Während einer Konversation können die Subprogramme, die Mitglieder dieser Konversation sind, einen gemeinsamen Kontextbereich auf diesem Server nutzen.

Dazu deklarieren Sie in jedem der betroffenen Unterprogramme einen Datenbereich mit dem `DEFINE DATA CONTEXT`-Statement.

Die Subprogramme, die einen Kontextbereich verwenden, verhalten sich so, als ob die Konversation lokal oder remote wäre. Das `DEFINE DATA CONTEXT`-Statement entspricht weitgehend dem `DEFINE DATA INDEPENDENT`-Statement. Alle Regeln, die für die Definition von AIV-Variablen gelten, gelten auch für Kontextvariablen, mit der Ausnahme, dass einer Kontextvariablen kein Pluszeichen (+) vorangestellt werden muss.

Der Compiler prüft keine Format-/Längendefinition, da dies voraussetzt, dass die Variablen durch Ausführen eines Programms erstellt werden, das alle Definitionen für diese Anwendung enthält (wie bei AIVs üblich). Bei Kontextvariablen hat dies keinen Sinn, da eine Library, die RPC-Service-Routinen enthält, normalerweise nicht anwendungsabhängig ist.

Im Gegensatz zu AIVs werden die Kontextvariablen des Aufrufers nicht über `CALLNAT`-Grenzen hinweg übergeben. Kontextvariablen werden durch ihren Namen und die Kontextkennung, für die sie gelten, referenziert. Eine Kontextvariable wird von allen Service-Routinen, die sich innerhalb einer Konversation auf denselben Variablennamen beziehen, gemeinsam genutzt. Daher hat jede Konversation ihren eigenen Satz an Kontextvariablen. Kontextvariablen können nicht von verschiedenen Konversationen gemeinsam genutzt werden, auch wenn sie denselben Variablennamen haben.

Der Kontextbereich wird auf die Anfangswerte zurückgesetzt, wenn ein `OPEN CONVERSATION`-Statement oder ein nicht-konversationelles `CALLNAT`-Statement ausgeführt wird.

Systemvariable *CONVID ändern

Die Systemvariable `*CONVID` (Format I4) wird durch das `OPEN CONVERSATION`-Statement gesetzt und kann durch das Anwendungsprogramm geändert werden.

Eine Änderung von `*CONVID` ist nur notwendig, wenn Sie mehrere Konversationen parallel verwenden.

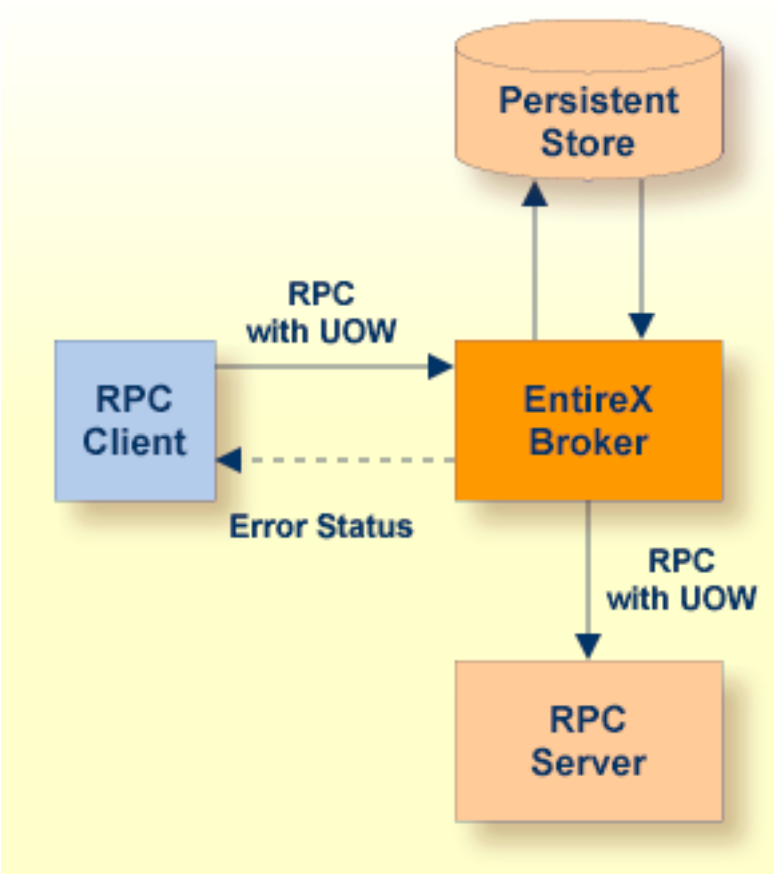
10 Reliable RPC

Allgemeine Informationen zu Reliable RPC

In der Architektur moderner E-Business-Anwendungen (SOA) gewinnen lose gekoppelte Systeme immer mehr an Bedeutung. Reliable Messaging (Verlässliche Nachrichtenübermittlung) ist eine wichtige Technologie bei dieser Art von Systemen.

Reliable RPC ist die Natural RPC-Implementierung eines Reliable Messaging-Systems. Sie kombiniert die Natural RPC-Technologie mit der Persistenz, die mittels Units of Work (UOWs) implementiert wird, die vom EntireX Broker angeboten werden. Reliable RPC zeichnet sich durch folgende Merkmale aus:

- Der Natural RPC-Client führt ein `CALLNAT`-Statement aus, ohne auf eine Antwort des Servers zu warten (die RPC-Nachricht wird im asynchronen Modus gesendet).
- Ein RPC-Server muss zum Zeitpunkt der Ausführung des `CALLNAT` nicht aktiv sein.
- Die Reliable RPC-Nachricht wird im Persistent Store des EntireX Brokers gespeichert, bis ein RPC Server verfügbar ist.
- Der Natural RPC Server führt den Reliable RPC aus, indem er das angeforderte Subprogramm aufruft, sendet aber keine Antwort an den RPC-Client.
- Ein Natural RPC-Client kann den Status der gesendeten Reliable RPC-Nachrichten abfragen.
- Ein Natural RPC-Client kann eine Reliable RPC-Nachricht an einen EntireX RPC-Server senden.
- Ein Natural RPC Server kann eine Reliable RPC-Nachricht von einem EntireX RPC-Client empfangen.



Reliable RPC auf der Natural RPC-Client-Seite

Der Natural RPC-Client für einen Reliable RPC wird auf die gleiche Weise konfiguriert wie bei einem normalen Natural RPC. Dieselbe Natural RPC-Client-Sitzung kann Standard-RPC-Anforderungen (Requests) und zuverlässige RPC-Nachrichten senden.

Damit ein Natural RPC-Client Reliable RPC verwenden kann, muss der Natural RPC-Client die Anwendungsprogrammierschnittstelle `USR2071N` für eine explizite EntireX Broker-Anmeldung verwenden. Dies bedeutet, dass der `RPC/NTRPC`-Schlüsselwort-Subparameter `ACIVERS` auf `2` oder höher gesetzt sein muss.

Reliable RPC wird verwendet, um Nachrichten an einen persistenten EntireX Broker-Dienst zu senden. Die Nachrichten werden durch den Parameterdatenbereich (PDA) des Aufrufers beschrieben und dürfen nur Ausgabeparameter enthalten. Ein Parameter wird auf eine der folgenden Arten als „Ausgabe“ definiert:

- Wenn ein Natural-**Interface-Objekt** verwendet wird:

- Setzen Sie im Feld Attr auf dem Bildschirm **Interface Object Generation** der SYSRPC Utility das Attribut des Parameters auf O (Output/Ausgabe).

  Anmerkung: Wenn Ihre Parameterdefinitionen keine Gruppenstrukturen enthalten, müssen Sie vor der Generierung des Interface-Objekts das Attribut COMPAT auf IDL setzen; siehe *Interface-Objekte generieren - Allgemeine Aspekte* in der SYSRPC Utility-Dokumentation.

- Wenn kein Natural-Interface-Objekt verwendet wird:
  - Verwenden Sie den Session-Parameter AD=O in dem CALLNAT-Statement.

    Anmerkung: Wenn Sie einen EntireX RPC-Server aufrufen wollen und die entsprechende IDL-Datei Gruppenstrukturen enthält, müssen Sie ein Natural Interface-Objekt verwenden, und die Parameterdefinition für das Natural Interface-Objekt muss der Gruppenstruktur der IDL-Datei entsprechen.

- Wenn Sie ein Natural-Interface-Objekt aus einer IDL-Datei generieren, wird das Attribut des Parameters aus der IDL-Datei übernommen. In diesem Fall darf die IDL-Datei nur Inbound-Parameter (aus Sicht des Servers) enthalten.

Reliable RPC wird zur Laufzeit aktiviert. Der Client muss einen von zwei verschiedenen Modi einstellen, bevor er eine Reliable RPC-Anforderung ausgibt:

- AUTO_COMMIT
- CLIENT_COMMIT

Während AUTO_COMMIT jede Nachricht nach dem Senden implizit festschreibt, kann eine Reihe von RPC-Nachrichten, die in einer Arbeitseinheit (Unit of Work, UOW) gesendet werden, im Modus CLIENT_COMMIT explizit festgeschrieben (Commit) oder zurückgenommen (Rollback) werden.

Dazu bietet Natural folgende Anwendungsprogrammierschnittstellen (APIs):

- Mit der API USR6304N wird der Modus für Reliable RPC eingestellt.
- Mit der API USR6305N kann eine Arbeitseinheit, die mit CLIENT_COMMIT erstellt wurde, bestätigt (Commit) oder zurückgesetzt (Rollback) werden.

Um USR6304N zu verwenden:

1 Kopieren Sie das Subprogramm USR6304N aus der Library SYSEXT in die Library SYSTEM oder in die Steplib Library oder in eine beliebige Anwendung in der Client-Umgebung.

2 Geben Sie mit dem Statement DEFINE DATA die folgenden Parameter an:

Parameter	I/O	Format	Beschreibung	
P-FUNC	I	A01	Funktionscode. Mögliche Werte:	
			P (Put)	Legt den Modus für Reliable RPC fest. Der Modus gilt für alle nachfolgenden Aufrufe.
			G (Get)	Ruft den zuvor festgelegten Modus ab.
P-MODE	I/O	N01	Modus für Reliable RPC:	
			0	Kein Reliable RPC (Standard-RPC-Ausführung).
			1	Reliable RPC AUTO_COMMIT
			2	Reliable RPC CLIENT_COMMIT
P-RC	O	N04	Rückgabecode	
P-MESSAGE	O	A80	Nachrichtentext	

3 Geben Sie im aufrufenden Programm auf der Client-Seite das folgende Statement an:

```
CALLNAT 'USR6304N' P-FUNC P-MODE P-RC P-MESSAGE
```

Anmerkung: Der Modus CLIENT_COMMIT kann nicht geändert werden, wenn Reliable RPC-Nachrichten zwar gesendet, aber noch nicht einem Commit oder Roll Back unterzogen wurden.

Um of USR6305N zu verwenden:

1 Kopieren Sie das Subprogramm USR6305N aus der Library SYSEXT in die Library SYSTEM oder in die Steplib Library oder in eine beliebige Anwendung in der Client-Umgebung.

2 Geben Sie mit dem Statement DEFINE DATA die folgenden Parameter an:

Parameter	I/O	Format	Beschreibung	
P-FUNC	I	A08	Funktionscode. Mögliche Werte:	
			COMMIT	Commit der gesendeten zuverlässigen RPC-Nachrichten. Die Nachrichten sind nun für einen RPC-Server verfügbar.
			ROLLBACK	Die bereits gesendeten Reliable RPC Nachrichten werden verworfen.
P-RC	O	N04	Rückgabecode	
P-MESSAGE	O	A80	Nachrichtentext	

3 Geben Sie im aufrufenden Programm auf der Client-Seite folgendes Statement an:

```
CALLNAT 'USR6305N' P-FUNC P-MODE P-RC P-MESSAGE
```

Reliable RPC auf der Natural RPC Server-Seite

Der Natural RPC Server für Reliable RPC wird auf dieselbe Weise konfiguriert wie für normales Natural RPC. Dieselbe Natural RPC Server-Sitzung kann Standard-RPC-Anforderungen (Requests) und Reliable RPC-Nachrichten verarbeiten.

Um die Verarbeitung von Reliable RPC-Nachrichten zu ermöglichen, muss der `RPC/NTRPC`-Schlüsselwort-Subparameter `ACIVERS` auf 2 oder höher gesetzt werden.

Status von Reliable RPC-Nachrichten anzeigen

Um den Status gesendeter Reliable RPC-Nachrichten anzuzeigen, steht in Natural die Anwendungsprogrammierschnittstelle `USR6306N` zur Verfügung:

- Mit `USR6306N` können Sie den Status aller Reliable RPC-Nachrichten abrufen, die Sie zuvor unter Ihrer Benutzerkennung gesendet haben.
- `USR6306N` muss nicht unbedingt innerhalb der Natural-Sitzung aufgerufen werden, in der die RPC-Nachrichten versendet wurden.
- Wenn `USR6306N` in einer anderen Natural-Sitzung verwendet wird, muss zunächst die Anwendungsprogrammierschnittstelle `USR2071N` verwendet werden, um sich mit derselben Benutzerkennung am EntireX Broker anzumelden, mit der die Reliable RPC-Nachrichten versendet wurden.

Die Reliable RPC-Nachrichten werden durch EntireX Broker-Arbeitseinheiten (Units of Work, UOW) implementiert. Die Informationen über Reliable RPC-Nachrichten sind daher Informationen über UOWs.

Um `USR6306N` zu verwenden:

1. Kopieren Sie das Subprogramm `USR6306N` aus der Library `SYSEXT` in die Library `SYSTEM` oder in die Steplib Library oder in eine beliebige Anwendung in der Client-Umgebung.
2. Geben Sie mit dem `DEFINE DATA`-Statement die folgenden Parameter an:

Parameter	I/O	Format	Beschreibung
P-UOW-ID-IN	I	A16	ID für die UOW, die abgerufen werden soll. Mögliche Werte: Kennung (UOWID) einer Arbeitseinheit (UOW). LAST: Die UOW für die letzte Reliable RPC-Nachricht in der aktuellen Natural-Sitzung. ALL oder leer: Alle UOWs für den angemeldeten EntireX Broker-Benutzer
P-USER-ID	O	A32	Benutzerkennung des Benutzers, der die UOWs erstellt hat.
P-BROKER-ID	O	A32	Broker-ID des EntireX Brokers, der die UOWs hostet.
P-UOW-COUNT	O	I4	Anzahl der UOWs im P-UOW-INFO-Array.
P-UOW-INFO (/1:*)			X-Array mit Informationen zu jeder UOW.
P-UOW-ID	O	A32	Kennung der UOW.
P-UOW-STATUS	O	A10	Status des UOW gemäß EntireX Broker. Der Status ist abhängig vom Stand der Verarbeitung und wird vom EntireX Broker zugewiesen.
P-USERR-STATUS	O	A32	Benutzerinformationen über die UOW. Dies ist typischerweise eine Fehlerinformation, die vom RPC-Server gesetzt wurde.
P-CREATE-TIME	O	A32	Erstellungszeitpunkt der UOW laut EntireX Broker.
P-RC	O	N04	Rückgabecode
P-MESSAGE	O	A80	Nachrichtentext

3 Geben Sie im aufrufenden Programm auf der Client-Seite das folgende Statement an:

```
CALLNAT 'USR6306N' P-UOW-ID-IN P-USER-ID P-BROKER-ID P-UOW-COUNT P-OUW-INFO(*) ↩
P-RC P-MESSAGE
```

11 Verwendung eines Remote Directory Servers - RDS

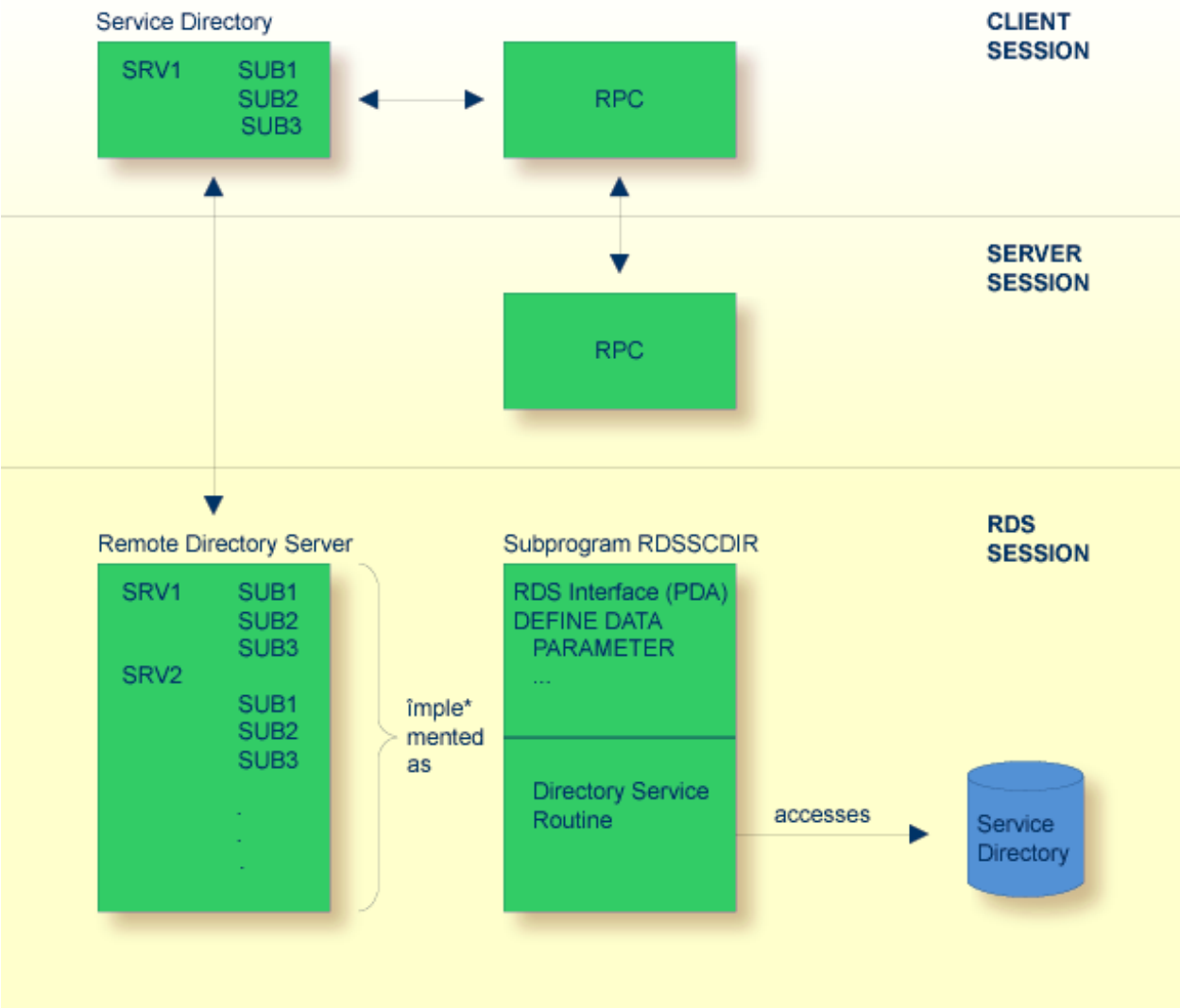
Funktionsprinzip eines RDS

Sie haben zwei Möglichkeiten, ein Diensteverzeichnis (Service Directory) zu verwenden:

1. **Verwendung eines Service Directory in einem Natural-Subprogramm**
   Normalerweise verwendet der Natural RPC zum Auffinden eines Dienstes ein Service Directory in einem Natural-Subprogramm. Dieses Verzeichnis ist eine initialisierte LDA-Datenstruktur im Programm `NATCLTGS`, die in der `SYSRPC` Utility mit der Funktion **Service Directory Maintenance** generiert wird und jeder RPC-Client-Anwendung zur Verfügung stehen muss.
2. **Verwendung eines Remote Directory**

   Sie können ein Remote Directory verwenden, um einen Dienst zu finden. Ein Remote Directory Server (RDS) ermöglicht es Ihnen, Verzeichnisdefinitionen an einer Stelle zu definieren, so dass die Dienste des RDS von allen Clients in Ihrer Umgebung verwendet werden können.

In diesem Abschnitt wird beschrieben, wie Sie einen Remote Directory Server verwenden, um einen Dienst zu finden.



Der Remote Directory Server wird in Form eines Natural-Subprogramms implementiert.

Ein Beispiel für ein solches Subprogramm ist in der Library `SYSRPC` enthalten. Es heißt `RDSSCDIR` und liest die erforderlichen Verzeichnisinformationen aus einer Arbeitsdatei. Die Schnittstelle dieses Subprogramms ist dokumentiert, so dass Sie Ihren eigenen Remote-Verzeichnisdienst entwickeln können. Weitere Informationen finden Sie im Abschnitt *Erstellen einer RDS-Schnittstelle*.

Die RDS-Schnittstelle ist der Natural-Parameterdatenbereich (PDA) des Natural-Subprogramms und die Verzeichnisdienstroutine ist der Codeteil des Natural-Subprogramms. Wenn ein Remote-`CALLNAT` nicht im lokalen Service Directory des Clients gefunden wird, kontaktiert die RPC-Laufzeit den Remote Directory Server, indem sie einen internen Remote-`CALLNAT` ausführt.

Ein interner Verzeichnisspeicher (Directory Cache) minimiert den Zugriff auf das Remote Directory. Die Cache-Informationen werden durch eine Verfallszeit gesteuert, die vom Remote Directory Server festgelegt wird.

Verwendung eines Remote Directory Servers

Um einen Remote Directory Server zu verwenden:

1 **Directory-Datei erstellen**

Erstellen Sie eine Directory-Datei für den Remote Directory Service mit Hilfe der **Service Directory Maintenance**-Funktion der SYSRPC Utility. Das Subprogramm RDSSCDIR wird in der Library SYSRPC bereitgestellt und liest die Verzeichnisinformationen aus einer Natural-Arbeitsdatei (Fixed-Block, Satzlänge 80 Byte). Dies ist die Arbeitsdatei CMWKF01, die dem entsprechenden Dataset in der Server Startup JCL zugewiesen ist.

2 **Remote Directory Server starten**

Starten Sie den Remote Directory Server und fahren Sie mit den folgenden Schritten fort.

3 **Remote Directory Server definieren**

Sie haben zwei Möglichkeiten:

- Geben Sie den Remote Directory Server im Schlüsselwort-Subparameter RDS des Profilparameters RPC oder des Parameter-Makros NTRPC an.
- Oder Sie verwenden die Maintenance-Funktion der Utility SYSRPC, die um Remote Directory Server zu definieren (siehe *Service Directory Maintenance* in der SYSRPC Utility-Dokumentation). Die Definition von Remote Directory Servern wird aus Kompatibilitätsgründen weiterhin unterstützt. Sie sollten jedoch Ihren RDS im Schlüsselwort-Subparameter RDS des Profilparameters RPC oder des Parameter-Makros NTRPC definieren. Zu diesem Zweck werden Einträge bereitgestellt, die es ermöglichen, den Standort des Directory Servers zu definieren. Auf diese Weise können Sie bestehende lokale Directory-Informationen um eine oder mehrere Remote Directory Server-Definitionen erweitern.

Nachfolgend finden Sie ein Beispiel für die Definition eines Remote Directory Servers im Service Directory NATCLTGS.

Service Directory					
	NODE	**SERVER**	**LIBRARY**	**PROGRAM**	**LOGON**
1	NODE1				
2		SERVER1			
3			SYSTEM		
4				TESTS1	
5				TESTS2	
6	RDSNODE				

Service Directory					
	NODE	SERVER	LIBRARY	PROGRAM	LOGON
7		DIRSRV1			
8			#ACI		
9				RDSSCDIR	

In diesem Beispiel wird lokal ein Server namens `SERVER1` definiert. Dieser Server kann die Dienste `TESTS1` und `TESTS2` ausführen.

Zusätzlich gibt es Definitionen für den Remote Directory Server `DIRSRV1`. Ein Remote Directory Server wird in der Library-Definition durch ein vorangestelltes Rautenzeichen (#) gekennzeichnet.

Die Definitionen von `NODE` und `SERVER` werden wie in Natural RPC üblich verwendet. Die Library-Definition definiert das Transportprotokoll (ACI), das für die Verbindung mit dem RDS verwendet werden muss.

Der Eintrag `PROGRAM` schließlich enthält den Namen des Remote-Subprogramms, das den Remote Directory Service repräsentiert (in diesem Fall verweist er auf das Beispiel-Subprogramm `RDSSCDIR`).

Erstellen einer RDS-Schnittstelle

Die RDS-Schnittstelle ist der Parameterdatenbereich (PDA) eines Natural-Subprogramms.

Um Ihre eigene RDS-Schnittstelle zu erstellen, können Sie den unten abgebildeten Parameterdatenbereich verwenden.

```
DEFINE DATA PARAMETER
  1 P_UDID(B8)                      /* OUT
  1 P_UDID_EXPIRATION(I4)           /* OUT
  1 P_CURSOR(I4)                    /* INOUT
  1 P_ENTRIES(I4)                   /* IN
  1 P_REQUEST(A16/1:250)            /* IN
  1 P_EXTENT (A16/1:250)            /* OUT
  1 P_RESULT(A32)                   /* OUT
  1 REDEFINE P_RESULT
    2 SRV_NODE(A8)
    2 SRV_NODE_EXT(A8)
    2 SRV_NAME(A8)
    2 SRV_NAME_EXT(A8)
END-DEFINE
```

Eine Erläuterung der Parameter finden Sie in der nachstehenden Tabelle.

Parameter	Format/Läange	Erläuterung
P_UDID	B8	Eindeutige Verzeichniskennung, sollte nach Änderung der Verzeichnisinformationen erhöht werden. Der Client speichert diese Kennung in seinem Cache. Erhöht sich die Binärzahl von einer Client-Anfrage zur nächsten, führt dies dazu, dass der Client seine lokalen Cache-Informationen löscht, da sie nicht mehr mit den Remote-Directory-Informationen übereinstimmen.
P_UDID_EXPIRATION	I4	Definiert die Verfallszeit in Sekunden, d. h. die Anzahl der Sekunden, in denen der Client seine lokalen Cache-Informationen verwenden kann, ohne eine Verbindung zum RDS herzustellen, um die UDID-Einstellung zu validieren. Damit können Sie eine Zeitspanne festlegen, nach der Sie sicher sein können, dass Ihre Verzeichnisänderungen für alle Clients aktiv sind. Wenn Sie diese Zeit auf einen unnötig niedrigen Wert setzen, können Sie erheblichen Netzwerkverkehr zum RDS verursachen.
P_CURSOR	I4	Der Remote Procedure Call hat die Möglichkeit, nach einem alternativen Server zu suchen, wenn keine Verbindung zum vorherigen Server hergestellt werden kann (siehe Schlüsselwort-Subparameter TRYALT des Profilparameters RPC oder des Parameter-Makros NTRPC. Dieser Parameter enthält den Wert Null für eine Suche von Beginn an und kann vom RDS geändert werden, um sich den Speicherort des Datensatzes zu merken und die Suche fortzusetzen. Der Wert wird vom Client nicht ausgewertet, er wird nur aus dem Cache eingefügt, um die Suche fortzusetzen.
P_ENTRIES	I4	Dieser Parameter enthält die Anzahl der Service-Definitionen in P_REQUEST.
P_REQUEST	A16/1:250	Eine Liste mit Services, für die eine Serveradresse abgefragt werden kann. Ein Eintrag ist wie folgt aufgebaut: Programmname (A8) Library-Name (A8)
P_EXTENT	A16/1:250	Reserviert für künftige Verwendung.
SRV_NODE	A8	Enthält den Serverknoten.
SRV_NODE_EXT	A8	Enthält die Erweiterung des Serverknotens.
SRV_NAME	A8	Enthält den Servernamen.
SRV_NAME_EXT	A8	Enthält die Erweiterung des Servernamens.

Erstellen einer Remote-Directory-Service-Routine

Die Remote Directory Service Routine ist der Codebereich eines Natural-Subprogramms (die Standardversion dieses Codebereichs ist das Subprogramm `RDSSCDIR` in der Library `SYSRPC`).

Um Ihre eigene RDS-Routine zu erstellen:

- Ändern Sie den unten dokumentierten Pseudocode.

```
Set UDID and UDID_EXPIRATION values
IF P_ENTRIES = 0
    ESCAPE ROUTINE
IF P_CURSOR != 0
   position to next server entry after P_CURSOR
   Scan for server which may execute P_REQUEST(*)
IF found
   SRV_NODE        = found node name
   SRV_NODE_EXT    = node extension
   SRV_NAME        = found server name
   SRV_NAME_EXT    = server extension
   P_CURSOR        = position of found server
ELSE
   P_CURSOR = 0
```

Remote Directory Service-Programm RDSSCDIR

Dieses Programm befindet sich in der Library `SYSRPC`. Es liest die Directory-Informationen aus einer Arbeitsdatei (Fixed-Block, Satzlänge 80 Byte).

Ihr Programm könnte die Directory-Informationen auch von anderer Stelle lesen (z.B. aus einer Datenbank). Bei der ausgelieferten Version von `RDSSCDIR` ist dies die Arbeitsdatei `CMWKF01`, die dem entsprechenden Dataset in der Server Startup JCL zugeordnet ist.

Struktur der Directory-Arbeitsdatei

```
* comment
UDID definition
UDID_EXPIRATION definition
node definition
...
node definition
```

UDID-Definition

```
(UDID)
   binary number            ↩
```

UDID_EXPIRATION-Definition

```
(UDID_EXPIRATION)
   number of seconds        ↩
```

Knoten-Definition

```
(NODE)
   namevalue     (logon-option)
   server definition
   ...
   server definition
```

Server-Definition

```
(SERVER)
   namevalue     (logon-option)
   library definition
   ...
   library definition
```

Library-Definition

```
(LIBRARY)
   namevalue
   program definition
   ...
   program definition
```

Programm-Definition

```
(PROGRAM)
   namevalue
   ...
   namevalue
```

Namevalue

```
Max. 8 characters in uppercase
```

Die *logon-option* nach *namevalue* sowie die folgenden Definitionszeilen sind optional. Die möglichen Werte für *logon-option* finden Sie unter *Service Directory Maintenance* in der *SYSRPC Utility*-Dokumentation.

Aus der Arbeitsdatei gelesenes Beispiel-Verzeichnis:

```
(UDID)
ACB8AAB4777CA000
  (UDID_EXPIRATION)
  3600
  * this is a comment
  (NODE)
  NODE1
         (SERVER)
         SERVER1
             (LIBRARY)
             SYSTEM
                  (PROGRAM)
                  TESTS1
                  TESTS2
                  TESTS3
         (SERVER)
         SERVER2   (logon-option)
               (LIBRARY)
               SYSTEM
                    (PROGRAM)
                    TESTS4
  (NODE)
  NODE2   (logon-option)
        (SERVER)
        SERVER1
              (LIBRARY)
              SYSTEM
                   (PROGRAM)
                   TESTS1
                   TESTS2
                   TESTS3
                   TESTS4    ↩
```

Im obigen Beispiel enthält das Verzeichnis:

- Zwei Server SERVER1 und SERVER2, die auf dem Knoten NODE1 laufen.

  Der Server SERVER1 darf die Programme TESTS1, TESTS2 und TESTS3 in der Library SYSTEM ausführen.

Der Server SERVER2 darf das Programm TESTS4 in der Library SYSTEM ausführen.

- Ein Server SERVER1 auf dem Knoten NODE2, der die Programme TESTS1 - TESTS4 in der Library SYSTEM ausführen darf.

Die Einrückung der Zeilen im obigen Beispiel ist nicht erforderlich. Alle Zeilen können an beliebiger Stelle (eins) beginnen. Sie können diese Datei manuell ändern oder sie mit der SYSRPC Utility-Funktion **Service Directory Maintenance** generieren.

12 Verwendung von Security

Natural RPC mit Natural Security verwenden

Der Natural RPC unterstützt auch Natural Security in Client/Server-Umgebungen, wobei Security auf einer (oder beiden) Seiten aktiv sein kann.

Allgemeine Informationen finden Sie in der *Natural Security*-Dokumentation.

Informationen darüber, wie Sie die Verwendung von Natural RPC-Aufrufen (Remote Procedure) in einer Client/Server-Umgebung kontrollieren können, finden Sie unter *Protecting Natural RPC Servers and Services* in der *Natural Security*-Dokumentation.

Client-Seite

Der Client muss die Anmeldedaten zusammen mit der RPC-Anforderung senden. Die Anmeldedaten bestehen aus Benutzerkennung, Passwort und Library.

- Benutzerkennung und Passwort werden zur Authentifizierung des Clients auf der Natural RPC Server-Seite verwendet.
- Die Library wird verwendet, um eine durch Natural Security geschützte Anmeldung bei der angeforderten Library durchzuführen.

Die folgenden Ausführungen gelten nur für Natural RPC-Clients. Für EntireX RPC-Clients, die auf einen durch Natural Security geschützten Natural RPC Server zugreifen, lesen Sie bitte die Dokumentation des EntireX Developer's Kit.

Um Anmeldedaten an den Natural RPC Server zu senden, muss die Logon-Option verwendet werden. Siehe *Betrieb einer Natural RPC-Umgebung* , ***Logon-Option benutzen***. Die Teile der Logon-Daten werden wie folgt aufgebaut:

1. Die Benutzerkennung und das Passwort:

   Wenn der Client unter Natural Security läuft

   Die Benutzerkennung und das Passwort aus der Natural Security-Anmeldung auf dem Client werden verwendet und an den Natural RPC Server weitergegeben.

   Wenn Sie eine andere Benutzerkennung und/oder ein anderes Passwort für die Natural Security-Anmeldung auf der Server-Seite verwenden wollen, können Sie die Anwendungsprogrammierschnittstelle `USR1071N` verwenden (siehe unten).

   Anmerkung: Sie können die Verwendung von `USR1071N` im Teil Natural RPC Restriktionen der *Session Parameters*-Restriktionen des Natural Security Library-Profils untersagen.

Wenn der Client nicht unter Natural Security läuft

Um die Benutzerkennung und das Passwort festzulegen, die an den Natural RPC Server übergeben werden, muss der Client die Anwendungsprogrammierschnittstelle `USR1071N` (siehe unten) aufrufen, bevor die erste RPC-Anforderung gesendet wird.

2. Die Library:

   Standardmäßig wird der Name der Library verwendet, bei der der Client gerade angemeldet ist. Wenn Sie dem Natural RPC Server einen anderen Library-Namen übergeben möchten, können Sie die Anwendungsprogrammierschnittstelle `USR4008N` verwenden.

Wenn für den Natural RPC Server Impersonation ohne Passwortprüfung aktiviert ist (das Feld `Impersonation` im Abschnitt *Components of an RPC Server Profile* in der *Natural Security*-Dokumentation ist auf `A` gesetzt), kann der Client optional eine ETID an den Natural RPC Server übergeben. Diese ETID wird vom Natural RPC Server für den Zugriff auf Adabas seitens des Clients verwendet. Um eine ETID auf der Natural RPC-Client-Seite anzugeben, können Sie die Anwendungsprogrammierschnittstelle `USR4371N` verwenden.

USR1071N

Die Anwendungsprogrammierschnittstelle `USR1071N` ist in der Library `SYSEXT` enthalten. Sie wird verwendet, um die Benutzerkennung und das Passwort anzugeben, die an den Natural RPC Server übergeben werden.

Um `USR1071N` zu verwenden:

1 Kopieren Sie das Subprogramm `USR1071N` und das Programm `USR1071P` aus der Library `SYSEXT` in die Library `SYSTEM` in der Systemdatei `FNAT` in der Serverumgebung (siehe *Eine Natural API verwenden* in the *SYSEXT Utility*-Dokumentation).

2 Geben Sie mit einem `DEFINE DATA`-Statement die folgenden Parameter an:

Parameter	I/O	Format	Beschreibung	
`USERID`	I	A08	Zu verwendende Benutzerkennung.	
`PASSWORD`	I	A08	Passwort zur Validierung der Benutzerkennung. Dieses Passwort wird nicht auf der Client-Seite validiert.	
`MIXEDCASE`	I	A01	Option für gemischte Groß- und Kleinschreibung beim Passwort (optional).	
			`Y`	Passwort in gemischter Groß-/Kleinschreibung zulassen.
			`N`	Passwörter in Großbuchstaben umwandeln.

3 Geben Sie im aufrufenden Programm auf der Client-Seite das folgende Statement an:

```
FETCH RETURN 'USR1071P' USERID PASSWORD [MIXEDCASE]
```

Alternativ können Sie USR1071P auch über die Kommandozeile aufrufen und Benutzerkennung und Passwort in das angezeigte Fenster eingeben.

Eine ausführliche Beschreibung finden Sie im Textobjekt USR1071T in der System Library SYSEXT.

Anmerkung: Für den Aufruf von USR1071N stehen zwei Beispiele zur Verfügung: USR1071P, bei dem nur Benutzerkennung und Passwort übergeben werden, und USR1071X (erweiterte Version), die es dem Benutzer zusätzlich ermöglicht, verschiedene Daten zu setzen/abzurufen.

USR4371N

Die Anwendungsprogrammierschnittstelle USR4371N wird in der Library SYSEXT bereitgestellt. Sie dient zur Angabe der Benutzerkennung und der ETID, die an den Natural RPC Server übergeben werden.

Um USR4371N zu verwenden:

1 Kopieren Sie das Subprogramm USR4371N und das Programm USR4371P aus der Library SYSEXT in die Library SYSTEM in der Systemdatei FNAT in der Client-Umgebung (siehe *Eine Natural API verwenden* in der *SYSEXT Utility*-Dokumentation).

2 Geben Sie mit einem DEFINE DATA-Statement die folgenden Parameter an:

Parameter	I/O	Format	Beschreibung
USERID	I	A08	Zu verwendende Benutzerkennung.
ETID	I	A08	Zu verwendende ETID.

3 Geben Sie im aufrufenden Programm auf der Client-Seite das folgende Statement an:

```
FETCH RETURN 'USR4371P' USERID ETID
```

Alternativ können Sie USR4371P auch über die Kommandozeile aufrufen und im angezeigten Fenster die Benutzerkennung und ETID eingeben.

Eine ausführliche Beschreibung finden Sie in dem Textobjekt USR4371T in der System Library SYSEXT.

Server-Seite

Wenn Natural Security auf der Server-Seite installiert ist und `AUTO` *nicht* angegeben ist, ist eine Natural-Anmeldung mit Benutzerkennung und Passwort erforderlich. Es wird empfohlen, den Natural-Profilparameter `STACK` zu verwenden, um das Natural-Systemkommando `LOGON` zu übergeben.

Wenn `AUTO=ON` angegeben ist, wird der Inhalt der Systemvariablen `*INIT-USER` wie üblich für eine interne Anmeldung verwendet.

Um die Logon-Option zu erzwingen - d.h. wenn Sie wollen, dass ein Server nur Anforderungen von Clients akzeptiert, bei denen die Logon-Option gesetzt ist - setzen Sie den Schlüsselwort-Unterparameter `LOGONRQ` des Profilparameters `RPC` oder des Parameter-Makros `NTRPC` für den Server auf `ON`.

Wenn die Logon-Option nicht erzwungen wird, werden Client-Anforderungen ohne Logon-Daten akzeptiert und in der Server Library oder einer ihrer Steplibs ausgeführt. So können Sie sowohl öffentliche (Public) als auch gesicherte (Secured) Dienste anbieten.

Wenn der Client Anmeldedaten übermittelt, werden die Benutzerkennung und das Passwort des Clients anhand des entsprechenden Benutzer-Sicherheitsprofils auf dem Server überprüft, und die Anmeldung bei der angeforderten Library und die Ausführung des Subprogramms erfolgen gemäß den entsprechenden Definitionen der Natural Security Library und des Benutzerprofils auf dem Server.

Nach der Ausführung des Subprogramms wird die Library, die vor der `CALLNAT`-Anforderung verwendet wurde, auf dem Server wieder aktualisiert. Im Falle eines konversationellen RPC setzt die erste `CALLNAT`-Anforderung innerhalb der Konversation die Library-Kennung auf dem Server, und das Statement `CLOSE CONVERSATION` setzt die Library-Kennung auf dem Server auf diejenige zurück, die vor der Eröffnung der Konversation verwendet wurde.

Als Teil der *Natural RPC Restrictions* in den Library Profiles von Natural Security wird eine Server-Session-Option `Close all databases` angeboten. Sie bewirkt, dass alle Datenbanken, die von Remote-Subprogrammen in der Library geöffnet wurden, geschlossen werden, wenn eine Natural-Anmeldung/Abmeldung an/von den Librarys erfolgt. Dies bedeutet, dass jeder Client seine eigene Datenbank-Sitzung verwendet.

Wenn die Option `Close all databases` gesetzt ist, ist es auch möglich, eine Client-spezifische ETID für alle Adabas-Zugriffe zu verwenden, die vom Server für diesen Client ausgeführt werden. In diesem Fall sollten Sie den Natural RPC Server mit `ETID=OFF` starten und für jeden Client, der eine ETID benötigt, eine entsprechende ETID im Benutzerprofil definieren, z.B. durch Angabe der ETID `*USER`. Bitte beachten Sie, dass in diesem Fall zwei Clients mit demselben Namen nicht zwei gleichzeitige Anforderungen mit Adabas-Aufrufen stellen können.

Passwort ändern

Es ist möglich, das Natural Security-Passwort auf dem Natural RPC Server über ein Natural RPC Service Request zu ändern. Zu diesem Zweck wird in der Library SYSEXT die Anwendungsprogrammierschnittstelle USR2074N bereitgestellt.

Um USR2074N zu nutzen:

1 Kopieren Sie das Subprogramm USR2074N und optional das Programm USR2074P aus der Library SYSEXT in die Library SYSTEM oder in die Steplib Library oder in eine beliebige Anwendung in der Serverumgebung.

2 Geben Sie mit einem DEFINE DATA-Statement die folgenden Parameter an:

Parameter	I/O	Format	Beschreibung	
USERID	I	A08	Zu verwendende Benutzerkennung.	
PASSWORD	I	A08	Passwort zur Validierung der Benutzerkennung. Dieses Passwort wird nicht auf der Client-Seite validiert.	
NEWPASSWORD	I	A08	Neues Passwort für die Benutzerkennung. Dieses Passwort wird nicht auf der Client-Seite validiert.	
NODE-NAME	I	A192	Name des zu adressierenden Serverknotens.	
SERVER-NAME	I	A32	Name des zu adressierenden Servers.	
PROTOCOL	I	A1	Transportprotokoll zur Adressierung des Server-Knotens. Gültiger Wert:	
			B	EntireX Broker
RC	O	I2	Rückgabewert:	
			0	OK, der Parameter MESSAGE enthält eine Bestätigungsnachricht.
			1	Fehler vom RPC- oder Server-Knoten, der Parameter MESSAGE enthält die Fehlermeldung.
			2	Fehler von der Schnittstelle, der Parameter MESSAGE enthält die Fehlermeldung.
			3	Natural Security-Fehler, der Parameter MESSAGE# enthält die Natural Fehlernummer und der Parameter MESSAGE enthält den entsprechenden Meldungstext.
MESSAGE#	O	N4	Zurückgegebene Meldungsnummer.	
MESSAGE	O	A80	Zurückgegebener Nachrichtentext.	

3 Geben Sie im aufrufenden Programm auf der Client-Seite das folgende Statement an:

```
CALLNAT 'USR2074N' user-id password newpassword node-name server-name protocol
rc message# message
```

Alternativ können Sie auch das Programm USR2074P aus der Library SYSEXT verwenden:

Rufen Sie USR2074P über die Kommandozeile auf und geben Sie die erforderlichen Daten in das angezeigte Fenster ein. In diesem Fall werden alle Eingaben außer den Passwörtern in Großbuchstaben umgewandelt. Bei den Passwörtern haben Sie die Möglichkeit, sie in gemischter Groß- und Kleinschreibung einzugeben oder nicht.

Impersonation (z/OS Batch-Modus)

Zweck der Impersonation

Impersonation ist eine optionale Funktion auf der Natural RPC Server-Seite und ist nur verfügbar, wenn der Natural RPC Server unter Natural Security läuft. Die Impersonierungsfunktion wird durch die Sicherheitsprofile für Natural RPC Server gesteuert (*Security Profiles for Natural RPC Servers*). Siehe das Feld Impersonation, das unter der Überschrift *Components of an RPC Server Profile* im Abschnitt *Protecting Natural RPC Servers and Services* in der *Natural Security*-Dokumentation beschrieben ist.

Impersonation im z/OS-Batch-Modus erfordert die Verwendung des Natural RPC Server-Frontends unter z/OS und verwendet die von z/OS bereitgestellte SAF-Schnittstelle.

Wenn Impersonation für den Natural RPC Server aktiv ist, wird eine Anforderung des Clients, die die **Logon-Option** verwendet, aus Sicht des Betriebssystems unter der Benutzerkennung ausgeführt, die der Client in den LOGON-Daten übergibt (Natural RPC-Benutzerkennung genannt).

Impersonation setzt voraus, dass der Zugriff auf das Betriebssystem, auf dem ein Natural RPC Server läuft, durch ein SAF-konformes externes Security-System kontrolliert wird. Die Benutzerauthentifizierung (Überprüfung der Natural RPC-Benutzerkennung und des Passworts) wird von diesem externen Security-System durchgeführt. Nach erfolgreicher Authentifizierung wird die Identität des Benutzers für das Betriebssystem festgestellt (d.h. es wird ein ACEE erstellt und mit der TCB verknüpft, unter der die aktuelle Client-Anforderung ausgeführt wird). Alle nachfolgenden Berechtigungsprüfungen werden auf der Grundlage dieser Identität durchgeführt. Dies bedeutet, dass alle Zugriffe auf Ressourcen, die durch das SAF-konforme externe Security-System kontrolliert werden, für diese Identität autorisiert sind. Dies gilt insbesondere für den Zugriff auf Arbeitsdateien und auf Datenbanken.

Impersonation bewirkt nicht, dass Natural Security ausgeschaltet wird. Nach erfolgreicher Authentifizierung der Identität des Benutzers durch das externe Security-System erfolgt eine Natural Security-Anmeldung mit denselben LOGON-Daten, aber ohne Passwortprüfung.

Wie Sie einen Natural RPC Server mit Impersonation starten, erfahren Sie unter *Starten eines Natural RPC Servers über das RPC-Server-Frontend* in *Starten eines Natural RPC Servers*.

Anmerkung: Ohne Impersonation wird eine Client-Anforderung, die die Logon-Option verwendet, aus Sicht des Betriebssystems mit der Benutzerkennung ausgeführt, unter der der Natural RPC Server gestartet wurde.

Schritte zur Aktivierung der Impersonation (Server-Seite)

1. **RPC-Server-Frontend installieren**
   Gehen Sie vor wie in den entsprechenden Schritten der Installationsdokumentation von Natural for Mainframes beschrieben; siehe *Installing Natural on z/OS*.

   Wenn Sie die empfohlene APF-autorisierte `LINKLIST` Library verwenden, müssen Sie sicherstellen, dass das resultierende Lademodul nicht in der `STEPLIB`- oder `JOBLIB`-Verkettung vorhanden ist.
2. **Natural z/OS-Batch-Nukleus mit der DB2-Schnittstelle DSNRLI verlinken**
   Dieser Schritt gilt nur für Natural for DB2-Benutzer.
3. **Reentrant Adabas-Batch-Link-Routine ADALNKR anstelle von ADALNK verwenden**
   Siehe *Besondere Aspekte bei Großrechner-Natural RPC Servers mit Replikaten* in *Starten eines Natural RPC Servers*.
4. **EntireX Broker Stub BROKER anstelle von NATETB23 verwenden**

   Siehe *Zugang zum EntireX Broker Stub auf dem Großrechner bereitstellen* in *Einrichten einer Natural RPC-Umgebung*.
5. **Alle erforderlichen RPC-Server-spezifischen Natural-Profilparameter definieren**
   Siehe *RPC-Server-spezifischen Natural-Parameter setzen* in *Einrichten einer Natural RPC-Umgebung*.

   Die Parameter werden entweder im Natural-Parameter-Modul oder im `CMPRMIN`-Dataset definiert. Der Parameter `PARM=` der JCL `EXEC`-Anweisung wird nicht verwendet, um Natural-Profilparameter bereitzustellen.
6. **RPC-Server-Profil in Natural Security definieren**
   Definieren Sie in Natural Security (NSC) ein RPC-Server-Profil für den Servernamen, der vom RPC-Server verwendet wird (`SRVNAME`), und aktivieren Sie die Impersonation.

   Siehe *Security Profiles for Natural RPC Servers* in *Protecting Natural RPC Servers and Services* in der *Natural Security*-Dokumentation.
7. **SAF-Definitionen prüfen**
   (Dieser Schritt gilt nur für Natural for DB2-Benutzer.)

   Wenn die SAF-Ressourcenklasse `DSNR` aktiv ist, müssen Sie prüfen, ob Sie die folgenden SAF-Definitionen benötigen:

```
RDEFINE DSNR (subsys.RRSAF) OWNER(DB2owner)

PERMIT subsys.RRSAF CLASS(DSNR) ID(DB2group) ACCESS(READ)
```

dabei ist *subsys* Ihre DB2-Subsystem-ID.

Jeder Benutzer, der auf DB2 zugreifen möchte, muss Mitglied der Gruppe *DB2group* sein.

Weitere Informationen finden Sie in der entsprechenden DB2-Dokumentation von IBM.

8. **User Exit NATRPC02 anlegen**
(Dieser Schritt gilt nur für Natural for DB2-Benutzer.)

   Legen Sie den Natural RPC User Exit NATRPC02 mit einem Aufruf von NATPLAN an, um den erforderlichen DB2-Plan zu setzen.

   Stellen Sie sicher, dass Sie einen NATPLAN Ihrer aktuellen Natural for DB2-Version verwenden.

   Beispiel NATRPC02:

```
DEFINE DATA PARAMETER
  1 SUBPROGRAM (A8) BY VALUE END-DEFINE
FETCH RETURN 'NATPLAN' 'planname'
```

9. **Natural RPC Server-Frontend starten**
Starten Sie das Natural RPC Server-Frontend.

   Siehe ***Starten eines Natural RPC Servers über das RPC-Server-Frontend*** in ***Starten eines Natural RPC Servers***.

   Vergewissern Sie sich, dass Sie alle erforderlichen Load Libraries zu Ihrer Steplib-Verkettung hinzugefügt haben. Sie benötigen insbesondere die folgenden Libraries:

   - Natural Load Library
   - EntireX Load Library
   - Adabas Load Library (wenn Sie die Adabas-Linkroutine ADAUSER verwenden)
   - DB2 Load Library (wenn Sie auf DB2 zugreifen wollen)

   Impersonation ist erfolgreich aktiviert, wenn Sie die folgenden Meldungen sehen:

   - Im Job-Protokoll:

```
RPC0010 Authorized environment for impersonation established
```

   - In der RPC-Trace-Datei:

```
M *** Server is running under NSC with impersonation
```

Schritte zur Verwendung von Impersonation (Client-Seite)

Der Client muss die Anmeldedaten zusammen mit der RPC-Anforderung senden, wie es bereits bei einem durch Natural Security (NSC) geschützten Standard Natural RPC Server der Fall ist. Im Gegensatz zu einem Standard Natural RPC Server muss die Benutzerkennung auch eine gültige SAF-Benutzerkennung sein und das Passwort muss das entsprechende SAF-Passwort sein. Benutzerkennung und Passwort werden vom Natural RPC Server gegen das externe Security-System auf dem z/OS-System validiert, unter dem der Server ausgeführt wird. Nach erfolgreicher Authentifizierung der Client-Identität durch das externe Security-System wird die Benutzerkennung von NSC gemäß den definierten Regeln validiert. Das Passwort wird ignoriert. Es ist daher nicht erforderlich, das NSC-Passwort auf Ihr SAF-Passwort zu setzen.

Wenn das im Abschnitt *Components of an RPC Server Profile* in der Natural Security-Dokumentation beschriebene Feld `Impersonation` auf `A` gesetzt ist, wird kein Kennwort zur Authentifizierung des Clients gegenüber dem externen Security-System verwendet. Diese Einstellung kann sinnvoll sein, wenn der Client bereits durch den EntireX Broker authentifiziert wurde.

Abhängig von der Art des Clients werden die Anmeldedaten unterschiedlich gesetzt:

Natural-Clients

1. **Logon-Option aktivieren**
   Aktivieren Sie die Logon-Option in der Funktion **Service Directory Maintenance** oder im Schlüsselwort-Subparameter `DFS` des Profilparameters `RPC` oder des Parameter-Makros `NTRPC`.

   Alternativ können Sie `USR2007N` verwenden, um die Option zu aktivieren.

   Siehe ***Logon-Option benutzen*** in ***Betrieb einer Natural RPC-Umgebung*** .
2. **Benutzerkennung und Passwort festlegen**

   Legen Sie die SAF-Benutzerkennung und das SAF-Passwort mit der Anwendungsprogrammierschnittstelle `USR1071P` fest.

   Wenn Ihr Client unter Natural Security (NSC) läuft und die Benutzerkennung und das Passwort von NSC mit der SAF-Benutzerkennung und dem SAF-Passwort identisch sind, ist `USR1071P` nicht erforderlich.

EntireX RPC-Clients

1. Aktivieren Sie die Natural **Logon-Option** entsprechend Ihrer Anwendungsumgebung.
2. Setzen Sie die RPC-Benutzerkennung und das RPC-Passwort auf die SAF-Benutzerkennung und das SAF-Passwort entsprechend Ihrer Anwendungsumgebung.

Regeln für Impersonation

- Die Impersonation findet zu Beginn jedes nicht-konversationellen `CALLNAT` und zu Beginn jeder Konversation statt.
- Die Authentifizierung von Natural RPC-Benutzerkennung und -Passwort wird durch das externe Security-System durchgeführt. Das Passwort in der `FSEC`-Systemdatei wird nicht verwendet.
- Nach erfolgreicher Authentifizierung wird die Natural RPC-Benutzerkennung für das Betriebssystem eingerichtet (der Benutzer wird impersoniert).
- Nach erfolgreicher Impersonierung:
  1. Für die Natural RPC-Benutzerkennung wird eine Natural Security-Anmeldung ohne Passwortprüfung durchgeführt.
  2. Alle Arbeitsdateien mit einem `DDNAME`, der nicht mit `CM` beginnt, werden mit der Natural RPC-Benutzerkennung geöffnet.
  3. Alle Adabas-Datenbanken werden mit der Natural RPC-Benutzerkennung geöffnet (gilt nur bei externer Adabas Security).
  4. Wenn im NSC-Benutzerprofil eine `ETID` angegeben ist, wird diese `ETID` in der Adabas-Open-Anforderung verwendet.
  5. Die DB2-Verbindung wird mit der Natural RPC-Benutzerkennung geöffnet (gilt nur für Natural for DB2-Benutzer).
- Am Ende jedes nicht-konversationellen `CALLNAT` und am Ende jeder Konversation wird die Natural RPC-Benutzerkennung beim Betriebssystem abgemeldet.
- Nach der Abmeldung:
  1. Alle Arbeitsdateien mit einem `DDNAME`, der nicht mit `CM` beginnt, werden geschlossen.
  2. Alle Adabas-Datenbanken werden geschlossen.

Impersonation (CICS)

Die folgenden Themen werden behandelt:

Zweck der Impersonation

Impersonation ist eine optionale Funktion auf der Natural RPC Server-Seite und ist nur verfügbar, wenn der Natural RPC Server unter Natural Security läuft. Die Impersonierungsfunktion wird durch die Sicherheitsprofile für Natural RPC Server gesteuert. Siehe *Security Profiles for Natural RPC Servers*) und das Feld `Impersonation`, das unter der Überschrift *Components of an RPC Server Profile* im Abschnitt *Protecting Natural RPC Servers and Services* in der Natural Security-Dokumentation beschrieben ist.

Impersonation unter CICS erfordert die Verwendung des Natural RPC Server-Frontends unter CICS und verwendet die von CICS bereitgestellte Schnittstelle.

Wenn Impersonation für den Natural RPC Server aktiv ist, wird eine Client-Anforderung, die die **Logon-OptionLogon-Option** verwendet, aus Sicht von CICS unter der Benutzerkennung ausgeführt, die der Client in den LOGON-Daten übergibt (Natural RPC-Benutzerkennung genannt). Impersonation unter CICS verwendet die CICS-Option, um eine CICS-Task unter einer gegebenen Benutzerkennung zu starten.

Nach Eingang einer Client-Anforderung startet das Natural RPC Server-Frontend eine neue CICS-Task mit der Option `USERID()` des Kommandos `EXEC CICS START TRANSID()`, wobei `USERID` die Natural RPC-Benutzerkennung ist. Die Benutzerauthentifizierung (Überprüfung von Natural RPC-Benutzerkennung und -Passwort) wird von CICS durchgeführt, typischerweise unter Verwendung des zugrunde liegenden externen Security-Systems. Nach erfolgreicher Authentifizierung wird die Identität des Benutzers für die CICS-Task festgelegt. Alle nachfolgenden Berechtigungsprüfungen werden auf der Grundlage dieser Identität durchgeführt. Das bedeutet, dass alle Zugriffe auf Ressourcen, die von CICS kontrolliert werden, für diese Identität autorisiert sind. Dies gilt insbesondere für Zugriffe auf CICS-Ressourcen und auf Datenbanken. Standardmäßig läuft die neu eingerichtete CICS-Task mit der gleichen CICS-Transaktionskennung, mit der Sie das RPC-Server-Frontend gestartet haben. Mit dem User-Exit `RPCSFEX1` können Sie eine auftragsspezifische Transaktionskennung einstellen.

Impersonation bewirkt nicht, dass Natural Security ausgeschaltet wird. Nach erfolgreicher Authentifizierung der Identität des Benutzers durch CICS erfolgt eine Natural Security-Anmeldung mit denselben `LOGON`-Daten ohne Passwortprüfung.

Wie Sie einen Natural RPC Server mit Impersonation starten, erfahren Sie unter *Starten eines Natural RPC Servers über das RPC-Server-Frontend* in *Starten eines Natural RPC Servers*.

Anmerkung: Ohne Impersonation wird eine Client-Anforderung, die die Logon-Option verwendet, aus Sicht des Betriebssystems mit der Benutzerkennung ausgeführt, mit der der Natural RPC Server gestartet wurde.

Schritte zur Aktivierung der Impersonation (Server-Seite)

1. **RPC-Server-Frontend installieren**
   Gehen Sie vor wie in den entsprechenden Schritten der Installationsdokumentation von Natural for Mainframes beschrieben; siehe *Installing the Natural CICS Interface on z/OS*.
2. **Adabas-Link-Routine für Adabas External Security installieren**
   Weitere Informationen finden Sie in der entsprechenden Adabas-Dokumentation (gilt nur für Benutzer der externen Security von Adabas).
3. **EntireX Broker Stub** `CICSETB` **anstelle von** `NATETB23` **verwenden**
   Siehe *Zugang zum EntireX Broker Stub auf dem Großrechner bereitstellen* in *Einrichten einer Natural RPC-Umgebung*.
4. **Alle erforderlichen RPC-Server-spezifischen Natural-Profilparameter definieren**
   Siehe *RPC-Server-spezifischen Natural-Parameter setzen* in *Einrichten einer Natural RPC-Umgebung*.

   Die Parameter werden entweder im Natural-Parameter-Modul oder zusammen mit der Transaktions-ID definiert.
5. **RPC-Server-Profil in Natural Security definieren**
   Definieren Sie in Natural Security (NSC) ein RPC-Server-Profil für den Server-Namen, der vom RPC-Server verwendet wird (`SRVNAME`), und aktivieren Sie die Impersonation.

   Siehe *Security Profiles for Natural RPC Servers* in *Protecting Natural RPC Servers and Services* of the *Natural Security* documentation.
6. **Bei CICS-Startparameter** `XUSER=YES`
   Wenn der CICS-Startparameter `XUSER=YES` angegeben ist, müssen Sie für jeden Client-Benutzer Ersatzbenutzer (Surrogate) definieren:

   ```
   RDEFINE SURROGATE userid1.DFHSTART UACC(NONE) OWNER(userid1) PERMIT
   userid1.DFHSTART CLASS(SURROGATE) ID(userid2) ACCESS(READ)
   ```

   Dabei ist:

   userid1 die Benutzerkennung des Clients,

   userid2 die Benutzerkennung, unter der das Natural RPC Server-Frontend gestartet wird.

   Weitere Informationen finden Sie in der entsprechenden CICS-Dokumentation von IBM.

7. **Einen CICS PROGRAM-Eintrag für das RPC-Server-Frontend definieren**
   Siehe entsprechenden Schritt in *Installing the Natural CICS Interface on z/OS*.
8. Definieren Sie einen CICS TRANSACTION-Eintrag für die Transaktions-ID, die das RPC-Server-Frontend aufruft.

   Siehe entsprechenden Schritt in *Installing the Natural CICS Interface on z/OS*.
9. **Einen `DB2TRAN`- und `DB2ENTRY`-Eintrag definieren**
   (Dieser Schritt gilt nur für Natural for DB2-Benutzer.)

   Define a `DB2TRAN` and `DB2ENTRY` entry for the transaction ID that invokes the RPC server front-end.

   Definieren Sie einen `DB2TRAN`- und `DB2ENTRY`-Eintrag für die Transaktions-ID, die das RPC-Server-Frontend aufruft. Starten Sie den Roll-Server
10. **Den Roll Server starten**
    Starten Sie den *Roll Server* für das vom Natural RPC Server verwendete Subsystem.

    (Dieser Schritt gilt nur, wenn der `NCMDIR`-Makro-Parameter `ROLLSRV` auf `YES` gesetzt ist).
11. **Das Natural RPC Server-Frontend unter CICS starten**
    Siehe ***Starten eines Natural RPC Servers über das RPC-Server-Frontend (nur bei CICS)*** in ***Starten eines Natural RPC Servers***.

    Die Impersonation ist erfolgreich aktiviert, wenn Sie die folgende Meldung in der RPC-Trace-Datei sehen:

    ```
    M *** Server is running under NSC with impersonation
    ```

Schritte zur Verwendung von Impersonation (Client-Seite)

Der Client muss die Anmeldedaten zusammen mit der RPC-Anforderung senden, wie dies bereits bei einem Standard-Natural-RPC-Server mit NSC-Schutz der Fall ist. Im Gegensatz zu einem Standard-Natural-RPC-Server muss die Benutzerkennung auch eine gültige CICS-Benutzerkennung sein und das Passwort muss das entsprechende Passwort des externen Security-Systems sein. Benutzerkennung und Passwort werden von CICS gegen das externe Security-System auf dem z/OS-System, unter dem CICS ausgeführt wird, validiert. Nach erfolgreicher Authentifizierung der Client-Identität durch das externe Security-System wird die Benutzerkennung durch Natural Security gemäß den definierten Regeln validiert. Das Passwort wird ignoriert. Es ist daher nicht erforderlich, das NSC-Passwort auf Ihr SAF-Passwort zu setzen.

Wenn das Feld `Impersonation`, das im Abschnitt *Components of an RPC Server Profile* in der *Natural Security*-Dokumentation beschrieben ist, auf `A` gesetzt ist, wird kein Passwort verwendet, um den Client gegenüber dem externen Security-System zu validieren. Diese Einstellung kann sinnvoll sein, wenn der Client bereits durch den EntireX Broker authentifiziert worden ist.

Je nach Art des Clients werden die Anmeldedaten unterschiedlich eingestellt:

Natural-Clients

1. **Logon-Option aktivieren**
   Aktivieren Sie die Logon-Option in der Funktion **Service Directory Maintenance** oder im Schlüsselwort-Subparameter `DFS` des Profilparameters `RPC` oder des Parameter-Makros `NTRPC`.

   Alternativ können Sie die Option auch über die Anwendungsprogrammierschnittstelle `USR2007N` einschalten.

   Siehe *Logon-Option benutzen* in *Betrieb einer Natural RPC-Umgebung* .
2. **Benutzerkennung und Passwort festlegen**

   Legen Sie die Benutzerkennung und das Passwort über die Anwendungsprogrammierschnittstelle `USR1071P` fest.

   Wenn Ihr Client unter Natural Security (NSC) läuft und die Benutzerkennung und das Passwort von NSC mit der Benutzerkennung und dem Passwort auf der Server-Seite identisch sind, dann ist `USR1071P` nicht erforderlich.

EntireX RPC Clients

1. **Natural-Logon-Option aktivieren**
   Aktivieren Sie die Natural-Logon-Option entsprechend Ihrer Anwendungsumgebung.
2. **RPC-Benutzerkennung und -Passwort festlegen**
   Legen Sie die RPC-Benutzerkennung und das RPC-Passwort entsprechend Ihrer Anwendungsumgebung fest.

Regeln für Impersonation

- Die Impersonation erfolgt zu Beginn jedes nicht-konversationellen `CALLNAT` und zu Beginn jeder Konversation.
- Der optionale User-Exit `RPCSFEX1` **user exit** wird aufgerufen, um eine anforderungsspezifische CICS-Transaktionskennung zu setzen.
- Die Authentifizierung der Natural RPC-Benutzerkennung und des Passworts wird von CICS durchgeführt. Das in der Systemdatei `FSEC` gespeicherte Kennwort wird nicht verwendet.
- Nach erfolgreicher Authentifizierung wird die Natural RPC-Benutzerkennung für CICS eingerichtet (Benutzer wird impersoniert).
- Nach erfolgreicher Impersonierung:
  1. Für die Natural RPC-Benutzerkennung wird eine Natural Security-Anmeldung ohne Passwortprüfung durchgeführt.
  2. Der Zugriff auf alle CICS-Ressourcen erfolgt mit der Natural RPC-Benutzerkennung.
  3. Alle Adabas-Datenbanken werden mit der Natural RPC-Benutzerkennung geöffnet (gilt nur bei externer Adabas Security).

4. Wenn im NSC-Benutzerprofil eine ETID angegeben ist, wird diese ETID in der Open-Anforderung von Adabas verwendet.
5. Die DB2-Verbindung wird mit der Natural RPC-Benutzerkennung geöffnet (gilt nur für Natural for DB2-Benutzer).

- Am Ende jedes nicht-konversationellen CALLNAT und am Ende jeder Konversation wird die Natural RPC-Benutzerkennung vom CICS abgemeldet.
- Nach der Abmeldung:
  1. Alle CICS-Ressourcen werden geschlossen.
  2. Alle Adabas-Datenbanken werden geschlossen.
  3. Die Verbindung zu DB2 wird geschlossen (gilt nur für Natural for DB2-Benutzer).

RPCSFEX1 - User-Exit für Impersonation unter CICS

Standardmäßig ist die CICS-Transaktionskennung, mit der die impersonierte RPC-Anforderung ausgeführt wird, dieselbe wie die CICS-Transaktionskennung, die zum Starten des RPC-Server-Frontends verwendet wird. Mit dem User-Exit RPCSFEX1 können Sie eine anforderungsspezifische Transaktionskennung einstellen.

Der User-Exit RPCSFEX1 wird aufgerufen, wenn nach dem Empfang einer RPC-Anforderung die Anmeldedaten eines Clients ausgewertet werden. Die Anmeldedaten werden dann verwendet, um die CICS-Transaktionskennung zu setzen, unter der die impersonierte RPC-Anforderung ausgeführt wird.

You must link RPCSFEX1 to the RPC server front-end for CICS and call the user exit by using the following standard conventions:

Sie müssen RPCSFEX1 mit dem RPC-Server-Frontend für CICS verlinken und den User-Exit unter Beachtung der Standardkonventionen aufrufen.

RPCSFEX1 wird unter Beachtung folgender Standardkonventionen aufgerufen:

Register	Inhalt
15	Eintragsadresse von NATSFEX1.
14	Rücksprungadresse des Natural RPC Server-Frontend.
13	Adresse eines Speicherbereichs von 18 Wörtern.
1	Adresse einer Parameterliste.

Die Parameterliste enthält die folgenden Adressen:

Adresse	Parameter	I/O	Natural-Daten Format/Länge
1	CICS-Transaktionskennung	I/O	A04
2	EntireX-Benutzerkennung des Client	I	A32
3	RPC-Benutzerkennung des Client	I	A8
4	RPC-Passwort des Client	I	A8
5	Name der Natural Library, in der die RPC-Anforderung auf dem Natural RPC Server ausgeführt werden soll.	I	A8
6	Name des Subprogramms, das auf dem Natural RPC Server ausgeführt werden soll.	I	A8

Sie können die von CICS vergebene Transaktionskennung nur mit dem User-Exit ändern.

Natural RPC mit EntireX Security verwenden

Der Natural RPC unterstützt EntireX Security sowohl auf der Client- als auch auf der Server-Seite vollständig.

- EntireX Security auf der Client-Seite
- EntireX Security auf der Server-Seite
- EntireX Security auf der Server-Seite mit Trusted User IDs (nur z/OS)

EntireX Security auf der Client-Seite

Für die An- und Abmeldung beim EntireX Broker steht die Natural Anwendungsprogrammierschnittstelle `USR2071N` zur Verfügung.

Um sich beim EntireX Broker anzumelden:

- Verwenden Sie die Anmeldefunktion von `USR2071N` und übermitteln Sie Ihre Benutzerkennung und Ihr Passwort an den ausgewählten EntireX Broker.

  Nach einer erfolgreichen Anmeldung wird das zurückgegebene Security Token von Natural gespeichert und bei jedem weiteren Aufruf an den EntireX Broker weitergegeben. Die Logon-Option ist für die Natural-Anwendung völlig transparent.

Wenn EntireX Security installiert ist oder wenn `AUTOLOGON=NO` in der EntireX Broker-Attributdatei angegeben ist, müssen Sie `USR2071N` mit der Anmeldefunktion vor der allerersten Remote-`CALLNAT`-Ausführung aufrufen.

Es wird empfohlen, `USR2071N` mit der Abmeldefunktion aufzurufen, sobald Sie nicht mehr beabsichtigen, einen Remote-`CALLNAT` zu verwenden.

Um USR2071N zu verwenden:

1 Kopieren Sie das Subprogramm USR2071N aus der Library SYSEXT in die Library SYSTEM oder in die Steplib Library oder in eine beliebige Anwendung in der Serverumgebung.

2 Geben Sie mit einem DEFINE DATA-Statement die folgenden Parameter an:

Parameter	I/O	Format	Beschreibung	
function	I	A08	Funktionscode. Mögliche Werte:	
			LOGON	Anmeldung beim EntireX Broker
			LOGOFF	Abmeldung vom EntireX Broker
broker-id	I	A192	Broker-Kennung:. **Anmerkung:** Aus Kompatibilitätsgründen ist *broker-id* mit BY VALUE RESULT definiert, um existierende Aufrufer zu unterstützen, die ein A8- oder A32-Feld für die *broker-id* übergeben. Das in der Library SYSEXT enthaltene Beispiel USR2071P unterstützt bis zu 32 Zeichen.	
user-id	I	A32	Benutzerkennung.	
password	I	A32	Passwort der Benutzerkennung.	
newpassw	I	A32	Neues Passwort der Benutzerkennung.	
rc	O	N04	Rückgabewert:	
			0	OK
			1	Ungültiger Funktionscode.
			9999	EntireX Broker-Fehler (siehe Parameter *message*).
message	O	A80	Vom EntireX Broker zurückgegebener Nachrichtentext.	

3 Geben Sie im aufrufenden Programm auf der Client-Seite das folgende Statement an:

```
CALLNAT 'USR2071N' function broker-id user-id password newpassword rc message
```

Siehe auch *Syntax-Beschreibung* beim CALLNAT-Statement.

Alternativ können Sie USR2071P auch über die Kommandozeile aufrufen und Benutzerkennung und Passwort in das angezeigte Fenster eingeben. In diesem Fall werden alle Eingaben außer den Passwörtern in Großbuchstaben umgewandelt. Bei den Passwörtern haben Sie die Möglichkeit, diese in gemischter Schreibweise einzugeben oder nicht.

Funktionen:

LOGON	Eine EntireX Broker LOGON-Funktion wird für die genannte Broker-Kennung (*broker-id*) mit der übergebenen Benutzerkennung (*user-id*) und dem Passwort (*password*) ausgeführt. Nach erfolgreichem LOGON-Aufruf kann der Client wie gewohnt mit der EntireX Broker-Broker-Kennung (*broker-id*) kommunizieren. Mit *newpassw* kann der Client-Benutzer sein Passwort mittels der EntireX Security-Funktionalität ändern. **Anmerkungen:** ■ Wenn eine erfolgreiche Anmeldung durchgeführt wurde, wird die in diesem LOGON verwendete Benutzerkennung bei allen nachfolgenden Remote Procedure CALLNATs, die über diesen EntireX Broker geleitet werden, an den genannten EntireX Broker weitergegeben. Ohne ein explizites LOGON wird der aktuelle Inhalt der Systemvariablen *USER verwendet. Dasselbe gilt, wenn Sie ein LOGON an EntireX Broker 1 abgesetzt haben, Ihr Remote Procedure CALLNAT aber über EntireX Broker 2 geleitet wird. ■ Es ist möglich, sich gleichzeitig bei mehreren EntireX Brokern anzumelden. Für jedes LOGON kann eine andere Benutzerkennung verwendet werden. ■ Die für die Anmeldung beim EntireX Broker verwendete Benutzerkennung kann sich von der Natural-Benutzerkennung unterscheiden, unter der die Client-Anwendung läuft. ■ Eine interne Neuanmeldung erfolgt, nachdem eine EntireX Broker-Zeitüberschreitung (Timeout) auftrat, wenn die ursprüngliche Anmeldung ohne Passwort erfolgte (das bei der Anmeldung verwendete Passwort wird nicht gespeichert). Wenn nach einer Zeitüberschreitung keine interne Neuanmeldung möglich ist, muss der Client das LOGON explizit neu durchführen. ■ Am Ende der Natural-Sitzung wird ein implizites LOGOFF bei allen EntireX Brokern durchgeführt, bei denen eine Anmeldung erfolgt ist.
LOGOFF	Eine EntireX-Broker-LOGOFF-Funktion wird für die genannte Broker-Kennung (*broker-id*) ausgeführt.

Besondere Aspekte, wenn die Client-Anfrage auf der Server-Seite ausgeführt wird:

Wenn eine RPC-Client-Anfrage auf der Natural RPC Server-Seite ausgeführt wird, muss vor der Ausführung der RPC-Client-Anforderung auch eine Anmeldung beim EntireX Broker über die Anwendungsprogrammierschnittstelle USR2071N erfolgen. Die Anmeldedaten des Natural RPC Servers selbst werden nicht für RPC-Client-Anforderung verwendet.

Wenn die RPC-Client-Anforderung an denselben EntireX Broker gesendet wird, bei dem der Natural RPC Server registriert ist, muss die Benutzerkennung verschieden vom Wert des Schlüsselwort-Subparameters SRVUSER sein.

EntireX Security auf der Server-Seite

Wenn der Wert des Schlüsselwort-Subparameters ACIVERS des Profilparameters RPC oder des Parameter-Makros NTRPC 2 oder höher ist, meldet sich der Server beim Start der Sitzung mit der LOGON-Funktion beim EntireX Broker an. Die Benutzerkennung ist die gleiche wie die durch SRVUSER definierte Benutzerkennung.

Wenn EntireX Security installiert wurde und wenn die Funktion EntireX Trusted User ID nicht verfügbar ist, gibt es zwei alternative Möglichkeiten, das erforderliche Passwort anzugeben:

- Schlüsselwort-Subparameter SRVUSER='*NSC' setzen
- Anwendungsprogrammierschnittstelle USR2072N zur Angabe eines Passworts verwenden

Diese Alternativen werden im Folgenden beschrieben.

Schlüsselwort-Subparameter SRVUSER='*NSC' setzen

Wenn Natural Security auf dem Server installiert ist, können Sie den Schlüsselwort-Subparameter SRVUSER des Profilparameters RPC oder des Parameter-Makros NTRPC auf '*NSC' setzen, um festzulegen, dass die aktuelle Natural Security-Benutzerkennung, die beim Start des Servers verwendet wurde, für das LOGON in Verbindung mit dem zugehörigen Natural Security-Passwort verwendet wird. In diesem Fall muss der bei ACIVERS eingestellte Wert mindestens 4 betragen.

Anwendungsprogrammierschnittstelle USR2072N zur Angabe eines Passworts verwenden

Mit der Anwendungsprogrammierschnittstelle USR2072N können Sie ein Passwort angeben, das zur Anmeldung (LOGON) in Verbindung mit dem Schlüsselwort-Subparameter SRVUSER des Profilparameters RPC oder des Parameter-Makros NTRPC verwendet wird.

Um USR2072N zu verwenden:

1. Kopieren Sie das Subprogramm USR2072N und optional das Programm USR2072P aus der Library SYSEXT in die Library SYSTEM oder in die Steplib Library oder in eine beliebige Anwendung in der Serverumgebung.
2. Geben Sie mit einem DEFINE DATA-Statement den folgenden Parameter an:

Parameter	I/O	Format	Beschreibung
password	I	A32	Das Passwort der Benutzerkennung.

3. Geben Sie im aufrufenden Programm auf der Client-Seite das folgende Statement an:

```
CALLNAT 'USR2072N' password
```

See also the of the CALLNAT statement.

Siehe auch *Syntax-Beschreibung* des CALLNAT-Statement.

4 Das aufrufende Programm muss ausgeführt werden, bevor der Natural RPC Server seine Initialisierung gestartet hat. Dazu legen Sie den Namen des aufrufenden Programms beim Start des Servers auf den Natural-Stack. Zu diesem Zweck können Sie auch das Programm USR2072P aus der Library SYSEXT verwenden. In diesem Fall wird das Passwort standardmäßig in Großbuchstaben umgewandelt. Sie haben die Möglichkeit, das Passwort in gemischter Groß- und Kleinschreibung einzugeben, indem Sie als zweiten Parameter die Mixed Case Option mit Y übergeben.

```
STACK=(LOGON server-library;USR2072P password [Y])
```

EntireX Security auf der Server-Seite mit Trusted User IDs (nur z/OS)

Die Funktion EntireX Trusted User ID von EntireX Security ist nur für die Transportmethode NET verfügbar. Wenn Sie die Transportmethode TCP mit Trusted User IDs verwenden möchten, können Sie die RACF PassTicket-Funktionalität nutzen.

Bei Verwendung der RACF-PassTicket-Funktionalität generiert der Natural RPC Server ein PassTicket für seine eigene Benutzerkennung und übergibt diese Kennung und das generierte PassTicket zur Authentifizierung an den Secured EntireX Broker, anstatt eine Benutzerkennung und ein Passwort bei der Anmeldung beim Secured EntireX Broker anzugeben. Die eigene Benutzerkennung des Servers wird aus der ACEE der Natural RPC Server-Task übernommen. Das PassTicket wird für diese Benutzerkennung und den Standard-Batch-Anwendungsnamen MVS*smf-id* mit Hilfe der Service-Routine R_GenSec IRRSGS00 erzeugt (siehe auch die ***z/OS Security Server RACF***-Dokumentation von IBM).

Die folgenden Schritte sind erforderlich, um vertrauenswürdige Benutzerkennungen für die Transportmethode TCP zu verwenden:

1. Geben Sie SRVUSER='*TRUSTED' beim Starten des Natural RPC Servers an.
2. Definieren Sie Folgendes in RACF:

```
SETROPTS CLASSACT(PTKTDATA) RACLIST(PTKTDATA)
RDEF PTKTDATA MVSsmf-id                    OWNER(#SECADM)  UACC(NONE)
     SSIGNON(KEYENCRYPTED(XXXXXXXXXXXXXXXX))
RDEF PTKTDATA IRRPTAUTH.MVSsmf-id.*        OWNER(#SECADM)  UACC(NONE)
PERMIT  IRRPTAUTH.MVSsmf-id.*   CL(PTKTDATA) ID(user-id) ACCESS(UPDATE)
```

Dabei ist:

smf-id SMF-Systemkennung des MVS-Systems, auf dem die Anwendung läuft.

Die SMF-Kennung befindet sich im SMFPRM*xx*-Member von SYS1.PARMLIB und wird durch den SID-Wert angegeben.

user-id Die Benutzerkennung der RPC-Server-Task oder die Benutzerkennung der CICS-Task, wenn diese unter CICS läuft.

13 EntireX Broker-Unterstützung

Security

Sowohl der Natural RPC-Client als auch der Natural RPC Server unterstützen EntireX Security.

Wenn ein Natural RPC-Client oder ein Natural RPC Server mit `ACIVERS=8` gestartet wird, wird EntireX Security ohne Stub-Exits (nur Großrechner) und ohne `SECUEXIT` (alle Plattformen) unterstützt. In diesem Fall führt Natural vor jedem anderen Broker-Aufruf einen `KERNELVERS`-Aufruf durch, um die aktuelle Einstellung des ACI-Feldes `KERNELSECURITY` zu erhalten. Diese `KERNELSECURITY`-Einstellung wird bei allen nachfolgenden Broker-Aufrufen weitergegeben. Mit dieser Funktion kann ein Natural RPC-Client innerhalb derselben Natural-Sitzung auf einen gesicherten und einen ungesicherten EntireX-Broker zugreifen.

Protokollierung und Abrechnung

Natural RPC-Client und Natural RPC Server unterstützen die Protokollierung (Logging) und die Abrechnung (Accounting) des RPC-Programms und der RPC Library innerhalb des EntireX Brokers.

Der Natural RPC-Client liefert dem EntireX Broker den Namen des auszuführenden Subprogramms und den Namen der Library, aus der das Subprogramm zum EntireX Broker ausgeführt werden soll.

Der Natural RPC Server gibt den Namen des ausgeführten Subprogramms und den Namen der Library zurück, aus der das Subprogramm tatsächlich ausgeführt wurde.

14 API zur Bereitstellung eines RPC-Kontexts von der Natural RPC-Client-Seite

Dieses Kapitel stellt die API RPC-CNTX vor, die zur Bereitstellung eines RPC-Kontexts von der Natural RPC-Client-Seite aus verwendet werden kann.

RPC-CNTX

Die API RPC-CNTX kann verwendet werden, um einen Kontext für RPC-Aufrufe bereitzustellen, wie es in Testprogrammen geschieht, die mit dem webMethods EntireX Natural Wrapper generiert werden.

RPC-CNTX kombiniert die Funktionalität mehrerer APIs und ist in der Library SYSTEM verfügbar. Es sind keine zusätzlichen Vorbereitungen wie das Festlegen einer Steplib oder das Kopieren von APIs aus der Library SYSEXT in die User Libraries erforderlich.

Um RPC-CNTX zu nutzen:

1 Specify a DEFINE DATA statement

Geben Sie ein DEFINE DATA-Statement an, wobei die Ebene 01 durch 01 RPC-CNTX-AREA definiert ist, und geben Sie die folgenden Parameter auf Ebene 02 an:

Parameter	Format	I/O	Beschreibung
FUNCTION	A2	I	'SL' Setzt den RPC-Kontext in Abhängigkeit von den angegebenen Parametern und meldet sich beim EntireX Broker an. 'SC' Setzt den RPC-Kontext in Abhängigkeit von den angegebenen Parametern. 'GC' RPC-Kontext abfragen. Standardmäßig beziehen sich die erforderlichen Informationen auf den Standardserver. 'LO' Anmelden beim EntireX Broker. 'LF' Abmelden beim EntireX Broker.
BROKERID	A dynamic	I/O	Geben Sie den Broker-Namen an. Siehe den Schlüsselwort-Subparameter SRVNODE des Profilparameters RPC.
CLASS	A32	I	Geben Sie die Klasse des Servers an. Bei Natural RPC Servern ist dies RPC. Siehe den Schlüsselwort-Subparameter SRVNAME des Profilparameters RPC.
SERVER	A32	I/O	Geben Sie den Namen des Servers an. Siehe den Schlüsselwort-Subparameter SRVNAME des Profilparameters RPC.

Parameter	Format	I/O	Beschreibung
SERVICE	A32	I	Geben Sie den Typ des Servers an. Bei Natural RPC Servern ist dies `CALLNAT`.
COMPRESSLEVEL	A1	I/O	Komprimierungsstufe erfordert API-Version 7.
RESERVED	I1	I/O	Reserviert für künftige Verwendung.
RPC-LIBRARY	A8	I	
RPC-RELIABLE-STATE	N1	I/O	`0` Reliable RPC (Standard-RPC-Ausführung) `1` Reliable RPC (`AUTO_COMMIT`) `2` Reliable RPC (`Client_COMMIT`)
NAT-LOGON	A11	I/O	
EXX-USERID	A32	I	
EXX-PASSWORD	A32	I	
RPC-USERID	A32	I	
RPC-PASSWORD	A32	I	
ERR-CODE	I4	O	Der Ursprung eines Fehlercodes kann anhand der folgenden Identifikatoren ermittelt werden: `USR1071N`,(`USR4371N`) 1nnnn `USR6304N` 2nnnn `USR2007N` 3nnnn `USR4008N` 5nnnn `USR4009N` 8nnnn `USR2071N` 9nnnn
ERR-TEXT	A dynamisch	O	
RPC-ETID	A8	I	Impersonation ohne Passwortüberprüfung zulassen (optional).

Anmerkung: Die wichtigsten Funktionscodes sind `'SL'` und `'LF'`. Der Funktionscode `'SL'` ist eine Kombination aus dem Setzen des RPC-Kontextes (`'SC'`) und einer Anmeldung am EntireX Broker (`'LO'`).

2 Geben Sie im aufrufenden Programm auf der Client-Seite das folgende Statement an:

```
CALLNAT 'RPC-CNTX' RPC-CNTX-AREA
```

Weitere Informationen finden Sie unter Syntax-Beschreibung beim CALLNAT-Statement in der *Statements*-Dokumentation.

Die Einstellung des RPC-Kontextes wird wie folgt verarbeitet:

1. Legen Sie die Anmeldeinformationen für den RPC-Server durch Angabe von RPC-USERID und RPC-PASSWORD fest (API USR1071N).
2. Setzen Sie den Modus für Reliable RPC durch Angabe von RPC-RELIABLE-STATE (API USR6304N).
3. Setzen Sie die Daten für den RPC-Standard-Server einschließlich der **Logon-Option** durch Angabe von BROKERID, SERVER und NAT-LOGON (API USR2007N).
4. Ändern Sie den Library-Namen auf dem RPC-Server für die Anmeldung durch Angabe von RPC-LIBRARY (API USR4008N). Wenn der Name leer gelassen wird, wird die API nicht aufgerufen.
5. Setzen Sie die Parameter für EntireX COMPRESSLEVEL, indem Sie die API USR4009N verwenden. Wenn keine Werte angegeben werden, wird die API nicht aufgerufen.
6. Legen Sie optional eine ETID fest, indem Sie RPC-ETID (API USR4371N) angeben, wenn für den Natural RPC Server Impersonation ohne Passwortprüfung aktiv ist.
7. Für die Anmeldung am EntireX Broker geben Sie BROKERID, EXX-USERID und EXX-PASSWORD an, die dann an USR2071N übergeben werden.

Nachdem Sie die Arbeit an der RPC-Verbindung beendet haben, müssen Sie sich über RPC-CNTX mit dem Funktionscode 'LF' beim EntireX Broker abmelden.

15 APIs zur Verwendung beim Natural RPC

Der Zweck von Natural-Anwendungsprogrammierschnittstellen (APIs) ist es, Informationen abzurufen oder zu ändern oder Dienste zu nutzen, die nicht über Natural-Statements zugänglich sind. Die folgenden in der Natural Library SYSEXT verfügbaren Anwendungsprogrammschnittstellen sind für die Verwendung beim Natural RPC vorgesehen. Einträge, die mit einem Link versehen sind, entsprechen APIs, die in der Natural-Dokumentation dokumentiert sind.

Darüber hinaus können Sie eine Beschreibung zu einer API in der Natural Library SYSEXT unter dem Namen USR*nnnn*T finden.

Eine Erläuterung der Natural-Objekttypen, die normalerweise zu jeder API verfügbar sind, können Sie der Dokumentation der Natural Utility SYSEXT entnehmen.

API	Zweck
USR1071N	Anmeldedaten für RPC-Server festlegen: Benutzerkennung, Passwort und Ticket-Kriterien für Natural RPC. Senden von Anmeldedaten an den Natural RPC Server, wenn der Client kein NSC hat, oder wenn der Client NSC hat: für die Verwendung einer anderen Benutzerkennung und eines anderen Passworts auf der Server-Seite. Siehe *Verwendung von Security*.
USR2007N	Abrufen oder Setzen von Daten für den RPC-Standardserver, einschließlich der Logon-Option. Festlegen einer Standard-Serveradresse, die immer dann verwendet werden soll, wenn ein entferntes Programm nicht über das Service Directory angesprochen werden kann. Siehe *Festlegen einer Standard-Serveradresse innerhalb einer Natural-Sitzung* in *Betrieb einer Natural RPC-Umgebung* .
USR2032N	Unterstützung des Commits bei einem CLOSE CONVERSATION-Statement auf der Client-Seite. Beim Aufruf dieser API wird am Ende der einzelnen Konversation ein implizites END TRANSACTION ausgelöst. Siehe *Datenbank-Transaktionen* im Kapitel *Einführung in Natural RPC*, Abschnitt *Konversationeller CALLNAT*.

API	Zweck
USR2035N	Abrufen oder Festlegen des erforderlichen SSL-Parameterstring, wenn Secure Socket Layer (SSL) für die TCP/IP-Kommunikation mit dem EntireX Broker verwendet wird. Siehe *Secure Socket Layer verwenden* in *Betrieb einer Natural RPC-Umgebung* .
USR2071N	Unterstützung von EntireX Security auf der Client-Seite. Anmeldung beim EntireX Broker. Muss im Falle von Nicht-Natural-Security-Clients aufgerufen werden, um Anmeldedaten anzugeben, die dann an den Server übergeben werden. Siehe *Natural RPC mit Natural Security verwenden* in *Verwendung von Security*.
USR2072N	Unterstützung von EntireX Security auf der Server-Seite. Festlegen eines Passworts, das für die Anmeldung (LOGON) in Verbindung mit dem Schlüsselwort-Subparameter SRVUSER des Profilparameters RPC oder dem Parameter-Makro NTRPC verwendet wird. Siehe *Server-Seite* in *Verwendung von Security*, Abschnitt *Natural RPC mit EntireX Security verwenden*.
USR2073N	Anpingen oder Beenden eines RPC-Servers aus Ihrer Anwendung heraus. Siehe *Anwendungsprogrammierschnittstelle USR2073N verwenden* in *Beenden eines Natural RPC Servers*.
USR2074N	Ändern des Natural Security-Passworts auf dem RPC-Server über eine Natural RPC-Service-Anfoderung. Siehe *Passwort ändern* in *Verwendung von Security*.
USR2075N	Beenden eines EntireX-Broker-Dienstes aus Ihrer Anwendung heraus. Eine erweiterte Version der API USR2075N ist die API USR8208N. Siehe *Anwendungsprogrammierschnittstelle USR2075N* in *Beenden eines Natural RPC Servers*.
USR2076N	Abrufen oder Festlegen des RPC TIMEOUT-Werts. Siehe *TIMEOUT-Wert mit API USR2076N abfragen/setzen*.
USR4008N	Auf der Client-Seite: Übergeben eines alternativen Library-Namens an den Natural RPC Server zwecks Anmeldung. Siehe *Logon-Option benutzen* in *Betrieb einer Natural RPC-Umgebung* .
USR4009N	Abrufen oder Festlegen von Parametern für EntireX in einer Natural RPC-Client- oder -Server-Umgebung. Siehe *Parameter für EntireX setzen/abfragen* in *Betrieb einer Natural RPC-Umgebung* .
USR4010N	Auf der Client-Seite: Abrufen der Laufzeiteinstellungen eines Servers. Siehe *Laufzeiteinstellungen eines Servers abrufen* in *Betrieb einer Natural RPC-Umgebung* .
USR4012N	Anwendungsfehler beim RPC-Server setzen.

API	Zweck
USR4371N	Auf der Client-Seite: Festlegen der Benutzerkennung und ETID für Natural RPC Server, die mit Impersonation = A (automatische Anmeldung) konfiguriert wurden. Wenn für den Natural RPC Server Impersonation ohne Passwortüberprüfung aktiv ist, kann der Client optional eine ETID an den Natural RPC Server übergeben.
USR6304N	Setzen oder Abrufen des Modus für Reliable Natural RPC. Siehe ***Reliable RPC auf der Natural RPC-Client-Seite*** in ***Reliable RPC***.
USR6305N	Commit oder Rollback einer Arbeitseinheit (Unit of Work), die mit CLIENT_COMMIT erstellt wurde. Diese API ist erforderlich, wenn der Status des Reliable RPC auf client commit gesetzt wurde. Siehe ***Reliable RPC auf der Natural RPC-Client-Seite*** in ***Reliable RPC***.
USR6306N	Abrufen des Status aller Reliable RPC-Nachrichten des Benutzers, der gerade am EntireX Broker angemeldet ist. Siehe ***Reliable RPC auf der Natural RPC Server-Seite*** in ***Reliable RPC***.
USR8208N	Erweiterte Version der API USR2075N. Beendet einen EntireX-Broker-Dienst. Pingt einen RPC-Server an oder beendet einen EntireX Broker-Dienst. Zeigt die EntireX Broker-Version und andere Informationen aus dem EntireX Broker Command and Information Service (CIS) an. Siehe ***Anwendungsprogrammierschnittstelle USR8220N verwenden*** in ***Beenden eines Natural RPC Servers***.
USR8220N	Auslösen der Beendigung des Natural RPC-Servers auf der Server-Seite. Siehe ***Anwendungsprogrammierschnittstelle USR8220N verwenden*** in ***Beenden eines Natural RPC Servers***.
USR8225N	Zugang zur Pflege der EntireX-Parameter MESSAGE_ID und CORRELATION_ID in einer Natural RPC Client- oder Server-Umgebung zur Verfügung stellen.

Beachten Sie, dass die RPC-spezifischen APIs alle Werte auch im gemischten Modus akzeptieren. Eine Umsetzung in Großbuchstaben findet nur statt, wenn das Beispielprogramm USR*nnnn*P (Quellcode-Objekt) verwendet wird, um das entsprechende Subprogramm USR*nnnn*N aufzurufen. Ausnahme: Alle USR*nnnn*P-Programme, die mit Passwörtern arbeiten, bieten die Möglichkeit, die Passwörter in gemischter Schreibweise einzugeben.

Stichwortverzeichnis

U