Natural for Mainframes

Natural リモートプロシージャコール（RPC）

バージョン 4.2.5

October 2009

Natural

This document applies to Natural バージョン 4.2.5 and to all subsequent releases.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

目次

1 Natural リモートプロシージャコール（RPC）

リモートプロシージャコール（RPC）技法は、同一コンピュータ上に配置することが可能であるか、または同一あるいは異種のマシンおよびオペレーティングシステムのネットワークに基づいた、サーバーシステムとクライアントシステム間の通信のフレームワークを確立します。 いくつかの基本的に類似したメソッドが知られています。

このドキュメントでは、設計を可能にし、分散ソフトウェアシステムのアプリケーションを単純化するために、Natural により提供される RPC 技法の使用および動作のセオリーについて説明します。 Natural RPC ベースの環境に関係する可能性がある他の製品の詳細については、EntireX RPC for 3GL、Entire Network、EntireX Broker のドキュメントを参照してください。

リモートプロシージャコールを管理するために提供されている機能の詳細については、Natural 『*SYSRPC ユーティリティ*』ドキュメントを参照してください。

このドキュメントは次の項目で構成されています。

	Natural RPC について	非会話型および会話型モードでの Natural RPC の動作など、基本的な情報について説明します。クライアントおよびサーバー側のデータベーストランザクションについて説明し、SYSRPC ユーティリティおよび Natural RPC ドキュメントで使用される重要な用語のリストを示します。
	前提条件および予備情報	一般的な前提条件の概要、Natural リモートプロシージャコール（RPC）環境を実装するために Natural で使用できる機能の概要、および Software AG IDL のデータタイプ、グループ、配列、および構造と Natural プログラミング言語との間の専用のマッピングについて説明します。
	制限および制約事項	Natural RPC を使用するときに注意する必要がある制限、および制約事項について説明します。
	Natural RPC 環境の設定	Natural RPC 環境を設定するために、すべてのクライアントおよびサーバー Natural に対して実行する必要がある基本的な手順について説明します。

Natural RPC サーバーの起動	さまざまなプラットフォームで Natural RPC サーバーを起動する方法。
Natural RPC サーバーの終了	Natural RPC サーバーを終了するさまざまな方法について説明します。
EntireX Broker サービスの終了	EntireX Broker サービスを終了するさまざまな方法について説明します。
Natural RPC 環境の運用	Natural RPC 環境を運用する方法。
会話型 **RPC** の使用	Natural RPC を会話型モードで使用する方法。会話を開く／閉じる、会話コンテキストの定義、複数の会話が並行して使用される場合のシステム変数 `*CONVID` の変更について説明します。
Reliable RPC	信頼性の高いメッセージングシステムの Natural RPC 実装である Reliable RPC について説明します。
リモートディレクトリサーバー（**RDS**）の使用	RDS の原則および使用方法について説明します。RDS インターフェイス、リモートディレクトリサービスルーチンの作成方法。ディレクトリ情報をワークファイルから読み取るために必要な RDS ディレクトリサービスプログラム `RDSSCDIR` に関する情報。
Security の使用	Natural RPC を Natural Security または EntireX Security とともに使用する方法。
EntireX Broker サポート	EntireX Broker サポートに関する特別な考慮事項について説明します。

注意: このドキュメントは、Natural を使用できるすべてのプラットフォームに適用されます。 ただし、現在使用している Natural ドキュメントセットに応じて、次のような違いがあります。

- Natural ユーティリティ `SYSRPC` の使用例では、GUI または CUI インターフェイスを使用した、プラットフォーム固有のマップが示されます。
- Natural for Windows、UNIX、および OpenVMS では、RPC 固有のパラメータをプロファイルパラメータとして使用できます。
- Natural for Mainframes では、RPC 固有のパラメータをプロファイルパラメータ `RPC` またはパラメータマクロ `NTRPC` のキーワードサブパラメータとして使用できます。

2 Natural RPC について

全般的な情報

目的

Natural RPC 機能によって、クライアント Natural プログラムでサーバー Natural のサブプログラムを呼び出すための CALLNAT ステートメントを発行できるようにします。 Natural クライアントおよびサーバーセッションは、同じまたは異なるコンピュータ上で実行します。 例えば、Windows コンピュータ上の Natural クライアントプログラムは、メインフレームデータベースからデータを取得できるよう、メインフレームサーバーに対して CALLNAT ステートメントを発行できます。 同じ Windows コンピュータが、サーバーとしての役割を果たすことができます。 例えば、UNIX 環境で実行している Natural クライアントプログラムが、このサーバー Natural からデータを要求する CALLNAT ステートメントを発行する場合です。

Natural リモートプロシージャコールの利点

Natural RPC は、クライアントサーバーコンピューティングの利点を利用します。 一般的なシナリオでは、Windows クライアントコンピュータ上の Natural が、メインフレームコンピュータ上の Natural のサーバーデータにミドルウェア層を使用してアクセスします。 このことには次の利点があります。

- クライアント側のエンドユーザーは、グラフィカルユーザーインターフェイスを備えた Natural アプリケーションを使用できます。
- 大規模データベースには、メインフレームサーバー上でアクセスできます。
- クライアントからサーバーに適切なデータだけを送信および返信するときには、ネットワークトラフィックを最小限に押さえることができます。

Natural RPC 動作モード

Natural リモートプロシージャコールには、次の動作モードがあります。

- 非会話型モード（以降のテキスト内では、特に指定がない限り、このモードを意味します）
- 会話型モード

これらのモードについて、以降のセクションで詳しく説明します。 これらのモードの長所および短所の比較については、「*会話型モードと非会話型モード*」を参照してください。

各種プラットフォームでの可用性

次のオペレーティングシステム環境下の各種プラットフォーム上で Natural RPC を使用することができます。

メインフレーム環境

- z/OS
- z/VSE
- z/VSE
- VM/CMS
- BS2000/OSD

メインフレーム上の Natural RPC は、次の TP モニタ環境下でサポートされます。

- Com-plete
- CICS
- IMS/TM
- TSO
- UTM

また、バッチモードでも有効です。

その他の環境

- Windows
- UNIX
- OpenVMS

これらの全プラットフォーム上で、Naturalはクライアントおよびサーバーの両方の役割を果たすことができます。

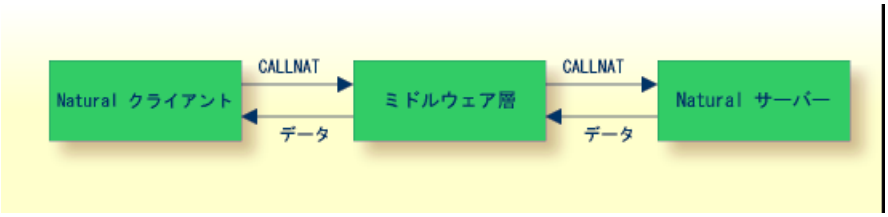
Natural 以外のサポート（EntireX RPC）

Natural以外（3GLおよび他のプログラミング言語）が、クライアント側およびサーバー側でサポートされます。 Natural 以外のクライアントは Natural RPC サーバーと通信が可能であり、Natural クライアントは Natural 以外の RPC サーバーと通信が可能です。 これらは、EntireX RPC の使用により有効となります。

非会話型モードでの Natural RPC の動作

非会話型モードは、1パートナーとのデータの単一交換を達成するためにだけ使用します。「会話型モードと非会話型モード」も参照してください。

ローカルおよびリモートのサブプログラムコールを並行して発行できるようにするため、Natural RPC 技法は Natural ステートメント CALLNAT を使用します。 リモートプログラムコールは同時に動作します。 リモートプロシージャコールでは、CALLNAT は単純に次のルートで通信します。



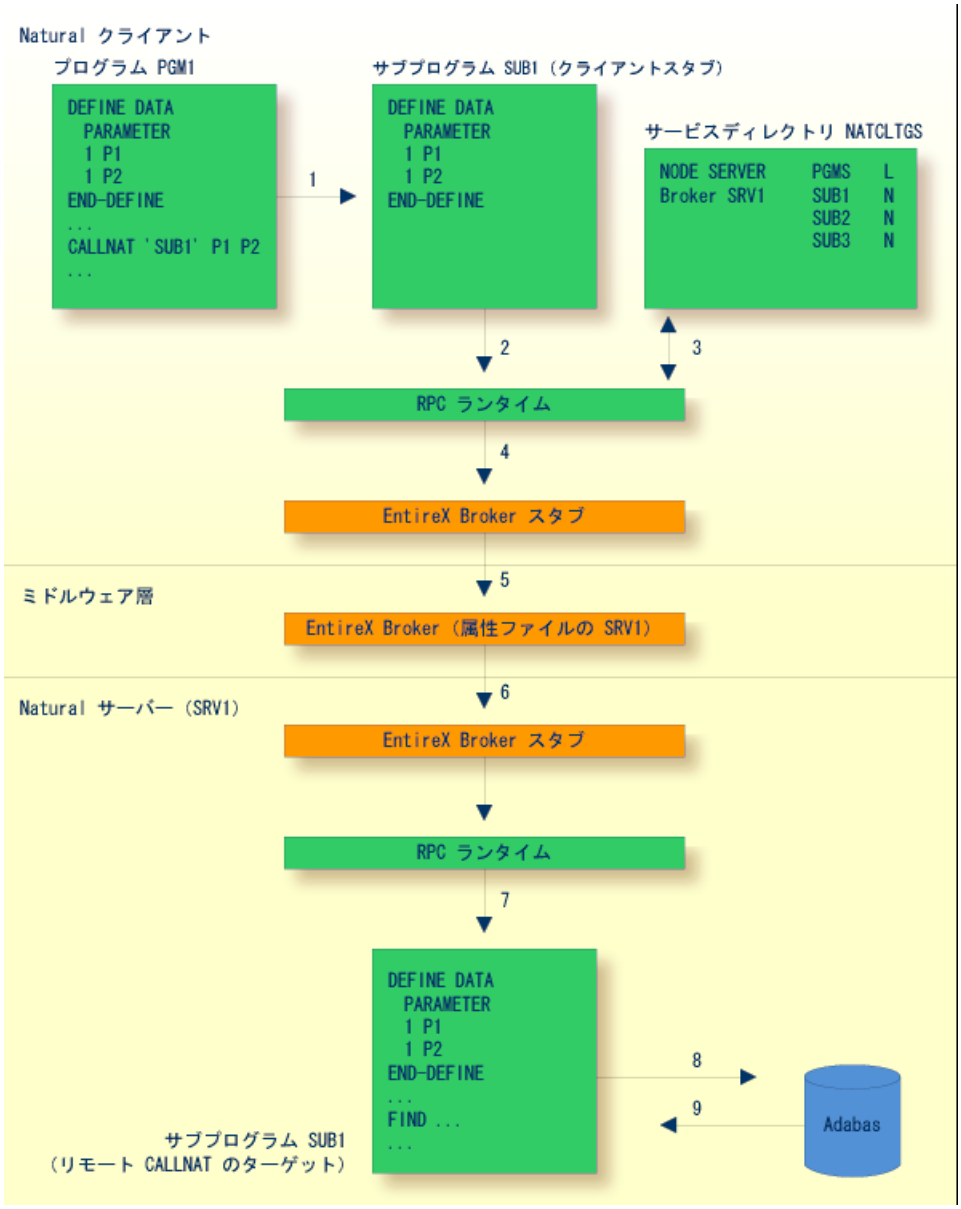
Natural クライアントから発行された CALLNAT は、ミドルウェア層経由で、クライアントにデータを送り返す Natural サーバーに進みます。

通常、ミドルウェア層は、ACI プロトコルを使用する Software AG 製品 EntireX Broker で構成されています。 EntireX Broker は、通信層として Entire Net-Work または TCP/IP のいずれかを使用します。

RPC 制御フローの詳細な例について、以降で説明します。

RPC 環境での CALLNAT の発行

リモートプロシージャでの CALLNAT 制御フローの詳細は、次のとおりです。 より明確にするために戻りパスは示していませんが、説明を参照する番号を示しています。



1. Natural クライアントから、プログラム PGM1 がサブプログラム SUB1 へ CALLNAT を発行します。 PGM1 は、その CALLNAT がローカルまたはリモート CALLNAT のどちらになるのかは認識しません。

   ターゲット SUB1 はサーバー上に存在するため、CALLNAT は代わりにスタブサブプログラム（インターフェイスオブジェクト）SUB1 にアクセスします。このクライアントスタブサブプログラムは、自動的に作成されたか、または SYSRPC ユーティリティのスタブ生成機能（SG）を使用して手動で作成されたものです。

   スタブはターゲットサブプログラムと同じ名前を持ち、プログラム PGM1 およびサーバー上のターゲットサブプログラム SUB1 で使用されるパラメータと同一のパラメータを含みます。RPC が内部的に使用する制御情報も含まれています。

Natural プロファイルパラメータ RPC またはパラメータマクロ NTRPC の キーワードサブパラメータ AUTORPC が ON に設定されており、Natural がローカル環境内でサブプログラムを検出できない場合は、Naturalはこのサブプログラムをリモートプロシージャコールであると解釈し、ランタイム時にダイナミックにパラメータデータエリア（PDA）を生成します。

Natural は、サービスディレクトリ NATCLTGS 内でもこのサブプログラムを検出しようとします。

SYSRPC のスタブ生成機能の詳細については、「*スタブサブプログラムの作成*」を参照してください。

スタブなしで操作する場合は、「***Natural RPC** 自動実行の操作*」を参照してください。

2. 次に、スタブは RPC クライアントサービスルーチンへの CALLNAT を設定します。
3. クライアント RPC ランタイムは、サービスディレクトリ NATCLTGS で、どのノードおよびサーバー上で CALLNAT を実行する必要があるか、およびログオンが必要かどうかを確認します。

   パラメータリストを含む CALLNAT データおよび必要な場合はログオンデータがミドルウェア層へ渡されます。
4. この例では、このミドルウェア層は Software AG 製品 EntireX Broker で構成されています。したがって、CALLNAT データは、最初にクライアント上の EntireX Broker スタブへ渡されます。
5. EntireX Broker スタブから、CALLNAT データが EntireX Brokerへ 渡されます。 EntireX Broker は、次の環境に配置できます。
   - クライアントコンピュータ
   - サーバーコンピュータ
   - 第 3 プラットフォーム

   データを正常に送信するには、EntireX Broker で登録されているサーバー SRV1 は EntireX Broker 属性ファイル内に定義され、SRV1 はすでに稼動している必要があります。

   EntireX Broker 属性ファイル内のサーバー定義については、EntireX Broker ドキュメントを参照してください。
6. CALLNAT データは、ミドルウェア層から Natural サーバープラットフォーム上の EntireX Broker スタブへ渡され、そこから RPC サーバーサービスルーチンへ渡されます。

   RPC サーバーサービスルーチンは、（存在する場合は）ログオンデータを確認し、（要求された場合は）ログオンを実行します。
7. RPC サーバーサービスルーチンは、ターゲットサブプログラム SUB1 を呼び出し、要求された場合にデータを渡します。

   この時点では、ターゲットサブプログラム SUB1 には、ローカルプログラム PGM1 により呼び出された場合と同じように実行するために必要な全データが含まれています。

8. その後、例えば、サブプログラム SUB1 はサーバーの Adabas データベースへ FIND ステートメントを発行できます。 SUB1 は、ローカルまたはリモート CALLNAT のいずれで起動されたかは認識しません。
9. Adabas はデータを検出（FIND）し、そのデータを SUB1 へ渡します。

   次に、SUB1 は、Adabas データを呼び出し元のサーバーサービスルーチンに返します。 そこから、データはミドルウェア層を経由して PGM1 に渡されます。 手順 1～8 で説明したルートと同じルートですが、順序は逆です。

会話型モードでの Natural RPC の動作

会話型 RPC は、クライアントとサーバー間の制限された期間のスタティックな接続です。 クライアントによって定義された数多くのサービス（サブプログラム）を提供します。それらはすべて、会話の期間にクライアントが排他的に利用可能な 1 サーバータスク内で実行されます。 OPEN CONVERSATION ステートメントおよび CLOSE CONVERSATION ステートメントを使用してプログラムに実装されます。

複数接続（会話）は同時に存在可能です。 会話 ID でクライアントによって管理され、それぞれが異なるサーバー上で実行されます。 指定された会話に属さないリモートプロシージャコールは、異なるサーバー上の、異なるサーバータスク内で実行されます。

会話中に、サーバー側のリモートサブプログラム間で、コンテキストエリアと呼ばれるデータエリアを定義して共有できます。 詳細については、Natural『ステートメント』ドキュメントの「*Natural RPC 用のコンテキスト変数の定義*」を参照してください。

会話はローカルまたはリモートになります。

例：

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

両方のサブプログラム（S1 および S2）には、同じ場所（ローカルまたはリモート）でアクセスする必要があります。 1 会話内でローカルおよびリモート CALLNAT を混在させることはできません。 サブプログラムがリモートで実行されると、両方のサブプログラムが同じサーバータスクによって実行されます。

非会話型 RPC CALLNAT と同様に、会話は最初にローカルで記述およびテストし、その後サーバーへ転送できます。

ローカル／リモートサブプログラム実行に対する一般ルール

ローカルサブプログラムの実行

サブプログラムをローカルに実行する場合は、次のルールが適用されます。

- サブプログラムによって、会話のメンバである別のサブプログラムを呼び出すことはできません。

OPEN CONVERSATION ステートメントにリストされていない他のサブプログラムを呼び出すことはできます。 ただし、それらは非会話型モードで実行されます。

リモートサブプログラムの実行

サブプログラムをリモートで実行する場合は、次のルールが適用されます。

- サブプログラム S1 によって、会話のメンバである別のサブプログラム S2 を呼び出すことができます。

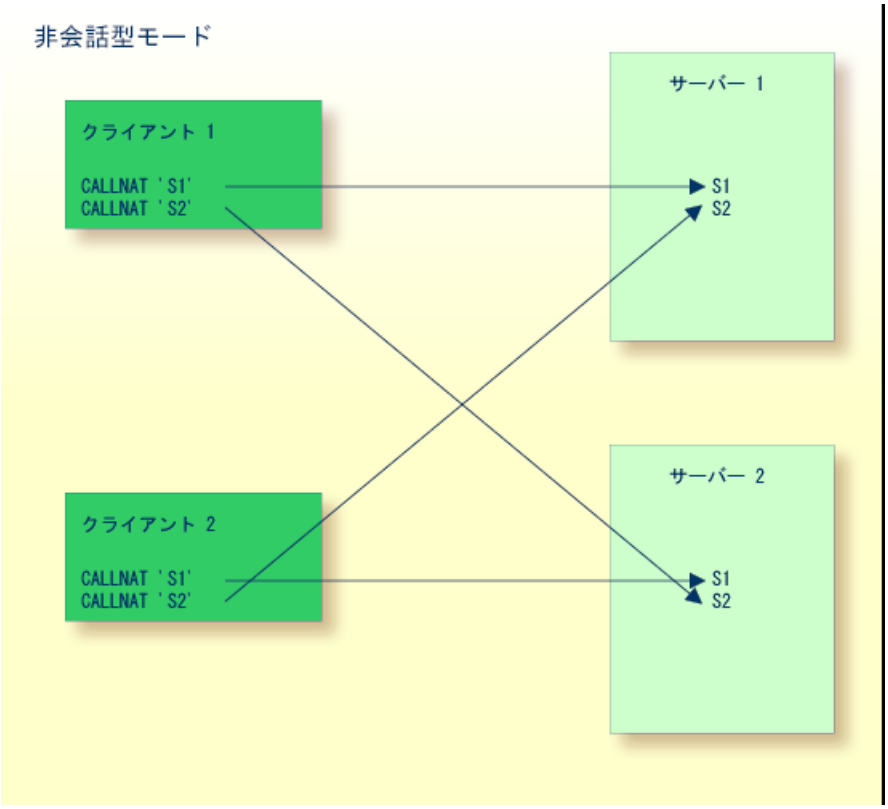
間接的に呼び出されたため、この CALLNAT は非会話型モードで実行されます。 したがって、サブプログラム S2 はコンテキストエリアへのアクセス権を持ちません。

会話型モードと非会話型モード

非会話型モードで複数クライアントが複数サーバーにアクセスするクライアントサーバー環境では、異なるクライアントからの同一の CALLNAT 要求が同じサーバー上で実行されるという問題が発生することがあります。

これは、次のことを意味します。例えば、クライアント 1 からの CALLNAT 'S1' がサーバー 1 上のサブプログラム S1 を実行します。この場合、S1 はデータベースにレコードを書き込みます。クライアント 1 のトランザクションがまだ完了していない（END TRANSACTION なし）ときに、クライアント 2 もサーバー 1 に CALLNAT 'S1' を送ります。したがって、クライアント 1 からのデータは上書きされます。 その後、クライアント 1 が CALLNAT 'S2'（END TRANSACTION を意味する）を送る場合、クライアント 1 は、実際にはクライアント 2 の同一の CALLNAT からのデータが保存されたにも関わらず、自身のデータが正常に保存されたものと認識します。

次の図に、2つのクライアントおよび2つのサーバーでのこの事象を示します。このようなシナリオでは、同じサーバー上の同じサブプログラムにアクセスする2つの異なるクライアントからの 2 つの同一 CALLNAT は制御できません。

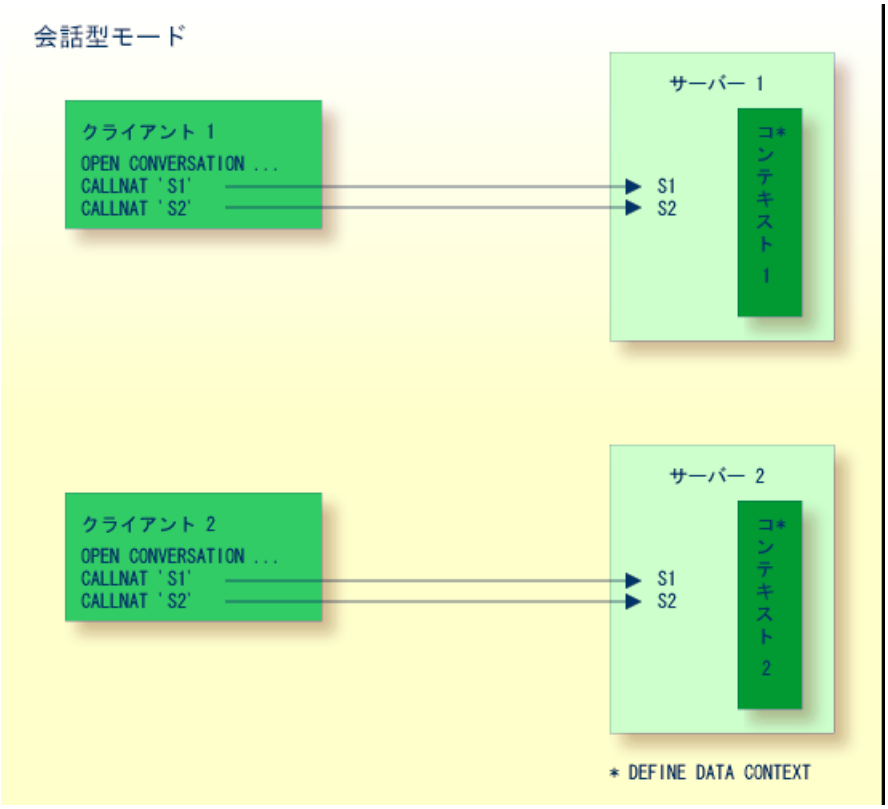


上記の例では、クライアント 1 からの CALLNAT 'S2' は、サーバー 1 およびサーバー 2 のサブプログラム S2 にアクセスする可能性があります。 クライアント 2 からの CALLNAT 'S2' も同じ選択肢を持ちます。

同様に、クライアント 1 からの CALLNAT 'S1' は、サーバー 1 およびサーバー 2 のサブプログラム S1 にアクセスする可能性があり、クライアント 2 からの CALLNAT 'S1' も同じ選択肢を持ちます。

サブプログラムが 1 つのサーバータスクコンテキスト内で実行されるように設計されている場合、インターフェイスが問題となる可能性があることは明白です。

会話型モードでの複雑なRPCトランザクションを定義することによって、非会話型RPCの潜在的な問題を回避できます。



この処理は、会話を開くことによって実行します。 この処理には、クライアント側での OPEN CONVERSATION ステートメントの使用が含まれ、CALLNAT 'S1' および CALLNAT 'S2' を参照します。 このような会話を開くと、1つのサーバータスク全体（例：サーバー1）が確保され、この会話が閉じられる前に、他のリモート CALLNAT がこのサーバー上のこの会話を中断することはありません。 また、DEFINE DATA CONTEXT ステートメントを使用することによって、サーバー側で2つのサブプログラムの共通コンテキストエリアを定義できます。

会話型／非会話型 RPC の使用に対する一般ルール

一般ルールとして、次のことが適用されます。

- サブプログラムの定義済みリストが1コンテキスト内で排他的に実行されることを保証するために、会話型 **RPC** を使用します。
- 各サブプログラムが異なるサーバータスク内で使用可能か、またはトランザクションが複数のサーバーコールに展開しない場合に、非会話型 **RPC** を使用します。 この場合の長所は、長時間に渡るサーバーブロックがなく、比較的少数のサーバータスクのみが必要となることです。

会話型 RPC 使用時の短所

会話型 RPC の短所となりうる点は、サーバータスク全体を確保するため、このサーバー上の他のサブプログラムすべてがブロックされることです。 その結果、他の CALLNAT が必然的に待ち状態になったり、サーバータスクをさらに増やして起動する必要性が出てきます。

データベーストランザクション

クライアント側とサーバー側のデータベーストランザクションは、互いに独立して実行されます。 つまり、サーバー側で実行される END TRANSACTION や BACKOUT TRANSACTION は、クライアント側のデータベーストランザクションに影響せず、逆もまた同様です。

各非会話型 CALLNAT の終了時および各会話の終了時に、暗黙的な BACKOUT TRANSACTION がサーバー側で実行されます。 リモート CALLNAT（複数可）によって行われた変更をコミットするには、次のオプションがあります。

会話型以外の **CALLNAT**

1. CALLNAT を終了する前に明示的な END TRANSACTION を実行します。
2. Natural プロファイルパラメータ ETEOP を ON に設定します。 これにより、各非会話型 CALLNAT の終了時に暗黙的な END TRANSACTION が生じます。

   END TRANSACTION の実行は、パラメータ SRVCMIT の設定に応じて、応答がクライアントに送信される前（SRVCMIT=B）または応答がクライアントに正常に送信された後（SRVCMIT=A）になります。 SRVCMIT=B がデフォルトで、RPC の旧バージョンと互換性があります。

会話型 **CALLNAT**

1. クライアントが会話を終了する前に、サーバーで明示的な END TRANSACTION を実行します。
2. Natural プロファイルパラメータ ETEOP を ON に設定します。 これにより、各会話の終了時に暗黙的な END TRANSACTION が生じます。

   END TRANSACTION の実行は、パラメータ SRVCMIT の設定に応じて、応答がクライアントに送信される前（SRVCMIT=B）または応答がクライアントに正常に送信された後（SRVCMIT=A）になります。 SRVCMIT=B がデフォルトで、RPC の旧バージョンと互換性があります。
3. CLOSE CONVERSATION ステートメントを実行する前に、クライアント側でアプリケーションプログラミングインターフェイス USR2032N を呼び出します。 これにより、個々の会話の終了時に暗黙的な END TRANSACTION が生じます。

会話の場所

両方のサブプログラム S1 および S2（上図に表示）には、同じ場所（ローカルまたはリモート）でアクセスする必要があります。 1 会話内にローカルおよびリモートの CALLNAT を混在させることはできません。 サブプログラムがリモートで実行されると、両方のサブプログラムが同じサーバータスクによって実行されます。

Natural RPC の用語

次の表は、SYSRPC ユーティリティおよび Natural RPC ドキュメントで使用される重要な用語の概要です。

用語	説明
クライアントスタブ	クライアント側で CALLNAT 要求を受け入れ、渡されたパラメータを整列し、Natural RPC ランタイムおよびトランスポート層を介してリモートサーバーにデータを転送します。結果を整列解除して呼び出し元に返します。 クライアントスタブはローカルサブプログラムであり、このサブプログラムを経由して、サーバーサブプログラムが呼び出されます。 クライアントスタブは、対応するサーバーサブプログラムと同じ名前を持ち、同じパラメータを含みます。
EntireX Broker スタブ	Natural RPC ランタイムと、クライアントおよびサーバー間で整列されたデータを交換する EntireX Broker トランスポート層の間のインターフェイス。
偽装	偽装の前提は、Natural RPC サーバーが実行されているオペレーティングシステムへのアクセスが、SAF 対応の外部セキュリティシステムによって制御されていることです。 ユーザー認証は、この外部セキュリティシステムによって実行されます。 偽装とは、認証が成功してユーザーの識別が確立された後、後続の権限チェックがこの識別に基づいて実行されることを意味します。 このことには、外部リソース（データベースやワークファイルなど）へのアクセスの権限チェックが含まれています。 認証が成功した後、ユーザーは「自分の識別を変更」できません。つまり、異なるユーザー ID を使用できません。 「**Security** の使用」の「偽装」を参照してください。
インターフェイスオブジェクト	EntireX の旧バージョンでは、用語「スタブ」が、リモートプロシージャコールを発行および受信するための、Workbench で生成される、アプリケーション依存のコード部分の意味でも使用されていました。 現在、このようなオブジェクトはインターフェイスオブジェクトと呼ばれています。
NATCLTGS	サービスディレクトリ（下記参照）を実装するために SYSRPC ユーティリティで生成された Natural サブプログラムの名前。
ノード名	リモート CALLNAT の送信先であるノードの名前。 例えば、EntireX Broker 経由の通信の場合には、ノード名は EntireX Broker 属性ファイルのフィールド BROKER-ID に定義された EntireX Broker の名前です。

RPC パラメータ	Natural RPC を制御するために使用できるすべてのパラメータについては、Natural の『パラメータリファレンス』ドキュメントに詳細に記載されています。「*プロファイルパラメータ*」セクションを参照してください。 RPC パラメータは、パラメータマクロ NTRPC に含まれるか（スタティック定義）、プロファイルパラメータ RPC で定義されます（ダイナミック定義）。 「*RPC - リモートプロシージャコールの設定*」および「*NTRPC マクロの構文*」（Natural『*パラメータリファレンス*』ドキュメント）を参照してください。
サービスディレクトリ	サービスディレクトリは、サーバーが提供するサービス（サブプログラム）の情報を含みます。 サービスディレクトリは、各クライアントノードからローカルでアクセスするか、またはリモートディレクトリサーバーに配置されている場合は、プロファイルパラメータ RPC またはパラメータマクロ NTRPC のキーワードサブパラメータ RDS で参照できます。
サーバー名	CALLNAT を実行するサーバーの名前。 EntireX Broker 経由の通信の場合には、サーバー名は、EntireX Broker 属性ファイルのフィールド SERVER に定義された名前です。
サーバータスク	サービス（サブプログラム）を提供する Natural タスクです。 これは、通常、バッチタスクまたは非同期タスクです。 これはサーバー名で識別されます。

3 前提条件および予備情報

このドキュメントでは、一般的な前提条件の概要を説明し、Naturalリモートプロシージャコール（RPC）環境を実装するために Natural で使用できる機能について簡単に説明します。

関連製品

RPC 環境を異なるプラットフォームに実装する場合、Natural for Mainframes、Windows、UNIX、またはOpenVMSの対応する最新バージョンが必要になります。また、次の必須またはオプションの製品、サブ製品、および機能を Natural RPC 環境で使用できます。

製品	目的
EntireX Communicator（EntireX Broker）	Software AG 製品 EntireX Communicator の EntireX Broker は、通常、Natural クライアントとNaturalサーバー間のミドルウェア層を確立します。ACIプロトコルを使用し、適切なクライアント／サーバースタブで構成されます。
Entire Net-Work	この Software AG 製品は、EntireX Broker が使用するトランスポートメソッドが Entire Net-work である場合に必要です。これは、推奨されるトランスポートメソッドです。
TCP/IP	EntireX Broker が使用するトランスポートメソッドが TCP/IP である場合に必要です。 「*Natural RPC 環境の設定*」の「*トランスポートメソッドとしての **TCP/IP** の使用*」も参照してください。
EntireX 開発者キット	EntireX Communicator製品に含まれているこのキットは、Natural以外のプログラミング言語サポートのために必要です。
ディレクトリサービス	リモートディレクトリサーバー（RDS）を使用すると、そのサービスを RPC 環境内の全クライアントが使用できるよう、1 つの場所にディレクトリ定義を定めることが可能となります。 RDSは、EntireX Brokerが提供するロケーショントランスペアレンシを使用する場合に必要です。
Natural Security	オプション。 このSoftware AGアドオン製品は、クライアントでセキュリティがアクティブな場合に、Natural RPC サーバーおよびサービスを保護するためにサーバー側で必要です。その逆も同様です。
EntireX RPC	オプション。 Natural RPC は、EntireX RPC for 3GL を完全にサポートします。EntireX RPC は、EntireX Communicator 製品に含まれています。
EntireX Security	オプション。 Natural RPC は、クライアント側およびサーバー側の EntireX Security を完全にサポートします。EntireX Security は、EntireX Broker 製品に含まれています。

サポートされる Software AG 製品のバージョンについては、Natural for Mainframes の最新リリースノートの「*Natural および他の Software AG 製品*」を参照してください。

Natural RPC ベースの環境に関係する可能性がある他の製品の詳細については、該当する製品のドキュメントを参照してください。

関係する Natural ステートメント

次の Natural ステートメントが、Natural RPC 環境の作成に使用されます。

- CALLNAT
- DEFINE DATA PARAMETER
- DEFINE DATA CONTEXT
- OPEN CONVERSATION
- CLOSE CONVERSATION
- PASSW
- STACK

セクション「*制限および制約事項*」の「***Natural** ステートメントの動作*」および「*サーバーに対する **Natural** ステートメントの注意事項*」には、これらのステートメントを Natural RPC 環境で使用するときに、これらのステートメントの異常な動作について知っておく必要があることが記載されています。

Natural RPC で使用する Natural ユーティリティ

次の Natural ユーティリティが、Natural RPC 環境の作成および管理に使用されます。

- SYSRPC
  このユーティリティは、リモートプロシージャコール環境を管理するために使用されます。
- SYSEXT

  このユーティリティは、現在のシステムライブラリ SYSEXT に含まれる Natural アプリケーションプログラミングインターフェイス（API。以下を参照）を配置およびテストするために使用されます。
- SYSPARM

  メインフレームでは、このユーティリティは、プロファイル名を付けて保存される Natural プロファイルパラメータセットを作成および管理するために使用されます。
- コンフィグレーションユーティリティ

Windows、UNIX、およびOpenVMSでは、このユーティリティは、グローバルおよびローカルコンフィグレーションファイルを変更し、パラメータファイルを作成および変更するために使用されます。

Natural RPCで使用するアプリケーションプログラミングインターフェイス

Naturalアプリケーションプログラミングインターフェイス（API）の目的は、Naturalステートメントではアクセス不可能な情報の検索や修正、またはサービスを使用することにあります。

Naturalライブラリ SYSEXT で使用可能な次のアプリケーションプログラミングインターフェイスは、Natural RPCでの使用が想定されています。

API	目的
USR1071N	Natural RPCのユーザーID、パスワード、およびチケット基準を設定できます。 「**Security** の使用」を参照してください。
USR2007N	リモートプログラムをサービスディレクトリ経由でアドレスできないときに使用されるデフォルトのサーバーアドレスを指定できます。 「***Natural RPC* 環境の運用**」の「***Natural* セッション内のデフォルトサーバーアドレスの指定**」を参照してください。
USR2032N	クライアント側での CLOSE CONVERSATION ステートメントのコミットをサポートします。呼び出されると、このAPIにより、個々の会話の終了時に暗黙的な END TRANSACTION が生じます。 「***Natural RPC* について**」の「*データベーストランザクション*」の「*会話型CALLNAT*」セクションを参照してください。
USR2035N	EntireX BrokerへのTCP/IP通信のために Secure Socket Layer（SSL）が使用される場合に、必要なSSLパラメータ文字列を設定できます。 「***Natural RPC* 環境の運用**」の「***Secure Socket Layer* の使用**」を参照してください。
USR2071N	Natural Security以外のクライアントの場合に、サーバーに渡されるログオンデータを指定するために呼び出す必要があります。 「***Security* の使用**」の「***Natural Security* での *Natural RPC* の使用**」を参照してください。
USR2072N	プロファイルパラメータ RPC またはパラメータマクロ NTRPC のキーワードサブパラメータ SRVUSER とともに LOGON に使用されるパスワードを指定できます。 「***Security* の使用**」の「*サーバー側*」の「***EntireX Security* での *Natural RPC* の使用**」セクションを参照してください。

API	目的
USR2073N	アプリケーション内から RPC サーバーを ping または終了できます。 「*Natural RPC サーバーの終了*」の「*アプリケーションプログラミングインターフェイス USR2073N の使用*」を参照してください。
USR2074N	Natural RPC サービス要求によって、RPC サーバーの Natural Security パスワードを変更できます。 「*Security の使用*」の「*パスワードの変更*」を参照してください。
USR2075N	アプリケーション内から EntireX Broker サービスを終了できます。 「*EntireX Broker サービスの終了*」の「*アプリケーションプログラミングインターフェイス USR2075N の使用*」を参照してください。
USR4008N	クライアント側で、サーバーがログオンするライブラリの代替名を指定できます。 「*Natural RPC 環境の運用*」の「*ログオンオプションの使用*」を参照してください。
USR4009N	Natural RPC クライアントまたはサーバー環境の EntireX のパラメータを設定／取得できます。 「*Natural RPC 環境の運用*」の「*EntireX のパラメータの設定／取得*」を参照してください。
USR4010N	クライアント側で、サーバーのランタイム設定を取得できます。 「*Natural RPC 環境の運用*」の「*サーバーのランタイム設定の取得*」を参照してください。
USR6304N	信頼性の高い Natural RPC の信頼性状態を設定／取得できます。 「*Reliable RPC*」の「*Natural RPC クライアント側の Reliable RPC*」を参照してください。
USR6305N	Reliable RPC メッセージをコミットまたはロールバックできます。 この API は、Reliable RPC の状態が 「クライアントコミット」 に設定されている場合に必要です。 「*Reliable RPC*」の「*Natural RPC クライアント側の Reliable RPC*」を参照してください。
USR6306N	現在 EntireX Broker にログオンしているユーザーの、すべての Reliable RPC メッセージのステータスを取得できます。 「*Reliable RPC*」の「*Natural RPC サーバー側の Reliable RPC*」を参照してください。

RPC 固有の API は、すべての値を混在モードでも受け入れます。 大文字変換は、プログラム例 USR*nnnn*P（ソースオブジェクト）が、対応するサブプログラム USR*nnnn*N を呼び出すために使用される場合にのみ行われます。 *例外：*パスワードを処理するすべての USR*nnnn*P プログラムには、大文字小文字混在モードでパスワードを入力するオプションがあります。

各 API に一般的に指定される Natural オブジェクトタイプについては、Natural ユーティリティ SYSEXT を参照してください。

Software AG IDL から Natural への対応関係

このセクションでは、Software AG IDL のデータタイプ、グループ、配列、および構造の、Natural プログラミング言語への対応関係について説明します。（*EntireX* ドキュメントの）「Software AG IDL File」に記載されている、すべての言語バインディングについて有効な、IDL データタイプに関する補記やヒントも参照してください。

Software AG IDL データタイプから Natural データフォーマットへの対応関係

下の表では、IDL に対して、次のメタ記号およびプレースホルダが使用されています。

- メタ記号 [および] で囲まれているのはオプションの字句エンティティとなります。
- プレースホルダ *number*（場合によっては *number.number*）は、一連の数字を示します。例：123。

Software AG IDL	説明	Natural データフォーマット	注
A*number*	英数字	A*number*	
AV	英数字変数長	A DYNAMIC	
AV*number*	最大長指定付き英数字変数長	A DYNAMIC	
B*number*	バイナリ	B*number*	
BV	バイナリ変数長	B DYNAMIC	
BV*number*	最大長指定付きバイナリ変数長	B DYNAMIC	
D	日付	D	3,5
F4	浮動小数点（小）	F4	2
F8	浮動小数点（大）	F8	2
I1	整数（小）	I1	
I2	整数（中）	I2	
I4	整数（大）	I4	
K*number*	漢字	A*number*	1

Software AG IDL	説明	Natural データフォーマット	注
KV	漢字変数長	A DYNAMIC	1
KV*number*	最大長指定付き漢字変数長	A DYNAMIC	1
L	論理	L	
N*number*[.*number*]	アンパック小数	N*number*[.*number*]	
NU*number*[.*number*]	符号なしアンパック小数	N*number*[.*number*]	
P*number*[.*number*]	パック小数	P*number*[.*number*]	
PU*number*[.*number*]	符号なしパック小数	P*number*[.*number*]	
T	時刻	T	3,4
*number*U	Unicode	*number*U	
UV	Unicode 変数長	U DYNAMIC	
UV*number*	最大長指定付き Unicode 変数長	U DYNAMIC	

（『*EntireX*』ドキュメントの）「Software AG IDL Data Types」に記載されている、すべての言語バインディングに有効なヒントや制限事項も参照してください。

注意：

1. データタイプKは、RPC固有のデータフォーマットで、Natural言語の一部ではありません。
2. 浮動小数点データタイプが使用された場合は、丸めが原因でエラーが発生する場合があるため、送信側と受信側で値が多少異なることがあります。 このことは、特に、使用している浮動小数点データ表現（IEEE、HFP）がクライアントとサーバーで異なる場合に当てはまります。
3. AD（紀元後の西暦）の日数のカウント。 有効な範囲は 1.1.0001～28.11.2737 です。 数値から 1.1.0001 以降の全日付範囲への対応関係は、ユリウス暦およびグレゴリオ暦に従い、以下のルールを考慮します。

   a. 4 で割り切れる年は閏年です。

   b. 100 で割り切れる年は、次の規則 3 が該当する場合を除き、閏年ではありません。

   c. 400 で割り切れる年は閏年です。

   d. 西暦 1582 年より前は、ユリウス暦の規則 1 が使用されます。 西暦 1582 年以降は、グレゴリオ暦の規則 1、2、および 3 が使用されます。

パック型の数値と実際の日付との関係については、次の表を参照してください。

日付／日付の範囲	値／値の範囲
1.1.0000	0（特別な値 - 日付なし）
未定義の日付	1～364（使用しないでください）
1.1.0001	365
1.1.1970	719527（日付カウント機能の始点）
28.11.2737	999999（上限の日付）

4. AD（紀元後の西暦）の 1/10 秒のカウント。 有効な範囲は 1.1.0001 00:00:00.0～16.11.3168 9:46:39+0.9 秒です。 パック型の数値と実際の時刻との関係については、次の表を参照してください。

時刻／時刻の範囲	値／値の範囲
1.1.0000 00:00:00.0	0（特別な値 - 日付なし）
未定義の時刻	1 - 315359999
1.1.0001 00:00:00.0	315360000
1.1.1970 00:00:00.0	621671328000（時間カウント機能の始点）

5. パック型数値と日時データタイプとの関係は、次のとおりです。

1 日に含まれる 1/10 秒数 = 24*60*60*10 = 864000

時刻数	= 日付数	* 864000
315360000	= 365	* 864000 1.1.0001 00:00:00.0
621671328000	= 719527	* 864000 1.1.1970 00:00:00.0
日付数	= 時刻数	/ 864000
365	= 315360000	/ 864000 1.1.0001
719527	= 621671328000	/ 864000 1.1.1970

ライブラリ名とエイリアスの対応先

IDL ファイルに指定されているライブラリ名は、Natural ではサポートされません。デフォルトで、Natural クライアントはライブラリ名 SYSTEM をサーバーに送信します。 クライアントからサーバーへ SYSTEM 以外のライブラリ名を送信するには、クライアントで次の手順を実行する必要があります。

手順 3.1. クライアントからサーバーへ SYSTEM 以外のライブラリ名を送信するには

1 クライアントでログオンオプションをオンにします。

2 アプリケーションプログラミングインターフェイス USR4008N を呼び出し、ライブラリ名を指定します。このようにしない場合は、現在のライブラリの名前が送信されます。

ライブラリ名の最大長は 8 文字です。

プログラム名とエイリアスの対応先

プログラム名はクライアントからサーバーへ送信されます。 特殊文字は置き換えられません。プログラムのエイリアスはサーバーに送信されません。

生成された Natural インターフェイスオブジェクト（スタブサブプログラム）は同じ名前になります。

RPC サーバーでは、送信された IDL プログラム名が Natural サブプログラムの特定に使用されます。

プログラム名の最大長は 8 文字です。

パラメータ名の対応先

IDL ファイルのパラメータデータ定義に指定されているパラメータ名は、生成された Natural インターフェイスオブジェクト（スタブサブプログラム）の生成名で置き換えられます。

『*EntireX*』ドキュメントのセクション「*Software AG IDL Grammar*」の「*parameter-data-definition*」を参照してください。

固定サイズ配列と境界なし配列の対応先

- IDL ファイルに含まれる固定サイズ配列は、固定サイズの Natural 配列になります。 下限は 1 に設定され、上限は IDL ファイルに指定されている上限に設定されます。

  IDL ファイルに固定サイズ配列を記述して固定境界の配列の添字を参照する構文については、「*array-definition*」（『*EntireX*』ドキュメントの「*Software AG IDL Grammar*」）を参照してください。

- IDL ファイルに含まれる境界なし配列は、Natural X-array になります。 下限は常に固定で 1 に設定されます。

  IDL ファイルに境界なし配列を記述して境界なし配列の添字を参照する構文については、「*array-definition*」（『*EntireX*』ドキュメントの「*Software AG IDL Grammar*」）を参照してください。

注意: Natural 可変配列（Natural 表記 (../1:V)）は、Natural RPC サーバー側で Natural 固定サイズ配列または X-array の代わりに使用できます。 RPC クライアントでは、このような Natural 可変配列を使用して、IDL 固定サイズ配列または IDL 境界なし配列のいずれかを Natural RPC サーバーに渡すことができます。 RPC サーバーでは、可変配列のサイズを変更できません。つまり、配列オカレンス数は変更できず、Natural RPC サーバーは常に同じ数のオカレンスを返します。

グループとピリオディックグループの対応先

IDL ファイルに含まれるグループは、Natural グループになります。 IDL ファイルにグループを作成する構文については、「*group-parameter-definition*」（『*EntireX*』ドキュメントの「*Software AG IDL Grammar*」）を参照してください。

構造の対応先

IDL ファイルに含まれる構造は、Natural グループになります。 IDL ファイルに構造を作成する構文については、「*structure-definition*」（『*EntireX*』ドキュメントの「*Software AG IDL Grammar*」）を参照してください。

方向属性 IN、OUT、INOUT の対応先

IDL 構文では、パラメータを IN パラメータ、OUT パラメータ、または INOUT パラメータ（指定がない場合のデフォルト）として定義できます。この方向指定は、Natural に次のように反映されます。

- `OUT` 属性を持つパラメータは、RPC クライアントから RPC サーバーへ送信されます。 このようなパラメータは参照渡しで指定されます。
- `IN` 属性を持つパラメータは、RPC サーバーから RPC クライアントへ送信されます。 このようなパラメータは参照渡しで指定されます。
- `INOUT` 属性を持つパラメータは、RPC クライアントから RPC サーバーへ送信され、その後、RPC クライアントに返されます。
- 最上位フィールド（レベル1）の方向情報のみが関係します。グループフィールドは、常に親の指定を継承します。 異なる指定は無視されます。

IDL ファイルに属性を記述して direction-attribute を参照する構文については、「*attribute-list*」（『*EntireX*』ドキュメントの「*Software AG IDL Grammar*」）を参照してください。

注意: インターフェイスオブジェクトレイアウトを Natural アプリケーション `SYSRPC` に定義する場合、方向属性 IN および OUT は IDL での記述に対して反転されます。

- `SYSTRPC` での `IN` は IDL での `OUT`
- `SYSTRPC` での `OUT` は IDL での `IN`

ALIGNED 属性の対応先

ALIGNED 属性は、Natural プログラミング言語とは関係ありません。 ただし、Natural クライアントは必要に応じて ALIGNED 属性を RPC サーバーに送信できます。 このことを実行するには、IDL ファイルを基に生成された Natural インターフェイスオブジェクト（スタブサブプログラム）が必要です。

IDL ファイルで属性を記述して aligned-attribute を参照する構文については、「*attribute-list*」（『*EntireX*』ドキュメントの「*Software AG IDL Grammar*」）を参照してください。

プロシージャまたは関数としてのサーバーの呼び出し

IDL 構文で定義できるのはプロシージャのみです。 IDL 構文に関数という概念はありません。関数とは、パラメータの他に値を返すプロシージャです。 プロシージャおよび関数は、クライアントとサーバーとの間で透過的です。つまり、関数を使用するクライアントは、プロシージャとして実装されているサーバーを呼び出すことができ、その逆も可能です。

クライアント側とサーバー側

Natural RPC では関数がサポートされていません。

4 制限および制約事項

このドキュメントでは、Naturalリモートプロシージャコール（RPC）機能を使用するときに注意する必要がある制限および制約事項について説明します。

RPCでサブプログラムを実行する場合、ローカル実行とは異なる点があります。これについて、以降のセクションで説明します。

ユーザーコンテキストの転送

ユーザーIDを除いて、次の例のように、ユーザーコンテキストはサーバーセッションへは転送されません。

- 全クライアントセッションパラメータは変更されないので、サーバー側の実行に影響しません。
- クライアント側でオープンされているトランザクションをサーバーによってクローズすることはできません。逆の場合も同様です。
- クライアントレポート処理およびワークファイル処理は、サーバー側で継続できません。逆の場合も同様です。
- Naturalスタックの処理も継続できません。

システム変数の転送

*USER以外のシステム変数は、クライアントからサーバー側へ転送できません。

アプリケーションに依存しない変数

RPCサーバーでは、アプリケーションに依存しない変数（AIV）は、暗黙的に割り当て解除されず、複数のRPC要求にわたってアクティブになります。これは、RPCサーバー上の同じ変数に複数のクライアントがアクセスする場合があるためです。このため、割り当てを解除するには、明示的にRELEASE VARIABLESステートメントを使用する必要があります。

あるRPC要求の終わりにAIVの割り当てを解除する場合は、NaturalRPCユーザー出口NATRPC03をその目的で使用できます。

エラー状況でのパラメータ処理

エラー状況でのパラメータ処理は異なります。

- ローカル実行中にエラーが発生すると、パラメータは「call by reference」（参照渡し）経由で渡されるので、それまでに実行された全パラメータ修正は反映されます。
- リモート実行中にエラーが発生すると、全パラメータは変更なしのままです。

サブプログラム内の可変配列

サブプログラムのパラメータデータエリアに可変オカレンス数（1:V 表記）が含まれている場合、このサブプログラムを呼び出すためにスタブを使用できません。 スタブは固定オカレンス数の配列定義だけをサポートするので、コールからコールへオカレンス数を変更することはできません。

スタブサブプログラム（インターフェイスオブジェクト）を使用する必要がある場合、例えば同じプログラムで EntireX RPC サーバーを呼び出す場合には、Natural クライアント側で X-array を使用する必要があります。 X-array を使用すると、スタブサブプログラムを使用するときでも、コールからコールへオカレンス数を変更できます。 この場合、クライアント側の X-array は、サーバー側の（固定の）可変配列に渡されます。 サーバープログラムはコールごとに変化するオカレンス数を受け取る可能性がありますが、オカレンス数を変更できないため、可変配列は固定されます。

X-array

X-array は、リモート CALLNAT ステートメント実行のパラメータリストでサポートされています。 サーバーは、オカレンス数を増減する場合があります。

制限

- 多次元配列の場合、配列のすべての次元が拡張可能である必要があります。
- 下限は拡張可能であってはいけません。つまり、拡張可能な上限のみが許可されます。
- 一定の限度を持つ配列を含む X-group 配列または X-array を含むグループ配列を使用する場合、スタブサブプログラムを使用する必要があります。 スタブサブプログラムを生成する場合、スタブ生成画面でグループ構造を定義する必要があります。

例：

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

グループおよびスタブサブプログラム

グループ配列または X-group 配列がリモート CALLNAT ステートメント実行のパラメータリスト内にあり、スタブサブプログラムが使用される場合、次の制限があります。

制限

- AD=O または AD=A セッションパラメータ設定（属性定義）は、CALLNAT ステートメント内で使用できません。
- クライアント Natural オブジェクトからスタブサブプログラムに渡されるグループ配列および X-Group 配列は連続している必要があります。 したがって、すべての次元に対してアスタリスク（*）表記を使用することによって、常に完全な配列をスタブサブプログラムに渡すことを強くお勧めします。 また、クライアント Natural プログラム、スタブサブプログラム、およびサーバープログラムで同一のデータ定義を使用することも強くお勧めします。

RPC サーバー側のグループ配列

RPC サーバー側のサブプログラムの DEFINE DATA PARAMETER エリア内のグループ配列のストレージレイアウトは、構文に関して必ずしも同一ではありません。 グループ配列内のフィールドを再定義したり、グループ配列を 3GL プログラムに渡したりしないでください。 これらの処理が必要な場合は、グループ配列を DEFINE DATA LOCAL エリア内の同じレイアウトのグループ配列にコピーし、このローカルグループ配列を 3GL プログラムへのコールで使用します。

EntireX RPC サーバー

リモート CALLNAT ステートメント実行によって EntireX RPC サーバーを呼び出す場合、スタブサブプログラムを使用することを強くお勧めします。 EntireX RPC サーバーで呼び出すサブプログラムの IDL（Interface Definition Language）定義にグループ構造が含まれている場合、スタブサブプログラムが必要です。 この場合、［Stub Generation］画面でスタブ生成中に同じグループ構造を定義するか、EntireX IDL ファイルからスタブサブプログラムを生成する必要があります（Windows のみ）。

VSAM の使用

サブタスキング環境で VSAM データセットにアクセスしたり、VSAM データセットをリージョン間で共有したりする場合、必要となる共有オプションを考慮する必要があります。例えば、1 つのアドレススペースで VSAM データセットにレコードが挿入され、別のアドレススペースで同じ VSAM データセットが読み取られる場合、SHAREOPTIONS 2 の代わりにリージョン間 SHAREOPTIONS 4 を設定して、バッファ無効化を強制する必要がある場合があります。そうしないと、新しく挿入されたレコードが検出されない場合があります。

レコードレベル共有（RLS）の使用も考慮する必要があります。

詳細については、関連する IBM の VSAM ドキュメントを参照してください。

Natural ステートメントの動作

いくつかの Natural ステートメントは、次の例のように、Natural RPC のコンテキストで使用したときに異なる動作をすることがあります。

ステートメント	説明
OPEN CONVERSATION CLOSE CONVERSATION	サーバー上で実行した場合は、これらのステートメントはクライアントセッションには影響しません。サーバー自身が（エージェントとしての）他のサーバーに対するクライアントとして動作するときは、これらのステートメントは 2 番目のサーバー上の会話にのみ影響します。
PASSW	パスワード設定はサーバー側でのみアクティブなままとなり、他ユーザーによる後続の実行に対してもアクティブなままとなります。
SET CONTROL SET GLOBALS SET KEY SET TIME SET WINDOW	呼び出し元に設定は返されません。
STACK	すべてのスタックデータは、実行後に解放されます。
STOP TERMINATE	これらのステートメントは、クライアントセッションを停止しません。 Natural RPC サーバーを終了する方法については、「*Natural RPC サーバーの終了*」を参照してください。

サーバーに対する Natural ステートメントの注意事項

現在の Natural バージョンでは、RPC との次のステートメントの組み合わせは理論的には可能ですが、予期しない結果を引き起こすためお勧めしません。

ステートメント	説明
TERMINATE	会話が開いている状態に関係なく、サーバーを終了します。
FETCH RUN STOP	これらのステートメントを使用すると、CALLNAT コンテキストの消失を引き起こします。 FETCH、RUN、または STOP ステートメント上では、サーバーは自身の CALLNAT コンテキストが消失したことを検出し、クライアントへNaturalエラーメッセージを返します。ただし、その時点で、そのステートメントはすでにサーバーにより実行されています。 *例外：*FETCH RETURN には適用されません。
INPUT	Natural スタックからではなくファイルから読み取る場合、入力データ値は予測できません。

5 Natural RPC 環境の設定

Natural RPC 環境を設定するには、すべてのクライアント Natural とサーバー Natural に対して次の手順を実行し、プラットフォーム固有の注意事項および考慮事項を確認する必要があります。

Natural クライアントの設定

別途注釈がない限り、この手順はすべての環境に適用されます。

Natural クライアントを設定するには、次の手順に従います。

- サーバー名の定義
- スタブサブプログラムの生成
- **RPC** クライアント固有の **Natural** パラメータの設定

サーバー名の定義

SYSRPC ユーティリティのサービスディレクトリメンテナンス機能を使用して、リモートに実行される各 CALLNAT ステートメントに対して使用されるサーバーの名前を定義します。

詳細および画面例については、SYSRPC ユーティリティのドキュメントの「*Service Directory Maintenance の呼び出し*」を参照してください。

生成したディレクトリサブプログラム NATCLTGS は、Natural クライアントアプリケーションで有効にする必要があります。クライアントライブラリに NATCLTGS を生成していない場合は、このライブラリまたはいずれかの steplib に NATCLTGS を移動します。

オプションで、次のサーバー選択技法のいずれかを使用できます。

- デフォルトサーバーのアドレス指定

  「***Natural** セッション内のデフォルトサーバーアドレスの指定*」、またはプロファイルパラメータ RPC またはパラメータマクロ NTRPC のキーワードサブパラメータ DFS を参照してください。
- リモートディレクトリサーバーの使用

  「リモートディレクトリサーバーの使用」、またはプロファイルパラメータ RPC またはパラメータマクロ NTRPC のキーワードサブパラメータ RDS を参照してください。

Windows、**UNIX**、および **OpenVMS** 環境に関する注：

Predict サーバーは、SYSRPC ユーティリティではメンテナンスできません。

Predict サーバーへの接続方法については、グローバルコンフィグレーションファイル内のディクショナリサーバー*割り当て*機能またはプロファイルパラメータ USEDIC を参照してください。

クライアントスタブサブプログラムの生成

この手順は、Natural RPC 自動実行を使用しないか、または使用できない場合にのみ適用されます。「*Natural RPC 環境の運用*」の「***Natural RPC 自動実行の操作***」を参照してください。

リモートに実行される各 CALLNAT ステートメントについては、SYSRPC ユーティリティのスタブ生成機能を使用してください。「*スタブサブプログラムの作成*」を参照してください。

生成したスタブは、Natural クライアント環境に対して有効である必要があります。クライアントライブラリ内にスタブサブプログラム（インターフェイスオブジェクト）を生成しなかった場合は、このライブラリまたはいずれかの steplib にスタブサブプログラムを移動します。

RPC クライアント固有の Natural パラメータの設定

リモートプロシージャコールのクライアント固有の処理に関連した Natural プロファイルパラメータ RPC のまたはパラメータマクロ NTRPC のキーワードサブパラメータを設定します。

必須パラメータ：

パラメータ	機能
MAXBUFF	最大バッファサイズ（RPC 自動実行の場合のみ）
RPCSIZE	Natural RPC によって使用されるバッファのサイズ（メインフレームクライアントの場合のみ）

オプションパラメータ：

パラメータ	機能
ACIVERS	EntireX Broker ACI で使用する ACI バージョンの定義
AUTORPC	Natural RPC 自動実行
COMPR	RPC バッファ圧縮の設定 「*Natural RPC 環境の運用*」の「*圧縮の使用*」も参照してください。
CPRPC	コードページ名の定義
DFS	RPC クライアントのデフォルトサーバーアドレスの指定
RDS	リモートディレクトリサーバーの定義
RPCSDIR	サービスディレクトリが存在する Natural ライブラリ名の指定（メインフレームの場合は UNIX サーバーと OpenVMS サーバーのみ）
TIMEOUT	RPC サーバーレスポンスに対する待機時間
TRYALT	代替サーバーアドレスの試行

次の注は、EntireX Broker を使用する場合に該当します。

注意:

1. 1. DFS パラメータで指定する名前はアクティブな EntireX Broker を識別する必要があり、EntireX Broker 属性ファイル内のサーバー定義と一致している必要があります。「*EntireX Broker 環境の設定*」を参照してください。
2. 2. TIMEOUT で指定する待機時間は、EntireX Broker ACI の WAIT フィールドの設定に使用されます。 TIMEOUT をゼロに設定した場合、WAIT=YES が設定され、クライアントは CLIENT-NONACT 時間待機します。待機時間が経過すると、リモートプロシージャコールは該当するエラーメッセージで終了します。 TIMEOUT を使用すると、EntireX Broker スタブによって提供されるトランスポートタイムアウトメカニズムを利用できます。

Natural サーバーの設定

Natural サーバーは、Natural サブプログラム（サービス）の実行が可能な Natural タスク（サーバータスク）です。この Natural タスクは、基本的に非同期またはバックグラウンドタスク（デタッチプロセス）です。 EntireX Broker およびクライアントは、*nodename* および *servername* を使用してそれを識別します。

Natural サーバーを設定するには、次の手順に従います。

- RPC サーバー固有の Natural パラメータの設定
- サーバーセッションでのコマンドモード使用の確認
- 一意の Adabas ETID の使用の確認
- Natural サーバーの起動

RPC サーバー固有の Natural パラメータの設定

Natural サーバーに対するリモートプロシージャコールの全体的およびサーバー固有の処理に関連するプラットフォーム依存の Natural パラメータを設定します。

メインフレームサーバーの場合：

1. RPC 固有の Natural パラメータモジュールを作成します。
2. 必要に応じて、プロファイルパラメータ RPC またはパラメータマクロ NTRPC（次の表を参照）のキーワードサブパラメータを設定します。

Windows、**UNIX**、または **OpenVMS** サーバーの場合：

1. RPC 固有の Natural パラメータファイルを作成します。
2. 必要に応じて、Natural プロファイルパラメータ（次の表を参照）を設定します。

必須パラメータ：

パラメータ	機能
MAXBUFF	最大バッファサイズ
RPCSIZE	Natural RPC によって使用されるバッファのサイズ（メインフレームサーバーの場合のみ）
SERVER	RPC サーバーセッションとしての Natural セッションの開始
SRVNAME	RPC サーバーの名前。次の *EntireX Broker に関する注意事項*を参照してください。
SRVNODE	ノードの名前。次の *EntireX Broker に関する注意事項*を参照してください。

オプションパラメータ：

パラメータ	機能
ACIVERS	EntireX Broker ACI で使用する ACI バージョンの定義
CPRPC	コードページ名の定義
LOGONRQ	RPC サーバー要求のログオン指定
NTASKS	サーバーレプリカ数の最小および最大（メインフレームサーバーの場合のみ）
SRVCMIT	Natural RPC サーバーが RPC 会話または非会話型 RPC 要求を自動的にコミットする時刻
SRVTERM	サーバー終了イベント
SRVUSER	RPC サーバーレジストリ用のユーザー ID
SRVWAIT	クライアント要求に対する RPC サーバーの待機時間
TRACE	トレースするコンポーネントの定義
TRANSP	サーバートランスポートプロトコル（現在は不要）

次の注は、EntireX Broker を使用する場合に該当します。

注意:

1. SRVNODE で指定する名前はアクティブな EntireX Broker を識別する必要があり、SRVNAME で指定する名前は EntireX Broker 属性ファイル内のサーバー定義と一致している必要があります。「*EntireX Broker 環境の設定*」を参照してください。
2. SRVWAIT パラメータで指定する待機時間は、EntireX Broker ACI の WAIT フィールドの設定に使用されます。 SRVWAIT を指定しないかゼロに設定した場合、WAIT=YES が設定され、サーバーは SERVER-NONACT 時間待機します。 待機時間が経過すると、該当するメッセージが RPC サーバートレースファイルに書き込まれ、RPC サーバーは次のクライアント要求を待機し続けます。 SRVWAIT パラメータを使用すると、EntireX Broker スタブによって提供されるトランスポートタイムアウトメカニズムを利用できます。

サーバーセッションでのコマンドモード使用の確認

手順 5.1. Natural サーバーセッションがコマンドモードに入ることを確認するには

- Naturalプロファイルパラメータ MENU=OFF を設定して、Naturalメニューモードを無効にします（メインフレームサーバーのみに適用されます）。

次のことは行わないでください。

- 終了しない Natural スタックにプログラムを置く。
- 終了しない STARTUP プログラムを使用する。
- Natural Security で、サーバーライブラリに対して NEXT モードを禁止にする。

一意の Adabas ETID の使用の確認

Natural サーバーセッションが使用する Adabas ETID が特定の Adabas ニュークリアス内で一意であることを確認してください。

Natural サーバーの起動

Natural サーバーを起動するには、「*Natural RPC サーバーの起動*」セクションの説明に従ってください。

その後、このサーバーはクライアントからのリモート CALLNAT 要求を待機します。

z/OS または **z/VSE** 上のバッチモードの **Natural** に関する注：

プロファイルパラメータ RPC またはパラメータマクロ NTRPC のキーワードサブパラメータ NTASKS を使用するサーバーについては、「*レプリカ付きメインフレーム Natural RPC サーバーの考慮事項*」を参照してください。

EntireX Broker アクセスの設定

EntireX Broker インターフェイスを設定するには、次の手順に従います。

- EntireX Broker スタブへのアクセスの提供
- ACI バージョンの設定

- トランスポートメソッドとしての TCP/IP の使用

EntireX Broker スタブへのアクセスの提供

Natural 環境へ EntireX Broker スタブをアクセス可能にします。この手順は、使用するプラットフォームによって異なります。

- メインフレームでの EntireX Broker スタブへのアクセスの提供
- UNIX での EntireX Broker スタブへのアクセスの提供
- Windows での EntireX Broker スタブへのアクセスの提供

メインフレームでの EntireX Broker スタブへのアクセスの提供

Natural に EntireX Broker スタブ NATETB23 をリンクします。または、ランタイム時にダイナミックに NATETB23 をロードするためにプロファイルパラメータ RCA=BROKER を指定します。

次の場合には NATETB23 は使用できないため、別の Broker スタブを使用する必要があります。

- BS2000/OSD で TCP/IP プロトコルを使用する場合は、代わりに BKIMBTIA を使用する必要があります。
- z/OS バッチモードで偽装を使用する場合は、代わりに BKIMBTSO を使用する必要があります。
- CICS で偽装を使用する場合は、代わりに CICSETB を使用する必要があります。

注意: CICSETB を Natural CICS インターフェイスニュークリアスにリンクする必要があります。

BKIMBTIA または BKIMBTSO をランタイム時にダイナミックにロードするには、RCA=BROKER RCALIAS=(BROKER,*stubname*) を指定します。

現在、CICSETB をランタイム時にダイナミックにロードすることはできません。

詳細については、EntireX Communicator のドキュメントを参照してください。

UNIX での EntireX Broker スタブへのアクセスの提供

Natural 固有のブローカースタブ natetb.so/natetb.sl を、ディレクトリ $EXXDIR/$EXX/VERS/lib から、ローカルコンフィグレーションファイル *NATURAL.INI* の NATEXTLIB で指定されたディレクトリにコピーします。

Windows での EntireX Broker スタブへのアクセスの提供

EntireX Broker スタブは、EntireX のインストール中に自動的に使用可能になります。

ACI バージョンの設定

プロファイルパラメータ RPC またはパラメータマクロ NTRPC のキーワードサブパラメータ ACIVERS を、要件に合わせて設定します。

注意: Natural パラメータモジュール（メインフレーム）またはパラメータファイル（Windows、UNIX、または OpenVMS）内に設定する ACIVERS 値は、EntireX Broker および EntireX Broker スタブもこのバージョンをサポートする場合のみ作用可能です。

次の表に、Natural RPC に関連する機能に関連付けられた ACIVERS の値のみを示します。

設定	機能
ACIVERS=2	デフォルト。EntireX Broker の LOGON および LOGOFF 機能をサポートします。 サーバーは、EntireX Broker への LOGON を実行してから REGISTER を実行し、DEREGISTER の後で LOGOFF を実行します。 これは、どのようなセキュリティチェックも伴わない、純粋な EntireX Broker 管理機能です。EntireX Broker のドキュメントの EntireX Broker の LOGON 機能を参照してください。
ACIVERS=3	EntireX Broker の非数値会話 ID をサポートし、データ量は 30 KB を超えます。 ACIVERS を 3 以上の値に設定すると、EntireX Broker は非数値会話 ID も割り当てます。 Natural クライアントが OPEN CONVERSATION ステートメントを発行し、クライアントの ACIVERS が 3 以上の場合、EntireX Broker は自動的に非数値会話 ID を割り当てることができます。 関連するサーバーが非数値会話 ID を受け付けるかどうかはチェックしませんが、要求側（この場合は Natural クライアント）の ACIVERS のみが明白になります。 したがって、Natural クライアントおよびサーバーの両方が、対応する ACI バージョンをサポートすることを確認してください。 また、EntireX Broker ACI バージョン 3 以上では、トランスポートメソッド TCP/IP が使用される場合、1 つの要求でクライアントとサーバー間で交換できるデータ量が 30 KB を超えることが可能です。 **注意:** 1. EntireX Broker ACI バージョン 1 または 2 では、データ量は 30 KB に制限されます。 2. トランスポートメソッド NET では、30 KB を超えるデータ量をサポートするために、EntireX Broker 属性 EXTENDED-ACB-SUPPORT を YES に設定する必要があります。
ACIVERS=4	コードページおよび（サーバーのみ）Natural Security をサポートします。

設定	機能
	EntireX Broker ACI V4 以上では、Natural RPC はコードページをサポートします。 このために、コードページの名前を、クライアントおよびサーバーのプロファイルパラメータ RPC またはパラメータマクロ NTRPC のキーワードサブパラメータ CPRPC に指定することができます。 コードページの評価は EntireX Broker により行われます。 つまり、EntireX Broker は送信した RPC データをクライアントおよびサーバーのコードページから該当するターゲットコードページに変換します。 プロファイルパラメータ RPC またはパラメータマクロ NTRPC のキーワードサブパラメータ CPRPC は、クライアント／サーバーで設定できます。現在のプロセスに対して適用します。これは、クライアントコードページはサーバーコードページと同一である必要のないことを意味します。 サーバーは認可したユーザー ID を使用することにより、EntireX Broker へのログオンが可能となります。 プロファイルパラメータ RPC またはパラメータマクロ NTRPC のキーワードサブパラメータ SRVUSER が *NSC に設定されており、サーバーが Natural Security 環境で実行されている場合、Natural RPC は、Natural Security 内に定義された現在の Natural ユーザーID（システム変数 *USER の内容）およびパスワードを EntireX Broker へ自動的に渡します。そこで、EntireX Broker セキュリティデータとの一致が確認されます。
ACIVERS=6	EntireX 暗号化（ENCRYPTION-LEVEL）をサポートします。 EntireX Broker ACI バージョン 6 以上では、アプリケーションプログラミングインターフェイス USR4009N を使用して ACI フィールド ENCRYPTION-LEVEL を設定できます。
ACIVERS=7	EntireX 圧縮（COMPRESSLEVEL）をサポートします。 EntireX Broker ACI バージョン 7 以上では、アプリケーションプログラミングインターフェイス USR4009N を使用して ACI フィールド COMPRESSLEVEL を設定できます。
ACIVERS=8	スタブ出口のない EntireX Security をサポートします（メインフレームのみ）。 EntireX Broker ACI バージョン 8 以上では、Natural RPC サーバーは、KERNELVERS コールを発行して ACI フィールド KERNELSECURITY の正しい値を取得します。 この場合、EntireX Security 出口を EntireX Broker スタブにリンクする必要はなくなりました。
ACIVERS=9	クライアントおよびサーバー環境の EntireX アプリケーション識別および（サーバーのみ）Integrated Authentication Framework（IAF）をサポートします。 EntireX Broker ACI バージョン 9 以上では、EntireX Broker スタブは、クライアントおよびサーバーの環境情報（ジョブ名など）を EntireX Broker に送信します。 EntireX Broker ACI バージョン 9 以上では、Natural RPC サーバーは、オプションでクライアント認証に IAF を使用できます。 詳細については、最新の EntireX ドキュメントを参照してください。

トランスポートメソッドとしての TCP/IP の使用

トランスポートメソッドとして TCP/IP が使用されており、ホスト名を使用してサーバーノードをアドレスする場合、次の代替手段があります。

- TCP/IP インストールの Hosts および Services ディレクトリ内にサーバーノードを定義します。
- ドメイン名の解決に Domain Name System（DNS）を使用します。

EntireX Broker 環境の設定

EntireX Broker 属性ファイル内に、以下を追加します。

1. 各 Natural RPC サーバーについて、サービス定義を次のように指定する必要があります。

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

2. 会話サービスを使用する場合は、`CONVERSION=`*`userexit`* を設定します。 この場合は、この設定に合わせてプロファイルパラメータ `RPC` またはパラメータマクロ `NTRPC` のキーワードサブパラメータ `CPRPC` を設定する必要があります。

Reliable RPC を使用する場合は、Reliable RPC がサポートされている各 Natural RPC サーバーに追加設定が必要です。

- EntireX Broker 属性 `MAX-UOWS` を正の値に設定する必要があります。
- クライアントが Reliable RPC メッセージを、EntireX Broker にとって既知の、まだ起動されていない RPC サーバーに送信できるようにする場合は、EntireX Broker 属性 `DEFERRED` を `YES` に設定する必要があります。
- システム障害後に Reliable RPC メッセージを復旧できるようにする場合は、EntireX Broker 属性 `STORE` を `BROKER` に設定する必要があります。 また、EntireX Broker のパーシスタントストアを有効にする必要もあります。
- Reliable RPC メッセージそのものの寿命（EntireX Broker 属性 `UWTIME`）およびそのステータスの寿命（EntireX Broker 属性 `UWSTAT-LIFETIME`）を、要件に合わせて調整する必要があります。

注意: EntireX 属性ファイルで `AUTOLOGON=NO` または `SECURITY=YES` が設定されている場合、`ACIVERS=2` 以上を設定する必要があります。

6 Natural RPC サーバーの起動

このセクションでは、さまざまなプラットフォームで Natural RPC サーバーを起動する方法について説明します。

Natural RPC サーバーを起動する前の予備知識

任意の種類の Natural セッションを Natural RPC サーバーとして使用できますが、通常、Natural サーバーは非同期タスクまたはバックグラウンドタスクとして開始された Natural セッションとなります。

メインフレーム：

サーバーを起動するには、次のオプションがあります。

- RPC 固有の Natural パラメータモジュールを作成します。

  関連するパラメータのリストについては、「*Natural RPC 環境の設定*」の「*RPC サーバー固有の Natural パラメータの設定*」を参照してください。

  このパラメータモジュールは、`PARM=serverparm` でダイナミックに指定されます。*serverparm* は、Natural にリンクされたパラメータモジュールの名前です。
- または、プロファイルパラメータをダイナミックに指定することもできます。

  RPC 固有の Natural プロファイルパラメータは、`SYSPARM` ユーティリティで作成したプロファイル内に指定できます。 その後 Natural は、下記で起動されます。

  ```
  PROFILE=serverprofile
  ```

  serverprofile は、プロファイルの名前です。

Windows、**UNIX**、または **OpenVMS**：

サーバーを起動するには、次のオプションがあります。

- RPC 固有の Natural パラメータファイルを作成します。

  関連するパラメータのリストについては、「*Natural RPC 環境の設定*」の「*RPC サーバー固有の Natural パラメータの設定*」を参照してください。

  このパラメータファイルは、`PARM=serverparm` でダイナミックに指定されます。*serverparm* は、パラメータファイルの名前です。
- 代わりに、プロファイルパラメータをダイナミックに指定することもできます。

Natural サーバーの起動方法は環境によって異なります。次の該当する説明を参照してください。

メインフレームオンライン環境での Natural RPC サーバーの起動（すべての TP モニタ）

メインフレームオンライン環境で Natural サーバーを起動するには、TP モニタ環境で次のコマンドを入力します。

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

<natural> は、Natural を起動するときの名前です（トランザクションコード、トランザクション ID、環境従属ニュークリアス名）。

メインフレームオンライン環境での Natural RPC サーバーの起動（Com-plete および CICS のみ）

Com-plete 環境または CICS 環境では、Natural RPC サーバーを起動するための次のオプションがあります。

- すべての TP モニタについて前述したのと同じコマンドを使用できます。
- 非同期モードで Natural サーバーを起動するために、SYSRPC ライブラリ内の Natural プログラム STARTSRV を使用できます。
- TP モニタの起動時に、非同期モードで Natural RPC サーバーを起動できます。

STARTSRV による非同期モードでの Natural RPC サーバーの起動

STARTSRV は、非同期 Natural セッションを起動する RPCSSRV のためのサンプルフロントエンドです。

デフォルトで、非同期 Natural は、現在のセッションと同じライブラリの同じ Natural 名で起動されます。

Natural Security（NSC）を使用している場合、現在の Natural セッションのユーザー ID も伝播されます。 入力を要件に適合させることができます。

一部の Natural プロファイルパラメータは、RPCSSRV によって暗黙的に追加されます。 このことは特に、プロファイルパラメータ RPC またはパラメータマクロ NTRPC のキーワードサブパラメータ RPCSIZE に適用されます。 RPCSIZE のデフォルトは MAXBUFF+4 です。MAXBUFF は、マップフィールド **Receiving buffer** に入力された値です。 マップフィールド **Session parameter** に RPC=(RPCSIZE=*n*) を入力し、RPCSIZE のデフォルト値を上書きできます。

Natural セッションの起動時に Natural プログラムを実行する場合は、マップフィールド **User Stack** を使用できます。 **User Stack** の内容は Natural STACK に置かれ、Natural サーバーがアクティブになる前に実行されます。

関係する Natural プロファイルパラメータを表示するには、*SHOW* を **Transaction ID** フィールドに入力します。 STARTSRV によって、非同期 Natural セッションを開始するために RPCSSRV によって使用されるダイナミックプロファイルパラメータがすべて表示されます。

後述の「*CICS に関する注意事項*」も参照してください。

TP モニタの起動時の非同期モードでの **Natural RPC** サーバーセッションの開始

TP モニタの起動時に非同期モードで Natural RPC サーバーセッションを開始するには、次の手順に従います。

- **Com-plete** の場合：

  起動オプション（sysparms）STARTUPPGM を使用し、必要なすべての RPC 固有の Natural プロファイルパラメータを指定して、Natural セッションを開始します。

- **CICS** の場合：

  PLTPI を使用して EXEC CICS START を使用するプログラムを起動し、必要なすべての RPC 固有の Natural プロファイルパラメータを指定して、Natural セッションを開始します。 Natural CICS ソースライブラリに用意されているサンプル PLTPI プログラム XNCIFRNP を適合させることができます。

どちらの場合にも、Natural セッションは TP モニタによって割り当てられたユーザー ID で開始される点に注意してください。

- Com-plete では、このユーザー ID は Com-plete が起動されたユーザー ID です。
- CICS では、このユーザー ID は CICS のデフォルトのユーザー ID です（デフォルトは CICSUSER）。

どちらの場合にも、このユーザー ID は Natural システム変数 *INIT-USER および *USER に割り当てられます。 したがって、Natural セッションが Natural Security 環境で実行されている場合、Natural LOGON コマンドを Natural STACK に置く必要がある場合があります。

CICS に関する注意事項：

非同期 Natural RPC サーバーを起動する場合は、Natural プロファイルパラメータ TTYPE および SENDER を次のように設定することをお勧めします。

```
TTYPE=ASYL,SENDER=CSSL
```

これにより、プライマリ出力先への各出力は、3270 モードではなく行モードで書き込まれます。CSSL の代わりに、他の任意の CICS 出力先を使用できます。

TTYPE=ASYL を使用する場合、NATBTCH が Natural ニュークリアスにリンクされている必要があります。

メインフレーム環境でのバッチサーバーの起動

バッチサーバーは、標準 Natural バッチセッションです。「*Natural RPC 環境の設定*」の「*RPC サーバー固有の Natural パラメータの設定*」で説明されている RPC パラメータで起動されます。

以下では次のトピックについて説明します。

- z/OS でのバッチサーバーの起動
- z/VSE でのバッチサーバーの起動
- BS2000/OSD でのバッチサーバーの起動

注意: トレース機能を使用するサンプル JCL については、「*Natural RPC 環境の運用*」の「*サーバートレース機能の使用*」を参照してください。

z/OS でのバッチサーバーの起動

z/OS のサンプル JCL

```
//NATRPC   JOB  CLASS=K,MSGCLASS=X
//         EXEC PGM=NATOS,REGION=8M
//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
//         DD   DISP=SHR,DSN=SAG.SSX.LOAD          <== 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
/*
```

注意:

1. Adabas リンクルーチン ADAUSER または Natural プロファイルパラメータ ADANAME が使用されている場合のみ適用されます。
2. DB2 ユーザーのみに適用されます。
3. Integrated Authentication Framework（IAF）が使用されている場合のみ適用されます。
4. SSL が使用されている場合のみ適用されます。

開始タスクのサンプル **JCL**

開始タスクのサンプル JCL は、Natural for Mainframes の『*インストール*』ドキュメントの「*z/OS 環境での Natural のインストール*」セクションの「*Natural RPC サーバー用サンプル JCL の作成*」で紹介されています。

レプリカでのバッチサーバーの実行

また、プロファイルパラメータ RPC またはパラメータマクロ NTRPC のキーワードサブパラメータ NTASKS を 1 より大きい値に設定することによって、レプリカでバッチサーバーを実行することもできます。

レプリカは、追加のサーバータスクとして、Natural メインタスクにアタッチされます。これにより、同じリージョンで複数の同一のサーバーを起動できるようになります。

z/VSE でのバッチサーバーの起動

z/VSE のサンプル **JCL**

```
// 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)
/*
```

レプリカでのバッチサーバーの実行

また、プロファイルパラメータ RPC またはパラメータマクロ NTRPC のキーワードサブパラメータ NTASKS を 1 より大きい値に設定することによって、レプリカでバッチサーバーを実行することもできます。

レプリカは、追加のサーバータスクとして、Natural メインタスクにアタッチされます。これにより、同じリージョンで複数の同一のサーバーを起動できるようになります。

BS2000/OSD でのバッチサーバーの起動

BS2000/OSD のサンプル **JCL**

```
/.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
```

Windows 環境での Natural RPC サーバーの起動

Windows 環境で Natural RPC サーバーを起動するには、次の手順に従います。

1. Natural 用ショートカットを作成します。
2. ショートカットのプロパティを入力します。
3. ***RPC サーバーパラメータ***設定で Natural パラメータファイルを作成します（「*代替パラメータファイルによる Natural の呼び出し*」を参照）。
4. ［**Target**］テキストボックスで、Natural パスを編集して追加します。

```
parm=serverparm batch
```

serverparm は、パラメータファイルの名前です。

または

```
server=on,srvname=servername,srvnode=nodename,maxbuff=n batch
```

UNIX 環境での Natural RPC サーバーの起動

UNIX 環境で Natural RPC サーバーを起動するには、次のコマンドを入力します。

```
natural parm=serverparm >/dev/null </dev/null &
```

serverparm は、パラメータファイルの名前です。

または

```
natural server=on,srvname=servername,srvnode=nodename,maxbuff=n >/dev/null </dev/null
&
```

OpenVMS 環境での Natural RPC サーバーの起動

OpenVMS 環境で Natural RPC サーバーを起動するには、DCL コマンドプロシージャ *myserver*.com に、次のコマンドを入力します。

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

次に、*myserver*.com をバッチキューに送信します。

```
$ SUBMIT myserver.com
```

レプリカ付きメインフレーム Natural RPC サーバーの考慮事項

このセクションは、z/OS および z/VSE 環境のメインフレーム Natural サーバーに適用されます。

- 1 より大きい NTASKS が指定された Natural RPC バッチサーバー
- レプリカでのバッチサーバーの実行

1 より大きい NTASKS が指定された Natural RPC バッチサーバー

メインタスクおよび全レプリカは、同じ z/OS リージョンまたは z/VSE パーティション内で実行されます。

1. Adabas バッチリンクルーチン ADALNK のリエントラントバージョン ADALNKR を使用します。

   ADAUSER を使用する場合、ADAUSER は非リエントラントであるため、ADAUSER をフロントエンドとリンクしないようにしてください（項目 5 を参照）。 代わりに、Natural プロファイルパラメータ ADANAME を使用し、ADANAME=ADAUSER を設定します。 これにより、Natural はランタイム時に ADAUSER をダイナミックにロードします。

   z/VSE に関する注：ADAUSER を使用する場合、ADALNKR の名前を ADALNK に変更する必要があります。

2. NATPARM モジュール内：

   - プロファイルパラメータ RPC またはパラメータマクロ NTRPC のキーワードサブパラメータ NTASKS=*n* を設定します。*n* は、メインタスクを含む起動される並列サーバーの数（< 100）です。

     z/VSE に関する注：サブタスクの数はオペレーティングシステムによって制限されます。システム管理者に確認してください。

   - Natural プロファイルパラメータ ETID を使用して、Adabas ユーザー ID を空白文字として指定します。 これは、サブタスクが起動されるときに NAT3048 エラー（Adabas ニュークリアス内で ETID が一意でない）を回避するために必要です。

3. ダイナミック Natural プロファイルパラメータ使用時：

   ダイナミック Natural プロファイルパラメータを Natural に渡すために、ダイナミックパラメータデータセット CMPRMIN を使用します。 PARM カードやプライマリコマンド入力データセット CMSYNIN は*使わないでください*。

4. ローカルバッファプール使用時（z/OS のみ）：

   共有ローカルバッファプールを指定しない限り、各サブタスクはそれぞれのローカルバッファプールを割り当てます。Natural の『*オペレーション*』ドキュメントの「*Natural z/OS 生成パラメータ*」セクションのパラメータ LBPNAME に関する記載を参照してください。

5. Natural フロントエンドリンクジョブ内（z/OS のみ）：

   リンケージエディタの RENT オプションを使用してフロントエンドリエントラントをリンクします。

   フロントエンドが RENT オプションでリンクされなかった場合、メインタスクは EntireX Broker との通信だけが開始されます。 すべてのサブタスクは、メインタスクが終了するまで、z/OS で WAIT ステータスに設定されます。 RPC サーバーを後で終了すると、アドレススペースはハングし、キャンセルする必要があります。

6. Naturalニュークリアスに追加リンクされている他のモジュールは、リエントラントであることを明確にしてください。 ダイナミックにロードされる任意のプログラムもリエントラントである必要があります。

   z/OS関する注：モジュールをリエントラントにできない場合、モジュールを再利用不可としてリンクしてください（リンクオプション RENT または REUS を指定しないでください）。 これは、サブタスクが自分のコピーを取得するようにするためです。

レプリカでのバッチサーバーの実行

サンプル JCL については、「*サーバートレース機能の使用*」を参照してください。

RPC サーバーフロントエンドを使用した Natural RPC サーバーの起動（z/OS バッチモードのみ）

z/OS バッチモードでは、Natural RPC サーバーが RPC サーバーフロントエンドを使用して起動される場合があります。 この方法は、偽装で必要であり、それ以外の場合はオプションです。

偽装なしで RPC サーバーフロントエンドを使用する場合、プロファイルパラメータ RPC またはパラメータマクロ NTRPC のキーワードサブパラメータ NTASKS を 1 より大きい値に設定することをお勧めします。 それ以外の場合は、メリットがありません。「*レプリカ付きメインフレーム Natural RPC サーバーの考慮事項*」は、RPC サーバーフロントエンドを使用する場合にも適用されます。

RPC サーバーフロントエンドでは、Natural サーバー機能が使用されます。Natural の『オペレーション』ドキュメントの「*z/OS 環境のサーバーとしてのNatural*」を参照してください。その機能は、次のとおりです。

- Natural RPC サーバーフロントエンドは、プロファイルパラメータ RPC またはパラメータマクロ NTRPC のキーワードサブパラメータ NTASKS で指定された数の Natural RPC サーバーセッションを開始します。
- すべての Natural RPC サーバーセッションは、同じ Natural プロファイルパラメータの設定で実行されます。

  Natural プロファイルパラメータの設定は、Natural パラメータモジュール NATPARM およびダイナミックパラメータデータセット CMPRMIN（使用可能な場合）から取得されます。 JCL ステートメント EXEC のパラメータ PARM= は、Natural プロファイルパラメータを指定するために使用されません。
- 現在、すべての Natural RPC サーバーセッションがクライアントで使用されており（クライアント要求を実行中か、会話内で次の要求を待機中）、プロファイルパラメータ RPC またはパラメータマクロ NTRPC のキーワードサブパラメータ NTASKS が 1 より大きな値に設定されている場合、補助 Natural RPC サーバーセッションが開始されます。 これらの Natural RPC サーバーセッションは、クライアントで使用されていない Natural RPC サーバーセッション

が少なくとも他に1つある場合、最初のEntireX Brokerタイムアウトで自動的に終了します。他のすべてのNatural RPCサーバーセッションがクライアントで使用されている場合は、次のEntireX Brokerタイムアウトまで、補助RPCサーバーセッションが稼働を続けます。これにより、新しいクライアント要求を処理するNatural RPCサーバーが常に存在するようになります。

- TPモニタシステムで実行されるNaturalセッションに類似したスレッド環境で、Natural RPCサーバーセッションが実行されます。
- すべての無効なNatural RPCサーバーセッション（クライアント要求を機するセッション）は、Naturalロールサーバーに発行されます。
- 偽装を使用する場合：
  非会話型 `CALLNAT` の終了時および会話の終了時に、すべてのデータベースセッションおよびすべてのワークファイルはクローズされます。これにより、次のクライアント要求ではその固有のユーザーIDでデータベースおよびワークファイルがオープンされます。
- 偽装を使用しない場合：
  最初のEntireX Brokerタイムアウトの後、すべてのデータベースセッションおよびすべてのワークファイルはクローズされます。これにより、待機中にリソースがブロックされないようになります。

起動パラメータ：

必要な起動パラメータは、JCL内の `EXEC` ステートメントの `PARM=` パラメータで渡されます。これらのパラメータは次のとおりです。

- Natural z/OSバッチニュークリアスの名前。
- ストレージスレッドのサイズ。
- 割り当てられるストレージスレッドの数。

  同時に実行できるNatural RPCサーバーセッションの数は、ストレージスレッドの数によって決まります。ストレージスレッドの数は、プロファイルパラメータ `RPC` またはパラメータマクロ `NTRPC` のキーワードサブパラメータ `NTASKS=`*n* の値以上である必要があります。
- オプションのキーワード `UCTRAN`。

  `UCTRAN` は、RPCサーバーフロントエンドのすべてのメッセージが大文字に変換されることを表します。

これらのパラメータはコンマで区切る必要があります。先頭または末尾に空白を入力しないでください。

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

次の「*サンプル JCL*」も参照してください。

実行上の注意：

- Natural ロールサーバーが必要です。

  （プロファイルパラメータ SUBSID で定義された）使用する *subsystem-id* に対して Natural ロールサーバーを起動してから、Natural RPC サーバーフロントエンドを起動する必要があります。
- ジョブ名（開始されるタスクの名前）は、ロールファイルに発行されるすべての Natural セッションのハイレベル修飾子として使用されます。

  同じジョブ名（同じ開始タスク）および同じ *subsystem-id* の複数の Natural RPC サーバーフロントエンドを起動しないことを強くお勧めします。
- Natural z/OS バッチニュークリアスはダイナミックにロードされます。

  z/OS バッチニュークリアスを含むロードライブラリが、STEPLIB 連結で使用可能である必要があります。
- EntireX Broker スタブ NATETB23 は使用しないでください。

  EntireX Broker は、Natural コンテキストの*outside*でアクセスされます。したがって、EntireX Broker スタブ BKIMBTSO を使用する必要があります。

偽装を使用する場合のみ：
偽装機能を使用する場合、RPC サーバーフロントエンドは、許可プログラム機能（APF）ライブラリから実行する必要があります。RPC サーバーフロントエンドは、APF 認可 LINKLIST ライブラリから実行することをお勧めします。このことによって、STEPLIB 連結 APF 認可全体を指定する必要がなくなります。

Natural Security の場合のみ：
Natural RPC サーバーフロントエンドをプロファイルパラメータ AUTO=OFF で開始する場合、Natural スタックで、ライブラリ ID、ユーザー ID、およびパスワードを指定して Natural LOGON コマンドを実行する必要があります。つまり、STACK=(LOGON *library-id;user-id;password*) のように指定します。

サンプル **JCL**：

```
//NATRPC   JOB  CLASS=K,MSGCLASS=X
//         EXEC PGM=RPC-FRONT,REGION=8M
//         PARM='Natural-z/OS-interface-module,1000,5'
//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
//         DD   DISP=SHR,DSN=SAG.SSX.LOAD         <== Note 3
//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,BKIMBTSO)
STACK=(LOGON serverlibrary,userID,password)
/*
//CEEOPTS  DD *                                  <== Note 4
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
/*
```

注意：

1. Adabas リンクルーチン ADAUSER または Natural プロファイルパラメータ ADANAME が使用されている場合のみ適用されます。
2. DB2 ユーザーのみに適用されます。
3. Integrated Authentication Framework（IAF）が使用されている場合のみ適用されます。
4. SSL が使用されている場合のみ適用されます。

RPC サーバーフロントエンドを使用した Natural RPC サーバーの起動（CICS のみ）

CICS では、Natural RPC サーバーが RPC サーバーフロントエンドを使用して起動される場合があります。 この方法は、偽装が使用される場合に必要であり、それ以外の場合はオプションです。

偽装なしで RPC サーバーフロントエンドを使用する場合、プロファイルパラメータ RPC またはパラメータマクロ NTRPC のキーワードサブパラメータ NTASKS を 1 より大きい値に設定することをお勧めします。 それ以外の場合は、メリットがありません。

RPC サーバーフロントエンドでは、Natural サーバー機能が使用されます。 その機能は、次のとおりです。

- Natural RPC サーバーフロントエンドは、Natural RPC サーバーフロントエンドの「*CICS のカスタマイズ*」手順で定義されたトランザクション ID によって起動されます。

  「*Natural CICS インターフェイスのインストール*」の「*CICS のカスタマイズ*」を参照してください。

  トランザクション ID を端末で入力するか、またはライブラリ SYSRPC の Natural プログラム STARTSFE を使用して、Natural RPC サーバーフロントエンドを非同期モードで起動できます。

  Natural RPC サーバーフロントエンドには、Natural CICS インターフェイスニュークリアスの名前が起動パラメータとして必要です。 この起動パラメータは、トランザクション ID とともに渡されます。
- Natural RPC サーバーフロントエンドは、プロファイルパラメータ RPC またはパラメータマクロ NTRPC のキーワードサブパラメータ NTASKS で指定された数の Natural RPC サーバーセッションを開始します。
- すべての Natural RPC サーバーセッションは、同じ Natural プロファイルパラメータの設定で実行されます。

  Natural プロファイルパラメータの設定は、Natural パラメータモジュール NATPARM およびダイナミックパラメータデータセット CMPRMIN（使用可能な場合）から取得され、トランザクション ID とともに渡されるダイナミックに指定されたプロファイルパラメータによって上書きできます。 ダイナミックに指定されたプロファイルパラメータは、起動パラメータの後に続いている必要があります。
- 現在、すべての Natural RPC サーバーセッションがクライアントで使用されており（クライアント要求を実行中か、会話内で次の要求を待機中）、プロファイルパラメータ RPC またはパラメータマクロ NTRPC のキーワードサブパラメータ NTASKS が 1 より大きな値に設定されている場合、補助 Natural RPC サーバーセッションが開始されます。 これらの Natural RPC サーバーセッションは、クライアントで使用されていない Natural RPC サーバーセッションが少なくとも他に 1 つある場合、最初の EntireX Broker タイムアウトで自動的に終了します。

他のすべての Natural RPC サーバーセッションがクライアントで使用されている場合は、次の EntireX Broker タイムアウトまで、補助 RPC サーバーセッションが稼働を続けます。 これにより、新しいクライアント要求を処理する Natural RPC サーバーが常に存在することが確実になります。

- TP モニタシステムで実行される Natural セッションに類似したスレッド環境で、Natural RPC サーバーセッションが実行されます。
- すべての無効な Natural RPC サーバーセッション（クライアント要求を機するセッション）は、Natural ロールサーバーに発行されます。
- 偽装を使用する場合：

  非会話型 CALLNAT の起動時および会話の起動時に、EXEC CICS START TRANSID() コマンドの USERID() オプションを使用することによって、クライアントのユーザー ID で新しい CICS ワーカータスクが開始されます。 クライアント要求は、Natural によってこのワーカータスクで実行されます。 クライアント要求が実行されている間、Natural RPC サーバーセッションはワーカータスクの終了を待機します。

  非会話型 CALLNAT の終了時および会話の終了時に、ワーカータスクは終了され、すべてのデータベースはクローズされて、すべての CICS リソースは解放されます。 このことによって、次のクライアント要求では、その固有のユーザー ID によって、データベースがオープンされ、CICS リソースがアクセスされます。
- 偽装を使用しない場合：

  クライアント要求は、Natural RPC サーバーセッション自身によって実行されます。

  最初の EntireX Broker タイムアウトの後、すべてのデータベースはクローズされ、すべての CICS リソースは解放されます。 これにより、待機中にリソースがブロックされないようになります。

起動パラメータ：

必要な起動パラメータは、トランザクション ID とともに渡されます。 これらのパラメータは次のとおりです。

- Natural CICS インターフェイスニュークリアス *<ncistart>* の名前。
- オプションのキーワード UCTRAN。

  UCTRAN は、RPC サーバーフロントエンドのすべてのメッセージが大文字に変換されることを表します。
- オプションの Natural プロファイルパラメータ文字列。

端末での起動例：

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

<natural>は、トランザクション ID です。これを使用して Natural RPC サーバーフロントエンドを起動します。 <ncistart>は、Natural CICS インターフェイスニュークリアスの名前です。

実行上の注意：

- NCMDIR パラメータ ROLLSRV が YES に設定されている場合、Natural ロールサーバー が必要です。

  （プロファイルパラメータ SUBSID で定義された）使用する subsystem-id に対して Natural ロールサーバーを起動してから、Natural RPC サーバーフロントエンドを起動する必要があります。
- RPC サーバーフロントエンドのトランザクション ID は、RPC サーバー環境を識別するために使用されます。

  同じトランザクション ID の複数の Natural RPC サーバーフロントエンドを起動しないでください。
- 実行可能 NCI モジュールは、ダイナミックにロードされます。

  実行可能 NCI モジュールを含むロードライブラリが、DFHRPL 連結で使用可能である必要があります。
- EntireX Broker スタブ NATETB23 は使用しないでください。

  EntireX Broker は、Natural コンテキストの外部でアクセスされます。 したがって、EntireX Broker スタブ CICSETB を使用する必要があります。

  CICSETB のバージョンによっては、プロファイルパラメータ RCA を使用できません。 その場合は、代わりに CICSETB を Natural CICS インターフェイスニュークリアスにリンクする必要があります。

偽装を使用する場合のみ：
偽装機能を使用する場合、RPC サーバーフロントエンドによってワーカータスクが起動されます。 CICS インストールの CICS コンフィグレーションオプション MAXTASKS に十分大きな値が設定されていることを確認します。

Natural Security の場合のみ：
Natural RPC サーバーフロントエンドをプロファイルパラメータ AUTO=OFF で開始する場合、Natural スタックで、ライブラリ ID、ユーザー ID、およびパスワードを指定して Natural LOGON コマンドを実行する必要があります。つまり、STACK=(LOGON *library-id;user-id;password*) のように指定します。

7 Natural RPC サーバーの終了

このセクションでは、Natural RPC サーバーを終了する方法について説明します。 いくつかの方法があります。

SYSRPC の使用

『*SYSRPC ユーティリティ*』ドキュメントの「サーバーの終了」で説明されているように、SYSRPC ユーティリティの TE（Terminate Server） コマンドを使用します。

Natural RPC サーバーは、サーバーが現在リモート CALLNAT を実行しておらず、会話の次の CALLNAT 要求を待機していない場合にのみ終了できます。

EntireX System Management Hub の使用

EntireX System Management Hub の Server サブツリーの［**Deregister**］ボタンを使用します。

アプリケーションプログラミングインターフェイス USR2073N の使用

アプリケーションプログラミングインターフェイス（API）USR2073N によって、Natural RPC サーバーを ping または終了できます。

インターフェイスによって、ping または終了コマンドが、ノード名およびサーバー名によって指定された RPC サーバーに送信されます。 返されたメッセージには、次の情報が含まれています。

- 実行されているサーバーのバージョン（PING）または
- 終了の確認（TERMINATE）または
- エラーメッセージ

手順 7.1. USR2073N を使用するには

1 サブプログラム USR2073N を、ライブラリ SYSEXT からライブラリ SYSTEM または steplib ライブラリまたはサーバー環境の任意のアプリケーションにコピーします。

2 DEFINE DATA PARAMETER ステートメントをストラクチャードモードで使用するか、RESET ステートメントをレポーティングモードで使用して、次のパラメータを指定します。

パラメータ	フォーマット	I/O	説明	
FUNCTION	A10	I	PING	RPC サーバーにそのバージョンを問い合わせます。
			TERMINATE	Natural RPC サーバータスクの終了を開始します。
SRVNODE	A192	I	サーバー名が EntireX のロケーショントランスペアレンシまたは LOGBROKER=*nodename* として論理 Broker ノード名を表す場合、ノード名または * を指定します。 プロファイルパラメータ RPC またはパラメータマクロ NTRPC のキーワードサブパラメータ SRVNODE を設定します。	
SRVNAME	A192	I	サーバー名または EntireX ロケーショントランスペアレンシ。 プロファイルパラメータ RPC またはパラメータマクロ NTRPC のキーワードサブパラメータ SRVNAME を設定します。	
LOGON	A1	I	ログオンフラグ Y（はい）は、ログオンデータを Natural RPC サーバーに転送する必要があることを示します。	
USER-ID PASSWORD	A8	I	LOGON=Y の場合、ユーザー ID およびパスワードが Natural Security のログオンデータです。 クライアントが Natural Security（NSC）環境で実行されておらず、ログオンフラグが設定されている場合は、データがアプリケーションプログラミングインターフェイス USR1071N 経由で入力されていない限り、ユーザー ID およびパスワードを指定する必要があります。 「*Security の使用*」の「***Natural Security での Natural RPC の使用***」を参照してください。	
MSG	A79	O	メッセージが返されました。	
RC	I2	O	レスポンスコード。設定可能値は次のとおりです。	
			0	MSG には、RPC サーバーまたは Broker からの正常メッセージが含まれています。
			1	MSG には、RPC サーバーまたは Broker からのエラーメッセージが含まれています。
			2	MSG には、インターフェイスからのエラーメッセージが含まれています。
			3	Natural Security エラー。

3 API を呼び出す前に、上述した入力変数を指定してください。

ユーザー出口 NATRPC99

この出口は、Natural RPC サーバーが登録解除し、サーバーノードからログオフした後に呼び出されます。

- NATRPC99 プログラムが見つからない場合、サーバーは通常どおりすぐに終了します。
- プログラム NATRPC99 が見つかった場合、サーバーは通常の Natural セッションとして実行を継続します。

NATRPC99 は、パラメータのない FETCH ステートメントで呼び出されます。つまり、NATRPC99 が呼び出される前に、Natural スタック上にデータは置かれません。

転送制御ステートメント（FETCH、CALLNAT、PERFORM）やプログラムを終了するステートメント（STOP、ESCAPE、TERMINATE）など、NATRPC99 にコーディングを追加できます。

RETURN または STOP ステートメントによって NATRPC99 が終了された場合、Natural は NEXT プロンプトに戻ります。 使用している環境で NEXT プロンプトがサポートされていない場合（プロファイルパラメータ CM=OFF、非同期 Natural セッションなど）、セッションは終了します。 それ以外の場合は、セッションは Natural コマンドのプライマリ入力ファイル／データセットおよび INPUT データ（CMSYNIN）から、次のコマンドを読み取ろうとします。

重要な注意：

1. NATRPC99 は Natural プログラムである必要があります。
2. NATRPC99 はシステムファイル FUSER のライブラリ SYSTEM に置く必要があります。 サーバーが現在ログオンしているライブラリの STEPLIB 連結は、NATRPC99 の検索で評価されません。
3. NATRPC99 は、『*SYSRPC ユーティリティ*』ドキュメントの「サーバーの終了」で説明されているように、現在は SYSRPC ユーティリティを使用して発行された TE（Terminate Server）コマンドでサーバーが終了された場合にのみ呼び出されます。 サーバーが **EntireX System Management Hub**. によって終了された場合は、出口は呼び出されません。
4. NATRPC99（FETCH、CALLNAT、PERFORM）によって呼び出される Natural オブジェクトは、サーバーがログオンしているライブラリまたはそのいずれかの steplib（システムファイル FUSER の SYSTEM など）に存在する必要があります。

Attach Manager を使用している場合のサーバーの終了

Natural RPC サーバーの終了動作には、プロファイルパラメータ RPC またはパラメータマクロ NTRPC のキーワードサブパラメータ SRVTERM が影響します。 デフォルトでは、前述の終了方法のいずれかが適用された場合を除き、サーバーは終了しません（SRVTERM=NEVER）。

Attach Manager を使用して Natural RPC サーバーを要求に応じてダイナミックに起動する場合は、SRVTERM を TIMEOUT に設定する必要があります。 このパラメータ設定の場合、RPC 会話外での次のクライアント要求までの待機時間が所定の時間を超えると、Natural RPC サーバーは自動的に終了します。

8 EntireX Broker サービスの終了

このセクションでは、EntireX Broker サービスを終了する方法について説明します。 いくつかの方法があります。

SYSRPC の使用

『*SYSRPC ユーティリティ*』ドキュメントの「サーバーの終了」で説明されているように、SYSRPC ユーティリティの TS（Terminate EntireX Broker Service） コマンドを使用します。

EntireX System Management Hub の使用

EntireX System Management Hub の Service サブツリーの［**Deregister**］ボタンを使用します。

アプリケーションプログラミングインターフェイス USR2075N の使用

アプリケーションプログラミングインターフェイス（API）USR2075N によって、アプリケーション内から EntireX Broker サービスを終了できます。

コマンド TERMINATE-SERVICE によって、インターフェイスは EntireX のコマンドおよび情報サービスを使用してタスクを実行します。 まず、EntireX Broker にログオン要求を送信します。 次に、サーバークラス、サーバー名、およびサービスタイプが指定された全サーバーのリストを取得します。 最後に、シャットダウン要求を各サーバーに送信します。 終了されたサーバーの数がメッセージで示されます。

手順 8.1. USR2075N を使用するには

1 サブプログラム USR2075N を、ライブラリ SYSEXT からライブラリ SYSTEM または steplib ライブラリまたはサーバー環境の任意のアプリケーションにコピーします。

2 DEFINE DATA PARAMETER ステートメントをストラクチャードモードで使用するか、RESET ステートメントをレポーティングモードで使用して、次のパラメータを指定します。

パラメータ	フォーマット	I/O	説明	
FUNCTION	A18	I	TERMINATE-SERVICE を指定します。 これにより、シャットダウン要求が、パラメータ SRVCLASS、SRVNAME、および SERVICE（サービスタイプ）によって指定された各サーバーに送信されます。下記を参照してください。	
SRVNODE	A192	I	サーバー名が、EntireXのロケーショントランスペアレンシを表す場合、または LOGBROKER=*nodename* のように論理 Broker ノード名を表す場合は、Broker 名またはアスタリスク（*）を指定します プロファイルパラメータ RPC またはパラメータマクロ NTRPC のキーワードサブパラメータ SRVNODE を参照してください。	
SRVCLASS	A32	I	サーバークラスを指定します。 Natural RPC サーバーの場合、これは RPC です。	
SRVNAME	A32/A192	I	サーバー名（A32）または EntireX ロケーショントランスペアレンシ（A192）。プロファイルパラメータ RPC またはパラメータマクロ NTRPC のキーワードサブパラメータ SRVNAME を参照してください。	
SERVICE	A32	I	サービスのタイプを指定します。 Natural RPC サーバーの場合、これは CALLNAT です。	
IMMEDIATE	L		TRUE	指定されたサーバーのすべての会話をすぐに終了します。
			FALSE	既存の会話は正常に終了されます。新しい会話は受け入れられません。
USER-ID	A32	I	EntireX Broker にログオンするためのユーザー ID を指定します。	
PASSWORD	A32	I	EntireX Broker 側で EntireX Broker Security が使用される場合、そのパスワードを指定します。	
MSG	A（ダイナミック）	O	メッセージが返されました。	
RC	I2	O	レスポンスコード。設定可能値は次のとおりです。	
			0	MSG には、EntireX Broker からの正常メッセージが含まれています。
			1	MSG には、EntireX Broker からのエラーメッセージが含まれています。
			3	MSG には、クライアント側の Natural Security からのエラーメッセージが含まれています。

3 API を呼び出す前に、上述した入力変数を指定してください。

9 Natural RPC 環境の運用

このセクションでは、Natural RPC 環境の操作に必要なタスクを説明します。

これらのタスクのいくつかは SYSRPC ユーティリティで実行されます。 SYSRPC ユーティリティが提供する機能の手順については、Natural『*SYSRPC ユーティリティ*』ドキュメントを参照してください。

RPC サーバーアドレスの指定

各リモート CALLNAT 要求に対して、CALLNAT が実行されるサーバーが割り当てられます（*servername* および *nodename* で識別される）。 したがって、リモートにアクセスされるすべてのサブプログラムは以下で定義する必要があります。

- クライアント側のローカルサービスディレクトリ
- または、リモートディレクトリサーバー経由でアクセスされるリモートディレクトリ
- または、プロファイルパラメータ RPC またはパラメータマクロ NTRPC のキーワードサブパラメータ DFS を使用したデフォルトサーバーアドレス指定
- または、デフォルトサーバーアドレッシング経由でクライアントアプリケーション自身内

上記のメソッドに加えて、代替サーバーを指定できます。

EntireX Broker を使用する場合、EntireX ロケーショントランスペアレンシを使用してサーバーを定義することも可能です。「*EntireX ロケーショントランスペアレンシの使用*」を参照してください。

以下に参考情報を示します。

ローカルディレクトリエントリの使用

クライアントのローカルサービスディレクトリのすべてのデータは、サブプログラム NATCLTGS に保存されます。 実行時、このサブプログラムは、ターゲットサーバーを検索するために使用されます。 結果として、NATCLTGS は、クライアントアプリケーションまたはアプリケーション用に定義された Natural steplib の 1 つで有効にする必要があります。

NATCLTGS が steplib に生成されなかったか、または別のマシンに存在する場合、適切な Natural ユーティリティ（SYSMAIN または Natural オブジェクトハンドラ）を使用して、NATCLTGS をアプリケーション用に定義された steplib のいずれかに移動してください。

共同使用のために NATCLTGS を使用している場合、例えば、そのサブプログラムをライブラリ SYSTEM にコピーして、すべてのクライアント環境に有効にする必要があります。または、クライアントごとに個々のコピーが使用される場合、SYSRPC ユーティリティのサービスディレクトリメンテナンス機能を使用してこのクライアントのためにメンテナンスする必要があります。

RPC サービスエントリを定義し、編集するには、『*SYSRPC ユーティリティ*』ドキュメントの「*サービスディレクトリメンテナンス*」を参照してください。

リモートディレクトリエントリの使用

リモートディレクトリには、各種Naturalクライアントに有効にできるサービスエントリが含まれています。Naturalクライアントは、リモートディレクトリサーバーからこれらのサービスエントリを検索できます。リモートディレクトリサーバーの目的およびインストールについては、「*リモートディレクトリサーバーの使用*」を参照してください。

SYSRPC リモートディレクトリメンテナンス機能については、『*SYSRPC ユーティリティ*』ドキュメントの関連するセクションを参照してください。

Natural 起動時のデフォルトサーバーアドレスの指定

ローカルまたはリモートサービスディレクトリを使用してサーバーにアドレスする代わりに、プロファイルパラメータ RPC またはパラメータマクロ NTRPC のキーワードサブパラメータ DFS（Natural『*オペレーション*』ドキュメントを参照）でデフォルトサーバーを事前に設定することができます。 サブプログラムがローカルでもリモートサービスディレクトリでも検出できない場合、このサーバーアドレスが使用されます。

DFS 設定によって、セッション全体に対するデフォルトサーバー、またはそれをダイナミックに上書きするまでのデフォルトサーバーが決定されます。

DFS 設定が存在せず、与えられたリモートプロシージャコールのサーバーアドレスがサービスディレクトリに見つからない場合、Natural エラーメッセージが返されます。

別のライブラリにログオンしたり、Naturalエラーが発生したりする場合でも、クライアントアプリケーション内で定義されたデフォルトサーバーアドレスは、アクティブな状態のままです。

Natural セッション内のデフォルトサーバーアドレスの指定

クライアントアプリケーション自体はランタイム時にデフォルトサーバーアドレスをダイナミックに指定できます。 この目的のために、Natural はアプリケーションプログラミングインターフェイス USR2007N を提供します。 このインターフェイスにより、リモートプログラムがサービスディレクトリ経由でアドレスできないときに使用されるデフォルトのサーバーアドレスを決定できます。

手順 9.1. USR2007N を使用するには

1 サブプログラム USR2007N を、ライブラリ SYSEXT からライブラリ SYSTEM または steplib ライブラリまたはサーバー環境の任意のアプリケーションにコピーします。

2 DEFINE DATA PARAMETER ステートメントをストラクチャードモードで使用するか、または RESET ステートメントをレポーティングモードで使用して、次のパラメータを指定します。

パラメータ	フォーマット	説明	
function	A1	関数。設定可能値は次のとおりです。	
		P（Put）	サーバーアドレス（パラメータ *nodename* および *servername* で構成、下記参照）が、サービスディレクトリに定義されていないすべての後続のリモートプロシージャコールに対するデフォルトアドレスであることを指定します。 デフォルトサーバーアドレスを削除するには、*nodename* および *servername* に空白を指定します。
		G（Get）	機能Pで設定した現在のデフォルトサーバーアドレスを取得します。
nodename	A192	アドレスされるサーバーノードの名前を指定／返します。 物理ノード名の場合は最大32文字、論理ノード名の場合は最大192文字をノード名に含めることができます。「*EntireX ロケーショントランスペアレンシの使用*」を参照してください。 **注意:** 互換性の理由から、*servername* は、オプション BY VALUE または BY VALUE RESULT（DEFINE DATA ステートメントの説明の「パラメータデータ定義」を参照）で定義し、*servername* の A8 フィールドを渡す既存の呼び出し元をサポートするようにします。 ライブラリ SYSEXT で提供されるサンプル USR2007P は、最大32文字をサポートします。	
servername	A192	アドレスされるサーバー名を指定／返します。 物理サーバー名の場合は最大 32 文字、論理サービス名の場合は最大 192 文字をサーバー名に含めることができます。「*EntireX ロケーショントランスペアレンシの使用*」を参照してください。 **注意:** 互換性の理由から、*nodename* は、オプション BY VALUE または BY VALUE RESULT（DEFINE DATA ステートメントの説明の「パラメータデータ定義」を参照）で定義し、*nodename* の A8 フィールドを渡す既存の呼び出し元をサポートするようにします。	

パラメータ	フォーマット	説明	
		ライブラリ SYSEXT で提供されるサンプル USR2007P は、最大32文字をサポートします。	
logon	A1	ログオンオプションを指定／返します。「*ログオンオプションの使用*」を参照してください。	
protocol	A1	トランスポートプロトコルを指定／返します。 有効な値：B（=EntireX Broker）	
noservdir	A1	サービスディレクトリオプションを指定／返します。プロファイルパラメータ RPC またはパラメータマクロ NTRPC のキーワードサブパラメータ DFS を参照してください。	
		Y	サービスディレクトリが存在していない必要があります。
		N	サービスディレクトリが存在する必要があります。

3 クライアント側の呼び出し元プログラムで、次のステートメントを指定します。

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

注意: ライブラリ SYSRPC の Natural サブプログラム NATCLTPS は、互換性保持のためにのみメンテナンスされます。

代替サーバーの使用

接続の失敗を回避するために、リモート CALLNAT のために代替サーバーを定義できます。 そのような代替サーバーを指定すると、Natural は次のように処理します。

- クライアントは最初に接続の設定を試みます。
- この試みに失敗すると、エラーメッセージの代わりに、2回目の試みが別のサーバー上で行われます。 サービスディレクトリが現在のエントリから再度検索される代わりに、指定したサービスを提供する別のサーバーが有効かどうかを検出します。
- 2番目のエントリが検出されると、Natural はこのサーバーへの接続の設定を試行します。 リモートプロシージャコールが正常に実行された場合は、クライアントアプリケーションは実行状態を保持します。 ユーザーは最初のサーバーまたは代替サーバーのどちらの接続が結果を生じたかは認識しません。
- 他のエントリが検出されなかったり、代替サーバーへの接続に失敗すると、Naturalは対応するエラーメッセージを発行します。

手順 9.2. 代替サーバーの使用を可能にするには

1 同じサービスに対するサービスディレクトリ内に複数のサーバーを定義します。

2 代替サーバーの使用を許可するために、プロファイルパラメータ RPC またはパラメータマクロ NTRPC のキーワードサブパラメータ TRYALT を ON に設定します。

このパラメータは、現在のセッションに対してダイナミックに設定することもできます。 パラメータメンテナンス機能（『*SYSRPC ユーティリティ*』ドキュメントを参照）を使用します。

EntireX ロケーショントランスペアレンシの使用

EntireX ロケーショントランスペアレンシを使用すると、何かを構成したり、クライアントやサーバーのプログラムを変更しなくても、物理的なノードおよびサーバーの名前を変更できます。 現在、物理ノードや物理サーバーの名前を使用する代わりに、論理的な名前でサーバーをアドレスすることができます。論理名は、ディレクトリサービスを使って物理的なノードとサーバーの名前にマップされます。

ロケーショントランスペアレンシを利用するため、以前にノードとサーバー名のみが指定された場合でも、Natural RPC では論理名の受け入れが可能となりました。 論理名は、最初に使用される前に EntireX Broker に渡されます。

論理名の最大長は 192 文字です。 新規 Natural プロファイルパラメータを避けるために、論理名は既存パラメータのサーバー名の部分に指定されます。 2 種類の論理名があります。

- 論理ノード名
  論理ノード名で、実際のサーバー名だけに関連してノードに対する論理名を指定します。 実際のノード名も使用できるすべての場所で、論理ノード名は使用できます。 論理ノード名を定義するにはキーワード LOGBROKER を使用する必要があります。

  例：

  ```
  SRVNVODE='LOGBROKER=logical_node_name,my_set'
  ```

- 論理サービス
  論理サービスで、ノードとサーバーの論理名を指定します。 実際のノードとサーバーの名前を使用できるすべての場所で、論理サービスは使用できます。論理サービスを定義するには、ノード名としてアスタリスク（*）を設定し（意図的に空にする）、サーバー名に論理サービス名を指定する必要があります。

  例：

  ```
  SRVNVODE='*' SRVNAME='logical_service_name,my_set'
  ```

  Natural アプリケーションプログラミングインターフェイス USR2071N が使用される場合、フィールド *broker-id* の論理サービス名とともにキーワード LOGSERVICE を使って論理サービス名に LOGON できます。

*EntireX ロケーショントランスペアレンシ*の詳細については、EntireX ドキュメントを参照してください。

次のコンポーネントはノードとサーバーの名前を参照します。

- プロファイルパラメータ RPC またはパラメータマクロ NTRPC の Natural キーワードサブパラメータ SRVNODE、SRVNAME、DFS、および RDS
- SYSRPC ユーティリティのサービスディレクトリメンテナンス機能
- サービスディレクトリ（NATCLTGS）
- Natural アプリケーションプログラミングインターフェイス USR2007N、USR2071N
- サービスプログラム RPCERR、RPCINFO

『*SYSRPC ユーティリティ*』ドキュメントの「*サービスディレクトリメンテナンス*」機能の「*ロケーショントランスペアレンシ*」も参照してください。

スタブおよび RPC 自動実行

Natural RPC 自動実行が使用される場合、「***Natural RPC** 自動実行の操作*」に説明されているスタブは必要ありません。

ただし、スタブを生成すると、リモートに実行する CALLNAT（複数可）の制御を有利にし、エラー診断を容易にします。 不正な CALLNAT 名のためにリモートコールが失敗すると、その後に発行された Natural エラーメッセージは、問題の原因を即座に識別するために役立ちます。 スタブがないと、不正な CALLNAT に対して、トランスポート層または Natural サーバーから返される追加のエラーを受け取ることがあります。

リモート CALLNAT 実行によって EntireX RPC サーバーを呼び出す場合、スタブサブプログラム（インターフェイスオブジェクト）を使用することを強くお勧めします。 EntireX RPC サーバーで呼び出すサブプログラムの IDL（Interface Definition Language）定義にグループ構造が含まれている場合、スタブサブプログラムが必要です。 この場合、［Stub Generation］画面でスタブ生成中に同じグループ構造を定義するか、EntireX IDL ファイルからスタブサブプログラムを生成する必要があります（Windows のみ）。

以下に参考情報を示します。

- **スタブサブプログラムの作成**

スタブサブプログラムの作成

SYSRPC ユーティリティのスタブ生成機能では、クライアントの呼び出し元プログラムをサーバー上のサブプログラムへ接続するために使用する Natural スタブサブプログラムを生成することができます。 スタブは、パラメータデータエリア（PDA）およびサーバーコールロジックで構成されます。『*SYSRPC ユーティリティ*』ドキュメントの*スタブ生成*に関するセクションを参照してください。

PDA は呼び出し元プログラムの CALLNAT ステートメントで使用されるパラメータと同じものを含み、スタブ生成機能の［Stub Generation］画面で定義されている必要があります。 同一名でコンパイルした Natural サブプログラムがすでに存在する場合は、このサブプログラムにより使用される PDA が画面を事前に設定するために使用されます。 サーバーコールロジックは、PDA が定義された後にスタブ生成機能により自動的に生成されます。

実行時に、CALLNAT ステートメントを持っている Natural アプリケーションプログラムおよびスタブサブプログラムは、クライアント側に存在する必要があります。 Natural アプリケーションサブプログラムは、サーバー側に存在する必要があります。 スタブおよびサーバーサブプログラムは、同じ名前にする必要があります。

Natural RPC 自動実行の操作

Natural RPC スタブの生成は必須ではありませんが、Natural RPC の自動実行（つまり、Natural スタブを使用しない）を操作できます。 Natural RPC 自動実行を操作するには、プロファイルパラメータ RPC またはパラメータマクロ NTRPC のキーワードサブパラメータ AUTORPC を次のように設定します。

```
AUTORPC=ON
```

この場合は、RPC 使用のための準備中にクライアントスタブの生成を省略することができます。Natural RPC 自動実行が有効（AUTORPC=ON）のとき、Natural は次のように動作します。

- サブプログラムがローカルで検出できない場合、Natural ではリモートでの実行が試行されます（スタブ生成の必要なし）。
- その後パラメータデータエリアは、実行中にダイナミックに生成されます。

クライアントプログラムに対してのみスタブが存在する場合、この機能はサーバー上の CALLNAT プログラムには何も影響しません。

プロファイルパラメータ RPC またはパラメータマクロ NTRPC のキーワードサブパラメータ AUTORPC が ON に設定されており、Natural スタブが存在する場合は、そのスタブが使用されます。

Natural セッション中の RPC プロファイルパラメータの修正

パラメータメンテナンス機能で、現在のセッションに対するNaturalプロファイルパラメータモジュール内の RPC プロファイルパラメータのいくつかを、セッション内でダイナミックに修正できます。

注意: これらの修正はユーザーセッションがアクティブである限り保持されます。セッションが終了すると削除されます。スタティックな設定はNaturalプロファイルパラメータを使用して行います。

サーバーコマンドの実行

サービスディレクトリで定義されたアクティブサーバー（「*RPC サーバーアドレスの指定*」を参照）は、SYSRPC サーバーコマンド実行機能で制御できます（『*SYSRPC ユーティリティ*』ドキュメントの関連するセクションを参照）。

サーバーライブラリへのログオン

CALLNAT が実行されるサーバーライブラリは、クライアント側の RPC のログオンオプションおよびサーバー側の 1 組のパラメータに依存します。

次の表に、適切なパラメータとそれらがライブラリ設定にどのように影響するかを示します。

	クライアント		サーバー				
	1	2	3	4	5	6	7
	*library-id	サーバーエントリの RPC LOGON フラグの設定	LOGONRQ の設定	STACK= で起動されるサーバー	NSC または ネイティブ Natural	NSC：ライブラリプロファイル内の RPC ログオンオプション	サーバー *library-id
1	Lib1	×	×	logon lib1	影響なし	N/--	Lib1
2	Lib1	×	×	logon lib2	影響なし	N/--	Lib2
3	Lib1	×	○	（クライアント LOGON フラグ = NO）かつ（LOGONRQ=YES）は不可。			
4	Lib1	○	影響なし	影響なし	NSC	AUTO	Lib1
5	Lib1	○	影響なし	影響なし	NSC	N	Lib1

6	Lib1	○	影響なし	影響なし	ネイティブ Natural	--	Lib1

表の項目の説明：

1. CALLNAT が開始されるクライアントアプリケーションのライブラリ ID。
2. RPC LOGON フラグの値。 ノードまたはサーバー全体に対して設定できます。

   このフラグは次のものを使用して設定できます。

   SYSRPC ユーティリティのサービスディレクトリメンテナンス機能

   プロファイルパラメータ RPC またはパラメータマクロ NTRPC のキーワードサブパラメータ DFS

   または、アプリケーションプログラミングインターフェイス USR2007N
3. プロファイルパラメータ RPC またはパラメータマクロ NTRPC のキーワードサブパラメータ LOGONRQ は、サーバー起動時に設定できます。
4. その起動時にサーバーが位置付けられるライブラリ ID。
5. サーバーは Natural Security（NSC）下で実行されるかどうか（「*Natural Security での Natural RPC の使用*」を参照）
6. NSC サーバーアプリケーションの NSC *ライブラリプロファイル項目*（*セッションオプション>Natural RPC 制限*）のログオンオプションの設定。 NSC *ログオンオプション*が A（AUTO）に設定されると、ライブラリとユーザー ID だけが取得されます。 N（デフォルト）に設定されると、ライブラリ、ユーザー ID、およびパスワードパラメータが評価されます。
7. 最終的に CALLNAT プログラムが実行されるサーバーのライブラリ。

ログオンオプションの使用

ログオンオプションは、リモートサブプログラムが実行されるライブラリを定義します。「*サーバーライブラリへのログオン*」も参照してください。

注意: ログオンオプションを使用しないとき、CALLNAT は、サーバーが現在ログオンしているライブラリで実行されます。 このサーバーログオンは、Natural プロファイルパラメータ STACK=(LOGON *library*) で定義されます。 サーバーは、*library*（および *library* に定義されたすべての関連 steplib）で実行される CALLNAT を検索します。

クライアントアプリケーションは、このサブプログラムにログオンオプションを設定することによって、異なるライブラリのサブプログラムの実行を可能にします。 これにより、クライアントは、このログオンオプションとともにその現在のライブラリの名前をサーバーに渡します。 そして、サーバーは、このライブラリへログオンし、必要なサブプログラムを検索します。後者が見つかった場合、それを実行します。 その後、前のライブラリからログオフします。

異なるライブラリへのログオン

クライアントの現在のライブラリ以外のライブラリにサーバーがログオンする必要がある場合、リモート CALLNAT が実行される前に、クライアントはアプリケーションプログラミングインターフェイス USR4008N を呼び出す必要があります。 USR4008N によって、クライアントはサーバーがログオンするライブラリの代替名を指定します。このライブラリの名前は、ログオンオプションが適用されるリモートサブプログラムへの後続のすべてのコールに使用されます。 ライブラリ名に空白が指定された場合は、現在のクライアントライブラリの名前が再度使用されます。

手順 9.3. USR4008N を使用するには

1 サブプログラム USR4008N を、ライブラリ SYSEXT からライブラリ SYSTEM または steplib ライブラリまたはサーバー環境の任意のアプリケーションにコピーします。

2 DEFINE DATA PARAMETER ステートメントを使用して、次のパラメータを指定します。

パラメータ	I/O	フォーマット	説明	
P-FUNC	I	A01	ファンクションコード。設定可能値は次のとおりです。	
			P（Put）	リモート CALLNAT 実行のために新しいライブラリを指定します。
			G（Get）	リモート CALLNAT 実行のために以前に指定されたライブラリを取得します。
P-LIB	I	A8	リモート CALLNAT 実行のためのサーバー上のライブラリ。	

3 クライアント側の呼び出し元プログラムで、次のステートメントを指定します。

```
CALLNAT 'USR4008N' P-FUNCP-LIB
```

注意: Natural RPC クライアントがリモート CALLNAT を呼び出す前に、呼び出し元プログラムを実行する必要があります。

クライアント側で必要な設定

ログオンオプションを設定するには、SYSRPC サービスディレクトリメンテナンス機能（『SYSRPC ユーティリティ』ドキュメントの関連セクションを参照）、または（デフォルトサーバーを使用しているときは）プロファイルパラメータロファイルパラメータ RPC またはパラメータマクロ NTRPC のキーワードサブパラメータ DFS やアプリケーションプログラミングインターフェイス USR2007N を使用できます。

サーバー側で必要な設定

サーバー側に設定は必要ありません。

圧縮の使用

圧縮タイプは、0、1、または 2 です。 COMPR=1 または 2 で生成したスタブは、データ転送率の削減を支援します。

圧縮タイプ	説明
COMPR=0	全 CALLNAT パラメータ値が送信され、サーバーから返信されます。つまり、圧縮は実行されません。
COMPR=1	M タイプのパラメータが送信され、サーバーから返信されます。一方、O タイプのパラメータは送信バッファ内でのみ転送されます。A タイプのパラメータは返信バッファ内にのみ含まれます。 返信バッファにはフォーマット記述は含まれません。 これはデフォルト設定です。
COMPR=2	サーバー返信メッセージが CALLNAT パラメータのフォーマット記述を含むことを除いては、COMP=1 と同じです。 これは、EntireX Broker によるデータ変換に対して特定のオプションを使用する場合に便利です（詳細については、EntireX Broker ドキュメントの変換サービスの説明を参照してください）。

Secure Socket Layer の使用

Natural RPC では、EntireX Broker への TCP/IP 通信のために Secure Socket Layer（SSL）がサポートされています。

TCP/IP 通信で SSL を使用する必要があることを EntireX Broker に認識させるには、次の方法の 1 つを使用する必要があります。

- 文字列 :SSL をノード名に付加します。 ノード名にすでに接尾辞 :TCP が付加されていた場合、:TCP を :SSL に置き換える必要があります。
- ノード名に接頭辞 //SSL: を付加します。

例：

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

必要な SSL パラメータ文字列を設定するには、SSL を使用して EntireX Broker にアクセスする前に、アプリケーションプログラミングインターフェイス USR2035N を最初に呼び出す必要があります。

手順 9.4. USR2035N を使用するには

1 サブプログラム USR2035N を、ライブラリ SYSEXT からライブラリ SYSTEM または steplib ライブラリまたはサーバー環境の任意のアプリケーションにコピーします。

2 DEFINE DATA PARAMETER ステートメントを使用して、次のパラメータを指定します。

パラメータ	I/O	フォーマット	説明	
FUNCTION	I	A01	ファンクションコード。設定可能値は次のとおりです。	

			P（Put）	新規 SSL パラメータ文字列を指定します。
				
				SSL パラメータ文字列は内部的に保存され、SSL 通信を使って EntireX Broker が初めて参照されるたびに、EntireX に渡されます。 複数の EntireX Broker 接続に対し、EntireX Broker に最初にアクセスする前に、毎回 アプリケーションプログラミングインターフェイス USR2035N を呼び出して異なる SSL パラメータ文字列を使用できます。
				
				例：
				
				`FUNCTION := 'P'`
				`SSLPARMS :=`
				`'TRUST_STORE=FILE://DDN:CACERT&VERIFY_SERVER=N'`
				`CALLNAT 'USR2035N' USING FUNCTION SSLPARMS`
				
				Natural RPC サーバーの場合に SSL パラメータを設定するには、サーバーを起動するときに、呼び出し元プログラムの名前を Natural スタックに置いてください。
				
				例：
				
				`STACK=(LOGON` *`server-library`*`;`*`set-SSL-parms`*`)`
				
				set-SSL-parms は、SSL パラメータ文字列を設定するためにユーザーアプリケーションプログラミングインターフェイス USR2035N を呼び出す Natural プログラムです。

			G（Get）	前に指定した SSL パラメータ文字列を取り出します。 前に指定した SSL パラメータ文字列は、呼び出し元に返されます。 SSL パラメータ文字列の詳細については、EntireX ドキュメントを参照してください。
SSLPARMS	I	A128	EntireX Broker に必要な SSL パラメータ文字列	

3 クライアント側の呼び出し元プログラムで、次のステートメントを指定します。

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

RPC セッションのステータスのモニタ

次のセクションで構成されています。

- RPCERR プログラムの使用
- RPCINFO サブプログラムの使用
- サーバートレース機能の使用
- トレースファイルの定義

RPCERR プログラムの使用

RPCERR プログラムは、コマンド行から実行するか、または Natural プログラム内から FETCH ステートメントを使用して呼び出すことができます。 RPCERR によって次の情報が表示されます。

- RPC に関連する場合に最後の Natural エラー番号およびメッセージ
- このエラーに関連付けられた最後の EntireX Broker メッセージ

さらに、最後の EntireX Broker コールからのノードおよびサーバー名も取得可能です。

RPC エラー表示例：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
```

RPCINFO サブプログラムの使用

現在の RPC セッションの状態に関する情報を取得するために、アプリケーションプログラム内で RPCINFO サブプログラムを使用できます。 これは、特定のエラークラスに対処することでより適切にエラーを制御することもできます。

RPCINFO サブプログラムは、RPCINFO を呼び出すサンプルプログラム TESTINFO とともに SYSRPC ライブラリ 内に含まれます。

例：

```
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
```

RPC Info のパラメータ

RPCINFO には、パラメータデータエリア RPCINFOL 内で提供されている次のパラメータがあります。

パラメータ	フォーマット	説明
SERVER-PARMS		サーバーとして動作しているときに、Natural セッションに関する情報を含みます 。 RPC サーバー上でリモートに RPCINFO を実行する場合のみ SERVER-PARMS を適用します。
S-BIKE	A1	使用されたトランスポートプロトコル。設定可能値は次のとおりです。
		B EntireX Broker
S-NODE	A8	サーバーのノード名です。

パラメータ	フォーマット	説明
S-NAME	A8	サーバー名です。
S-ERROR-TEXT	A80	トランスポート層から返されたメッセージテキストを含みます。
S-CON-ID	I4	現在の会話 ID です。 これは、論理 Natural ID ではなく、EntireX Broker の物理的な ID であることに注意してください。 このパラメータに含まれる値は常に、会話型および非会話型の両方のコールに対して EntireX Broker が生成する ID です。 物理会話 ID が非数値または I4 より大きい場合は、-1 が返されます。
S-CON-OPEN	L	オープンな会話があるかどうかを示します。 会話が継続中の場合は、このパラメータに値 TRUE が含まれます。それ以外の場合は、FALSE が含まれます。
CLIENT-PARMS		クライアントとして動作しているときに、Natural セッションに関する情報を含みます。 RPC クライアント上でリモートに RPCINFO を実行する場合のみ CLIENT-PARMS を適用します。
C-BIKE	A1	使用されたトランスポートプロトコル。設定可能値は次のとおりです。
		B EntireX Broker
C-NODE	A8	事前にアドレスされたサーバーのノード名です。
C-NAME	A8	事前にアドレスされたサーバー名です。
C-ERROR-TEXT	A80	トランスポート層から返されたメッセージテキストを含みます。
C-CON-ID	I4	最後のサーバーコールの会話 ID です。 これは、論理 Natural ID ではなく、EntireX Broker の物理的な ID であることに注意してください。 会話が開かれていない場合は、このパラメータの値は 0 以下になります。 物理会話 ID が非数値または I4 より大きい場合は、-1 が返されます。
C-CON-OPEN	L	オープンな会話があるかどうかを示します。 会話が継続中の場合は、このパラメータに値 TRUE が含まれます。それ以外の場合は、FALSE が含まれます。

サーバートレース機能の使用

Natural RPC には、サーバーアクティビティのモニタおよび可能性のあるエラー状況のトレースを有効にするトレース機能が含まれています。

サーバートレース機能の起動／停止

サーバートレース機能を起動／停止するには、以下のオプションでサーバーを開始します。

```
TRACE=n
```

整数値 *n*" は、必要なトレースレベルを意味します。つまり、トレースに記録されるサーバーの詳細の程度です。 可能な値は次のとおりです。

値	トレースレベル
0	トレースは実行されません（デフォルト）。
1	すべてのクライアント要求および対応するサーバーレスポンスがトレースされ、記録されます。
2	すべてのクライアント要求および対応するサーバーレスポンスがトレースされ、記録されます。さらに、全 RPC データがトレースファイルへ書き込まれます。

RPC トレース機能によって、Natural レポート番号 10 にトレースデータが書き込まれます。

変換エラーが Natural エラー番号 NAT6974 および理由コード 2 と 3 でレポートされた場合は、エラーが含まれている可能性があるデータのバッファにおける位置が示されます。

RPC サーバーのトレースに対する TS=ON のサポート

次の情報は、メインフレーム環境にのみ適用されます。

Natural RPC サーバーセッションで `TS=ON` が指定された場合、Natural RPC サーバートレース内のすべてのメッセージは大文字に変換されます。 クライアント間のデータのトレースに `TS=ON` の影響はなく、何も変更されることはありません。

トレースファイルの定義

トレースファイル定義は環境に依存します。

- メインフレーム環境のトレースファイル処理 - 全般的な情報
- z/OS バッチモードでのトレースファイル処理
- CICS 環境のトレースファイル処理
- z/VSE バッチモードでのトレースファイル処理
- BS2000/OSD バッチモードでのトレースファイル処理
- UNIX および OpenVMS 環境のトレースファイル処理

- Windows 環境のトレースファイル制御

メインフレーム環境のトレースファイル処理 - 全般的な情報

メインフレームでは、環境に適したトレースファイルを定義します。『パラメータリファレンス』ドキュメントの NTPRINT マクロも参照してください。

z/OS バッチモードでのトレースファイル処理

a）単一タスクとしてのサーバーの実行

サーバー起動ジョブで、z/OS データセットを Natural 追加レポート CMPRT10 に割り当てます。

例：

```
//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）レプリカでのサーバーの実行

1. RPC パラメータ NTASKS を 1 より大きな値に設定します。
2. CMPRMIN を DISP=SHR のデータセットまたは * に割り当てます。
3. 各タスクが別々の CMPRINT データセットに書き込むため、次の DD カード名を定義します。

   CMPRINT（メインタスク用）

   CMPRINT1～CMPRINT9（最初の 9 個のサブタスク用）

   CMPRIN10～CMPRIN*nn*（次の 2 桁の番号のサブタスク用）。*nn*=NTASKS-1。
4. プロファイルパラメータ RPC またはパラメータマクロ NTRPC のキーワードサブパラメータ TRACE が設定されると、トレース機能によってプリンタ 10 に書き込まれます。

   次の DD カード名を定義する必要があります。

   CMPRT10（メインタスク用）

   CMPRT101～CMPRT1*nn*（すべてのサブタスク用）*nn*=NTASKS-1。

例：

```
//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
/*
```

CICS 環境のトレースファイル処理

CICS環境では、トレースファイルは出力ファイル10に書き込まれます（このファイルが存在する場合）。 出力ファイル 10 が定義されていない場合、トレースファイルはワークファイル 10 に書き込まれます（このファイルが存在する場合）。 出力ファイル 10 もワークファイル 10 も定義されていない場合は、トレースは無効です。

Natural出力ファイルは、特別パーティション一時データキューに割り当てる必要があります。

例：

Natural 定義：

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

CICS 定義：

```
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 起動 JCL：

```
RPCTRACE DD    SYSOUT=*
```

z/VSE バッチモードでのトレースファイル処理

z/VSE バッチモードでは、トレースファイルをプリンタ番号 10 に割り当てます。

例：

```
// 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)
/*
```

vrs は、バージョン、リリース、システムメンテナンスレベルを表します。

BS2000/OSD バッチモードでのトレースファイル処理

BS2000/OSD バッチモードでは、トレースファイルをプリンタ番号 10 に割り当てます。

例：

```
/.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)
```

UNIX および OpenVMS 環境のトレースファイル処理

個別にトレースできるように、サーバーごとに異なるファイル名（つまり、異なる NATPARM パラメータファイル）を使用することをお勧めします。トレースファイルは、Naturalサーバーの NATPARM パラメータファイルに定義されます。

1. レポート割り当て
   論理デバイス LPT10 をレポート番号 10 に割り当てます。
2. デバイスパラメータ割り当て
   LPT10 に物理プリンタ指定を選択する代わりに、トレースファイルの名前を示すファイル名を指定します。

UNIX の例：

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

filename は、トレースファイルの名前です。

OpenVMS の例：

```
nattmp:filename
```

filename は、トレースファイルの名前です。

Windows 環境のトレースファイル制御

個別にトレースできるように、サーバーごとに異なるファイル名（つまり、異なる NATPARM パラメータファイル）を使用することをお勧めします。トレースファイルは、Naturalサーバーの NATPARMパラメータファイルに定義されます（『コンフィグレーションユーティリティ』の「デバイス／レポート割り当て」を参照）。

1. レポート

   論理デバイス LPT10 をレポート番号 10 に割り当てます。
2. デバイス

   LPT10 に物理プリンタ指定を選択する代わりに、トレースファイルの名前を示すファイル名を指定します。デフォルトで、同じ名前で新しいファイルが作成されるときに、古いトレースファイルは削除されます。

   既存のファイルに新しいログを追加する場合は、次のように指定します。

   >>*filename*

サーバーのランタイム設定の取得

Naturalアプリケーションプログラミングインターフェイス（API）USR4010Nによって、次のようなサーバーのランタイム設定を取得できます。

- FUSER、FNAT、および FSEC のシステムファイル割り当て
- steplib チェーン

▶手順 9.5. USR4010N を使用するには

1 サブプログラム USR4010N を、ライブラリ SYSEXT からライブラリ SYSTEM または steplib ライブラリまたはサーバー環境の任意のアプリケーションにコピーします。

2 DEFINE DATA PARAMETER ステートメントを使用して、次のパラメータを指定します。

パラメータ	フォーマット	説明
FUSER-DBID	N5	システムファイル FUSER のデータベース ID。
FUSER-FNR	N5	システムファイル FUSER のファイル番号。
FNAT-DBID	N5	システムファイル FNAT のデータベース ID。
FNAT-FNR	N5	システムファイル FNAT のファイル番号。
FSEC-DBID	N5	システムファイル FSEC のデータベース ID。
FSEC-FNR	N5	システムファイル FSEC のファイル番号。
STEP-NAME	A8/15	steplib の名前。
STEP-DBID	N5/15	steplib のデータベース ID。
STEP-FNR	N5/15	steplib のファイル番号。

3 クライアント側の呼び出し元プログラムで、次のステートメントを指定します。

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

CALLNAT ステートメントの「構文の説明」も参照してください。

4 RPC パラメータ AUTORPC=OFF の場合、スタブ USR4010X をクライアント環境にコピーします。

RPC パラメータ AUTORPC=ON の場合、API はクライアント環境で使用可能にできません。使用可能にすると、API はローカルで呼び出されます。

USR4010N が呼び出されるとき、上記の指定されたパラメータの値はフィールドグループ USR4010-PARM に出力されます。

EntireX の Setting/Getting パラメータ

アプリケーションプログラミングインターフェイス（API）USR4009N によって、Natural RPC で現在サポートされている EntireX パラメータを設定または取得できます。次のものがあります。

- 圧縮レベル
- 暗号化レベル

手順 9.6. USR4009N を使用するには

1 サブプログラム USR4009N を、ライブラリ SYSEXT からライブラリ SYSTEM または steplib ライブラリまたはサーバー環境の任意のアプリケーションにコピーします。

2 DEFINE DATA PARAMETER ステートメントを使用して、次のパラメータを指定します。

パラメータ	フォーマット	I/O	説明	
FUNCTION	A01	I	関数。設定可能値は次のとおりです。	
			G（Get）	EntireX パラメータにすでに設定されている値が返されます。 Natural セッションの前に PUT が呼び出されていない場合、すべての値はゼロまたは空白です。
			P（Put）	EntireX パラメータに対して指定した値が保存され、EntireX の後続のすべてのコールで使用されます。
ENVIRONMENT	A01	I	環境。設定可能値は次のとおりです。	
			S	サーバー
			C	クライアント
			B	両方
COMPRESSLEVEL	A01	I/O	圧縮レベル。	
ENCRYPTION-LEVEL	I01	I/O	暗号化レベル。	
ACIVERS	B02	O	使用される ACI バージョン。	
RC	B01	O	リターンコード（ゼロではない場合）。要求されたパラメータを設定するために必要な ACI バージョンが含まれています。	
			0	機能は成功しました。
			6	暗号化レベルに ACI バージョン 6 が必要です。
			7	圧縮レベルに ACI バージョン 7 が必要です。

3 インターフェイスは次の 2 つの方法で呼び出すことができます。

1. プログラム内から：

```
CALLNAT 'USR4009N' FUNCTION ENVIRONMENT

                   COMPRESSLEVEL

                   ENCRYPTION-LEVEL

                    ACIVERS RC
```

2. コマンドプロンプトから、または上記のパラメータの値とともにステートメント STACK を使用して：

   例：

```
USR4009P P,C,ENCRYPTION-LEVEL=1
USR4009P P,C,,2
USR4009P P,C,ENCRYPTION-LEVEL=1,COMPRESSLEVEL=6
```

コマンドモードで、*keyword=value* 表記を使用して EntireX パラメータのサブセットのみを設定できます。 参照されないパラメータの値は、変更されないままです。

注意：

- 現在の Natural セッションで使用されている ACI バージョンが、要求されたパラメータをサポートするのに十分に大きくない場合は、要求は拒否されて値は保存されません。 この場合、必要な ACI バージョンが RC に含まれます。
- EntireX パラメータは、Natural RPC でのみ有効です。

エラー処理

- ユーザー出口 NATRPC01

リモートエラー処理

サーバー上の Natural エラーは、次のようにクライアントへ返されます。

- Natural RPC は、適切なエラー番号を *ERROR-NR システム変数へ移動します。
- エラーがローカルで発生したかのように Natural が対処します。

注意: プロファイルパラメータ RPC またはパラメータマクロ NTRPC のキーワードサブパラメータ AUTORPC が ON に設定されており、サブプログラムがローカル環境内で検出できない場合は、Natural はこれをリモートプロシージャコールであると解釈します。 その後、サービスディレクトリ内でこのサブプログラムの検出を試行します。 そこで検出されない場合は、NAT6972 エラーが発行されます。 そのため、サブプログラムが検出できない場合に NAT0082 エラーは発行されません。

「*RPCERR プログラムの使用*」も参照してください。

サーバープログラムからのエラーメッセージ NAT3009 の回避

サーバーアプリケーションプログラムが長時間データベースコールを発行しない場合は、次のデータベースコールが NAT3009 エラーメッセージを返します。

この問題を回避するには、次の手順に従います。

1. SYSRPC ライブラリ内の NATRPC39 プログラム内に FIND FIRST または HISTOGRAM ステートメントを追加します。
2. 更新したプログラムを FUSER のライブラリ SYSTEM へコピーします。

   サーバーが現在ログオンしているライブラリの STEPLIB 連結は評価されません。

ユーザー出口 NATRPC01

ユーザー出口 NATRPC01 は、Natural がエラーを発行したとき、実際には、エラーが Natural RPC ランタイムによって処理された後、およびレスポンスがクライアントに送り返される直前に呼び出されます。 これは、エラートランザクションと同じ論理ポイント、つまり、Natural エラー処理の終了時、すべての ON ERROR ブロックが処理された後に出口が呼び出されることを意味します。

エラートランザクションと違って、この出口は CALLNAT ステートメントで呼び出されます。したがって、これはサブプログラムであり、その呼び出し元に戻る必要があります。

この出口へのインターフェイスは、エラートランザクションのインターフェイスに類似しています。 さらに、出口は、Natural RPC ランタイムによってトレースされる情報を 10 行まで戻すことができます。 空白以外の文字で始まる行だけがトレースされます。

重要な注意：

1. NATRPC01 は FUSER のライブラリ SYSTEM に存在させる必要があります。 サーバーが現在ログオンしているライブラリの STEPLIB 連結は評価されません。
2. DEFINE DATA PARAMETER ステートメントブロックを変更しないでください。

サービス実行の前後のユーザー出口

管理者がサービスの実行（リモート CALLNAT）をより制御できるようにするために、オプションの 2 つのユーザー出口が Natural RPC サーバー側で呼び出されます。

ユーザー出口	目的
NATRPC02	オプションのサービス実行前出口 NATRPC02 が、サービスが実行される直前に呼び出されます。 この時点で、要求はすべてのセキュリティチェックをパスしており、データは整列解除されています。
NATRPC03	オプションのサービス実行後出口 NATRPC03 が、サービスから正常に戻された直後に呼び出されます。 この時点で、データはまだ整列されていません。 処理されないエラーが発生した場合、出口は呼び出されません。

これらの出口は相互に独立しており、別々に使用できます。

両方の出口に次のルールが適用されます。

- 出口は、FUSER システムファイルのライブラリ SYSTEM に置く必要があります。

  Natural RPC サーバーの起動時に出口が見つかった場合、出口の有効化を示すメッセージが Natural RPC サーバートレースに書き込まれます。 出口は、後で無条件に呼び出されます。 サーバーセッションの存続期間に出口が削除された場合は、永久 NAT0082 エラーが発生します。

  Natural RPC サーバーの起動時に出口が見つからなかった場合、サーバーセッションの存続期間に出口が呼び出されることはありません。 出口はダイナミックに有効にできません。
- 出口は、ユーザーによってサブプログラムとして実装される必要があります。 出口は、単一のダイナミック変数によってパラメータとして呼び出されます。ダイナミック変数の内容は、リモートサブプログラムの 8 文字の名前です。

  ダイナミック変数を使用すると、ユーザーによって書かれた既存の出口での問題を起こすことなく、渡される情報についての将来の拡張を実装できます。
- 出口は、会話内でも呼び出されます。
- Natural RPC サーバーは、出口内の処理されないエラーをインターセプトしません。 処理されないエラーが出口で発生した場合、エラーはクライアントに伝播されます。

出口は、監査やトレースのために使用できます。NATRPC02 を追加のセキュリティチェックのために使用することもできます。

NATRPC02 の例：

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

10 会話型 RPC の使用

会話を開く

手順 10.1. 会話を開くには、次の手順に従います。

1 クライアント側で OPEN CONVERSATION ステートメントを指定します。

2 OPEN CONVERSATION ステートメント内で、この会話のメンバとしてサービス（サブプログラム）のリストを指定します。

OPEN CONVERSATION ステートメントは、システム変数 *CONVID に一意の会話識別子を割り当てます。

複数の会話を並行して開くこともできます。 サブプログラムが相互にインターフェイスとなる場合は、アプリケーションプログラムは CALLNAT 指示により評価される適切な *CONVID の設定でさまざまな会話の管理を行います。

- サブプログラムが現在の会話（*CONVIDにより参照）のメンバである場合は、この会話のために 排他的に確保したサーバータスクで実行されます。
- 現在の会話のメンバではない場合は、異なるサーバータスク内で実行されます。 これは、異なる会話へも適用します。

会話は任意のプログラムレベルで開くことができ、この会話内の CALLNAT は他の任意の上位または下位プログラムレベルで実行できます。

サーバーがエージェントの役割を果たすよう、サーバー上で実行されたリモート CALLNAT でクライアント会話を開くことができます。 クライアントはサーバーの会話ではなく独自の会話のみを制御するため、アプリケーションプログラマは、メインクライアントが閉じられる前にサーバー上の会話を正常に閉じることを保証する必要があります。

追加制限

会話型RPCをローカルにテストすることが可能です。 会話型CALLNATをリモートまたはローカルに実行する場合に同一の動作を保持するには、次の追加制限を適用します。

- この会話のメンバとして現在実行しているオブジェクト内では、CLOSE CONVERSATION は行うことができません。これは、リモートに実行しているプログラム内から会話を閉じることができないという制限に相当します。
- この会話の別（または同じ）メンバ内から、会話のメンバである会話型 CALLNAT を実行することはできません。 これは、サーバーサブプログラムからクライアントの会話のメンバである会話型 CALLNAT を実行できないという制限に相当します。
- 他の会話のサブプログラム内から会話を開くことは推奨できません。

会話を閉じる

手順 10.2. 会話を閉じるには、次の手順に従います。

- クライアント側で CLOSE CONVERSATION ステートメントを指定します。

これにより、クライアントは特定の会話またはすべての会話を閉じることができます。 閉じられた会話の全コンテキスト変数は、その後解放され、サーバータスクは別のクライアントのために再度有効になります。

Natural を終了すると、全会話は自動的に閉じます。

サーバーが CLOSE CONVERSATION 要求を受け取ると、CLOSE CONVERSATION ALL ステートメントが発行され、エージェントとして開かれていたサーバーを含む全会話も閉じられます。

暗黙の BACKOUT TRANSACTION（**Rollback**）で会話を閉じる

デフォルトで、CLOSE CONVERSATION ステートメントが実行されとき、Rollback オプションは CLOSE CONVERSATION ステートメントでサーバーに送られます。 これにより、会話処理の終わりにサーバー側で暗黙の BACKOUT TRANSACTION が生じます。

暗黙の END TRANSACTION（**Commit**）で会話を閉じる

サーバー側で暗黙の END TRANSACTION を起こすように ライブラリ SYSEXT で有効なインターフェイス USR2032N を使用できます。

次の CLOSE CONVERSATION ステートメントが実行される前に、出口を呼び出す必要があります。 結果として、CLOSE CONVERSATION ステートメントとともに Commit オプションがサーバーに送られ、会話処理の終わりにサーバーで END TRANSACTION ステートメントが実行されます。

Commitオプションは、クライアントアプリケーションにより実行される次のCLOSE CONVERSATION ステートメントに適用されます。 会話（複数可）が閉じた後に、デフォルトオプションは再度使用されます。 これは、次の CLOSE CONVERSATION ステートメントが再び BACKOUT TRANSACTION ステートメントを結果として生じることを意味しています。

会話コンテキストの定義

会話中、この会話のメンバであるサブプログラムは、このサーバー上でコンテキストエリアを共有することができます。

このことを行うには、関係している各サブプログラム内に DEFINE DATA CONTEXT ステートメントでデータエリアを宣言します。

会話がローカルまたはリモートであった場合は、コンテキストエリアを使用するサブプログラムは同じ方法で動作します。DEFINE DATA CONTEXT ステートメントは、DEFINE DATA INDEPENDENT ステートメントにほぼ一致します。 AIV 変数の定義に適用されるすべてのルールは、接頭文字としてプラス記号（+）の使用が必要でないことを除き、コンテキスト変数にも適用されます。

（通常は AIV で）このアプリケーションに対する全定義を含むプログラムの実行により変数の作成を必要とするため、コンパイラはフォーマット／長さ定義のチェックを行いません。 RPC サービスルーチンを含むライブラリは通常アプリケーション依存ではないため、これはコンテキスト変数に対して何も意味を持ちません。

AIV と対比して、呼び出し元のコンテキスト変数は CALLNAT 境界を渡って送信されません。 コンテキスト変数は名前で参照され、コンテキスト ID で適用します。 1 コンテキスト変数は、1 会話内の同じ変数名を参照する全サービスルーチンにより共有されます。 つまり、各会話は独自のコンテキスト変数の集合を持ちます。 コンテキスト変数は、同じ変数名であっても異なる会話間では共有できません。

OPEN CONVERSATION ステートメントまたは非会話型 CALLNAT ステートメントが実行されたときに、コンテキストエリアは初期値にリセットされます。

システム変数 *CONVID の修正

システム変数 *CONVID（フォーマット I4）は OPEN CONVERSATION ステートメントにより設定され、アプリケーションプログラムにより修正できます。

*CONVID の修正は、複数会話を並行して使用している場合にのみ必要です。

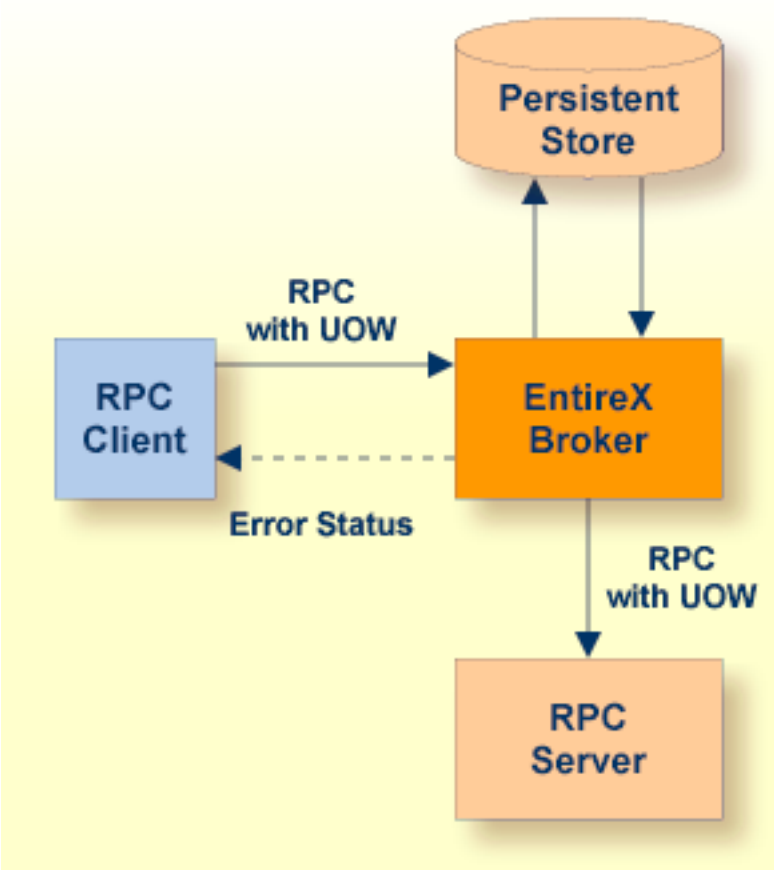
11 Reliable RPC

全般的な情報

近年、電子商取引アプリケーション（SOA）では、緩く結合されたシステムの重要性が増しています。 このタイプのシステムにとって、信頼性の高いメッセージングは重要なテクノロジとなります。

Reliable RPC は、信頼性の高いメッセージングシステムを Natural RPC で実装するものです。 Reliable RPC は Natural RPC のテクノロジと持続性を兼ね備えており、EntireX Broker によって提供される作業単位（UOW）を使用して実装されます。 Reliable RPC の特徴は次のとおりです。

- Natural RPC クライアントは、サーバーからの応答を待たずに `CALLNAT` ステートメントを実行します（RPC メッセージは非同期モードで送信される）。
- `CALLNAT` が実行されたときに RPC サーバーがアクティブである必要はありません。
- Reliable RPC メッセージは、RPC サーバーが利用可能になるまで Broker のパーシスタントストアに格納されます。
- Natural RPC サーバーは、要求されたサブプログラムをコールして Reliable RPC を実行しますが、RPC クライアントに応答を送信しません。
- Natural RPC クライアントは、送信済みの Reliable RPC メッセージのステータスを要求できます。
- Natural RPC クライアントは、Reliable RPC メッセージを EntireX RPC サーバーに送信できます。
- Natural RPC サーバーは、Reliable RPC メッセージを EntireX RPC クライアントから受信できます。



Natural RPC クライアント側の Reliable RPC

Reliable RPC 対応の Natural RPC クライアントは、通常の Natural RPC の場合と同じように構成します。同じ Natural RPC クライアントセッションで、普通の RPC 要求も Reliable RPC メッセージも送信できます。

Natural RPC クライアントで Reliable RPC を使用できるようにするには、Natural RPC クライアントでアプリケーションプログラミングインターフェイス `USR2071N` を使用して、明示的な EntireX Broker ログオンに対応する必要があります。したがって、`RPC/NTRPC` キーワードサブパラメータ `ACIVERS` を `2` 以上に設定する必要があります。

Reliable RPC は、EntireX Broker のパーシスタントサービスにメッセージを送信するために使用されます。メッセージは呼び出し元の PDA によって記述され、出力パラメータのみが含まれます。パラメータは、次のいずれかの方法で「出力」として定義されます。

- Natural スタブサブプログラム（インターフェイスオブジェクト）が使用される場合：

  `SYSRPC` ユーティリティの［Stub Generation］画面の `Attr` フィールドで、パラメータの属性を `O`（出力）に設定します。

- Natural スタブサブプログラムが使用されない場合：

  CALLNAT ステータスで AD=O セッションパラメータを使用します。

Natural スタブサブプログラムを IDL ファイルから作成する場合、パラメータの属性は IDL ファイルから取得されます。この場合、IDL ファイルには（クライアントにとっての）受信パラメータのみ含まれていることが必要です。

Reliable RPC はランタイムに有効になります。クライアントは、Reliable RPC 要求を発行する前に、次のいずれかのモードを設定する必要があります。

- AUTO_COMMIT
- CLIENT_COMMIT

AUTO_COMMIT モードでは各メッセージが送信後に暗黙でコミットされる一方、CLIENT_COMMIT モードでは、1 つの作業単位（UOW）の形で送信される一連の RPC メッセージを明示的にコミットまたはロールバックできます。

この目的のため、Natural には 2 つのアプリケーションプログラミングインターフェイス USR6304N および USR6305N が用意されています。インターフェイス USR6304N を使用すると、Reliable RPC のモードが設定されます。インターフェイス USR6305N を使用すると、CLIENT_COMMIT で作成された作業単位をコミットまたはロールバックできます。

手順 11.1. USR6304N を使用するには

1. サブプログラム USR6304N を、ライブラリ SYSEXT から、ライブラリ SYSTEM、steplib ライブラリ、またはクライアント環境の任意のアプリケーションにコピーします。
2. DEFINE DATA PARAMETER ステートメントを使用して、次のパラメータを指定します。

パラメータ	I/O	フォーマット	説明	
P-FUNC	I	A01	ファンクションコード。設定可能値は次のとおりです。	
			P（Put）	Reliable RPC のモードを設定します。このモードはその後すべてのコールに適用されます。
			G（Get）	以前指定されたモードを取得します。
P-MODE	I/O	N01	Reliable RPC のモード：	
			0	非 Reliable RPC（普通の RPC 実行）
			1	Reliable RPC AUTO_COMMIT
			2	Reliable RPC CLIENT_COMMIT
P-RC	O	N04	リターンコード	
P-MESSAGE	O	A80	メッセージテキスト	

3 クライアント側の呼び出し元プログラムで、次のステートメントを指定します。

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

注意: Reliable RPC メッセージが送信されたがまだコミットまたはロールバックされていない場合、モード CLIENT_COMMIT は変更できません。

手順 11.2. USR6305N を使用するには

1 サブプログラム USR6305N を、ライブラリ SYSEXT から、ライブラリ SYSTEM、steplib ライブラリ、またはクライアント環境の任意のアプリケーションにコピーします。

2 DEFINE DATA PARAMETER ステートメントを使用して、次のパラメータを指定します。

パラメータ	I/O	フォーマット	説明	
P-FUNC	I	A08	ファンクションコード。設定可能値は次のとおりです。	
			COMMIT	送信された Reliable RPC メッセージをコミットします。 これによりこのメッセージは RPC サーバーで使用できるようになります。
			ROLLBACK	送信済みの Reliable RPC メッセージを破棄します。
P-RC	O	N04	リターンコード	
P-MESSAGE	O	A80	メッセージテキスト	

3 クライアント側の呼び出し元プログラムで、次のステートメントを指定します。

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

Natural RPC サーバー側の Reliable RPC

Reliable RPC 対応の Natural RPC サーバーは、通常の Natural RPC の場合と同じように構成します。 同じ Natural RPC サーバーセッションで、普通の RPC 要求も Reliable RPC メッセージも処理できます。

Reliable RPC メッセージの処理を有効にするには、RPC/NTRPC キーワードサブパラメータ ACIVERS を 2 以上に設定する必要があります。

Reliable RPC メッセージのステータスの表示

送信済みの Reliable RPC メッセージのステータスを表示するため、Natural にはアプリケーションプログラミングインターフェイス USR6306N が用意されています。 USR6306N を使用すると、自分のユーザー ID で以前送信したすべての Reliable RPC メッセージのステータスを取得できます。 USR6306N は、Reliable RPC メッセージが送信された Natural セッションでコールすることが必須というわけではありません。 USR6306N が他の Natural セッションで使用された場合、まずアプリケーションプログラミングインターフェイス USR2071N を使用して、Reliable RPC メッセージの送信に使用されたものと同じユーザー ID で EntireX Broker にログインする必要があります。

Reliable RPC メッセージは、EntireX Broker 作業単位（UOW）によって実装されます。 したがって、Reliable RPC メッセージに関する情報は、UOW に関する情報になります。

手順 11.3. USR6306N を使用するには

1 サブプログラム USR6306N を、ライブラリ SYSEXT から、ライブラリ SYSTEM、steplib ライブラリ、またはクライアント環境の任意のアプリケーションにコピーします。

2 DEFINE DATA PARAMETER ステートメントを使用して、次のパラメータを指定します。

パラメータ	I/O	フォーマット	説明
P-UOW-ID-IN	I	A16	取得する UOW の ID。 可能な値は次のとおりです。 UOW の UOWID LAST： 現在の Natural セッションの最後の Reliable RPC メッセージの UOW ALL またはブランク： ログオンしている EntireX Broker ユーザーのすべての UOW
P-USER-ID	O	A32	UOW を作成したユーザーのユーザー ID。
P-BROKER-ID	O	A32	UOW をホストする EntireX Broker の Broker ID。
P-UOW-COUNT	O	I4	P-UOW-INFO 配列内の UOW 数。
P-UOW-INFO (/1:*)			各 UOW に関する情報を含む X-array
P-UOW-ID	O	A32	UOW の ID
P-UOW-STATUS	O	A10	EntireX Broker にとっての UOW のステータス このステータスは処理状態に応じて変わり、EntireX Broker によって指定されます。
P-USERR-STATUS	O	A32	UOW に関するユーザー情報。 通常は、RPC サーバーによって設定されたエラー情報です。

パラメータ	I/O	フォーマット	説明
P-CREATE-TIME	O	A32	EntireX Broker にとっての UOW の作成時刻。
P-RC	O	N04	リターンコード
P-MESSAGE	O	A80	メッセージテキスト

3 クライアント側の呼び出し元プログラムで、次のステートメントを指定します。

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

12 リモートディレクトリサーバーの使用 - RDS

RDS オペレーションの原則

サービスディレクトリを使用するために 2 つのオプションがあります。

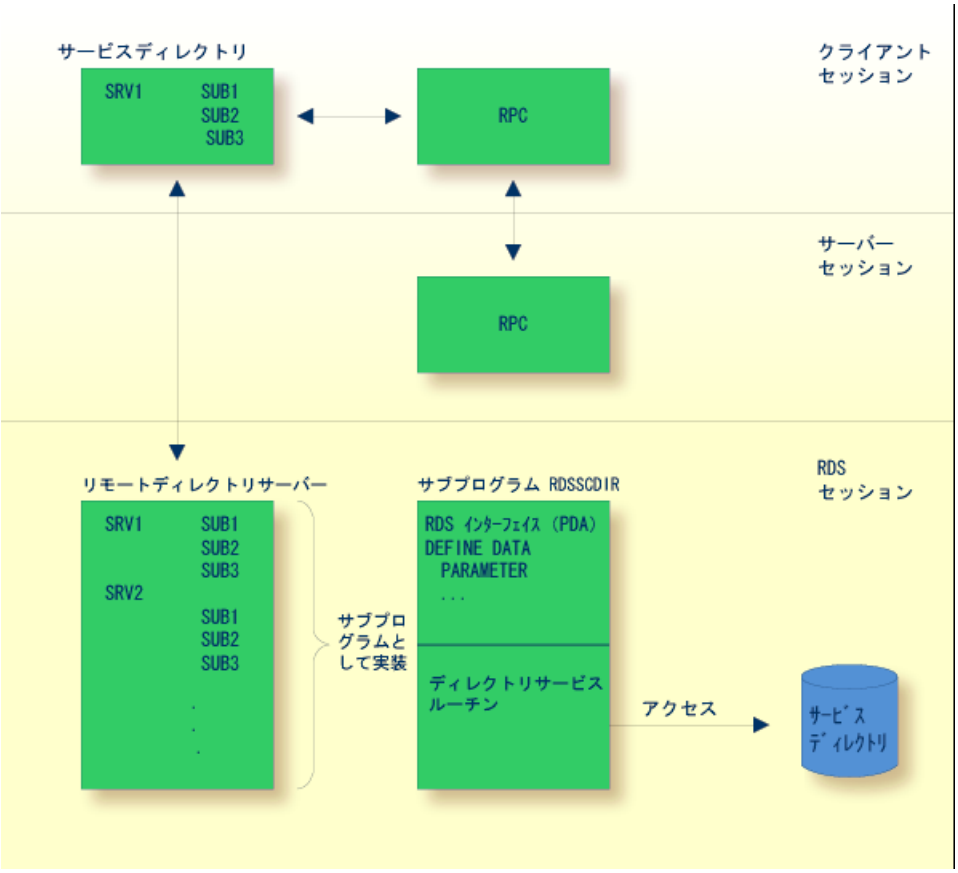
1. **Natural** サブプログラム内のサービスディレクトリの使用

   通常、サービスを見つけるには、Natural RPC では Natural サブプログラム内のサービスディレクトリを使用します。 このディレクトリは、SYSRPC *リモートディレクトリメンテナンス* サービスディレクトリメンテナンス機能で生成した NATCLTGS プログラム内の初期化した LDA データ構造であり、すべての RPC クライアントアプリケーションに有効にする必要があります。

2. リモートディレクトリの使用

   サービスを見つけるには、リモートディレクトリを使用します。 リモートディレクトリサーバー（RDS）を使用すると、そのサービスを RPC 環境内の全クライアントが使用できるよう、1 つの場所にディレクトリ定義を定めることができます。

このセクションでは、サービスを見つけるためのリモートディレクトリサーバーの使用方法を説明します。



リモートディレクトリサーバーは、Natural サブプログラムとして実装されます。

そのようなサブプログラムのサンプルが、SYSRPC ライブラリ内に提供されています。 このサブプログラムは RDSSCDIR と呼ばれ、必要なディレクトリ情報はワークファイルから読み取られます。 このサブプログラムのインターフェイスは、独自のリモートディレクトリサービスの開発を可能にするためにドキュメント化されています。 詳細については、「*RDS インターフェイスの作成*」を参照してください。

RDS インターフェイスは、Natural サブプログラムの Natural パラメータデータエリアであり、ディレクトリサービスルーチンは Natural サブプログラムのコードセクションです。 リモート CALLNAT がローカルサービスディレクトリ内で検出されない場合、RPC ランタイムは内部的なリモート CALLNAT を実行することでリモートディレクトリサーバーと通信します。

内部ディレクトリキャッシュは、リモートディレクトリへのアクセスを最小にします。 キャッシュ情報は、リモートディレクトリサーバーにより定義された有効時間で制御されます。

リモートディレクトリサーバーの使用

手順 12.1. リモートディレクトリサーバーを使用するには

1 ディレクトリファイルの作成

SYSRPC ユーティリティのリモートディレクトリメンテナンスサービスディレクトリメンテナンス機能を使用して、リモートディレクトリサービスのディレクトリファイルを作成します。 サブプログラム RDSSCDIR は SYSRPC ライブラリ内に提供されており、Natural ワークファイル（固定ブロック、レコード長 80 バイト）からディレクトリ情報を読み取ります。 これは、サーバー起動 JCL 内の適切なデータセットに割り当てられたワークファイル CMWKF01 です。

2 リモートディレクトリサーバーの起動

リモートディレクトリサーバーを起動し、次の手順へ進んでください。

3 **RDS** の定義

以下の 2 つのオプションがあります。

- プロファイルパラメータ RPC またはパラメータマクロ NTRPC のキーワードサブパラメータ RDS を指定します。
- または、SYSRPC ユーティリティのメンテナンス機能を使用して、リモートディレクトリサーバーを定義します。『*SYSRPC ユーティリティ*』ドキュメントの*リモートディレクトリメンテナンスサービスディレクトリメンテナンス*を参照してください。 リモートディレクトリサーバーの定義は、互換性の理由でサポートを継続しています。 ですが、使用する RDS は、プロファイルパラメータ RPC またはパラメータマクロ NTRPC のキーワードサブパラメータ RDS に定義するようにしてください。 このために、ディレクトリサーバー

の場所の定義を許可するエントリが提供されています。これにより、1つ以上のリモートディレクトリサーバー定義による既存のローカルディレクトリ情報の展開が可能となります。

次に示すのは、サービスディレクトリ NATCLTGS 内のリモートディレクトリサーバーの定義方法の例です。

サービスディレクトリ					
	NODE	**SERVER**	**LIBRARY**	**PROGRAM**	**LOGON**
1	NODE1				
2		SERVER1			
3			SYSTEM		
4				TESTS1	
5				TESTS2	
6	RDSNODE				
7		DIRSRV1			
8			#ACI		
9				RDSSCDIR	

この例では、ローカルに SERVER1 という名前のサーバーを定義します。 このサーバーは、サービス TEST1 および TEST2 を実行します。

さらに、リモートディレクトリサーバー DIRSRV1 に対する定義があります。 リモートディレクトリサーバーは、ライブラリ定義用にハッシュ（#）記号が先行することで識別されます。

NODE および SERVER の定義は、通常どおり Natural RPC 内で使用されます。 ライブラリ定義では、RDS との接続のために必ず使用されるトランスポートプロトコル（ACI）が定義されます。

最後に、PROGRAM エントリにはリモートディレクトリサービスを提供するリモートサブプログラムの名前が含まれます（この場合は、サンプルサブプログラム RDSSCDIR を参照します）。

RDS インターフェイスの作成

RDS インターフェイスは、Natural サブプログラムのパラメータデータエリア（PDA）です。

独自の RDS インターフェイスを作成するには、次に示すパラメータデータエリアを使用できます。

```
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
```

パラメータの説明については、次の表を参照してください。

パラメータ	フォーマット／長さ	説明
P_UDID	B8	一意のディレクトリ ID です。ディレクトリ情報の変更後に増加します。 クライアントは、この ID をキャッシュ内に保存します。 このバイナリ番号が 1 つのクライアント要求から次へと増える場合、リモートディレクトリ情報と対応しなくなるため、クライアントはそのローカルキャッシュ情報を削除します。
P_UDID_EXPIRATION	I4	これは、有効時間を秒単位で定義します。つまり、UDID 設定の確認のために RDS に接続することなく、ローカルキャッシュ情報をクライアントが使用できる秒数です。 全クライアントに対してディレクトリ修正がアクティブにされることを保証できた後の時間制限の定義を可能にします。この時間を不用意に低い値に設定すると、多数の RDS へのネットワークトラフィックを引き起こすことになります。
P_CURSOR	I4	リモートプロシージャコールには、以前のサーバーへの接続が設定できない場合に別のサーバーをスキャンするためのオプションがあります。（プロファイルパラメータ RPC またはパラメータマクロ NTRPC のキーワードサブパラメータ TRYALT を参照してください。 このパラメータは上部からの検索用にゼロを含み、スキャンを継続するためにレコード位置を記憶するよう、RDS により修正されることがあります。 値はクライアントにより評価されません。スキャンを継続するためにキャッシュからのみ挿入されます。
P_ENTRIES	I4	このパラメータは、P_REQUEST 内のサービス定義の数を含みます。

パラメータ	フォーマット／長さ	説明
P_REQUEST	A16/1:250	スキャン可能なサーバーアドレスに対するサービスのリストです。 エントリは次のように構成されます。 プログラム名（A8） ライブラリ名（A8）
P_EXTENT	A16/1:250	予約済み（将来的に使用される予定）です。
SRV_NODE	A8	サーバーノードを含みます。
SRV_NODE_EXT	A8	サーバーノード拡張を含みます。
SRV_NAME	A8	サーバー名を含みます。
SRV_NAME_EXT	A8	サーバー名拡張を含みます。

リモートディレクトリサービスルーチンの作成

リモートディレクトリサービスルーチンは、Naturalサブプログラムのコードエリアです（このコードエリアのデフォルトバージョンは SYSRPC ライブラリ内のプログラム RDSSCDIR です）。

手順 12.2. 独自の RDS ルーチンを作成するには

- 次に示す中間コードを修正します。

```
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
```

リモートディレクトリサービスプログラム RDSSCDIR

このプログラムは、SYSRPC ライブラリ内にあります。 ワークファイル（固定ブロック、レコード長 80 バイト）からディレクトリ情報を読み込みます。

独自のプログラムは、どこ（例：データベース）からでもディレクトリ情報を読み込むことができます。 配布バージョンの RDSSCDIR の場合、これはワークファイル CMWKF01 であり、サーバー起動 JCL 内の適切なデータセットに割り当てられます。

ディレクトリワークファイルの構造

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

UDID 定義

```
(UDID)
   binary number
```

UDID_EXPIRATION 定義

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

ノード定義

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

サーバー定義

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

ライブラリ定義

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

プログラム定義

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

namevalue

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

namevalue の後の *logon-option* は、次の定義行と同様にオプションです。 *logon-option* の有効な値については、『SYSRPC ユーティリティ』ドキュメントの「*サービスディレクトリメンテナンス*」を参照してください。

例：ワークファイルからディレクトリ情報を読み込む

```
(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
```

上記例では、ディレクトリは以下を含みます。

- ノード NODE1 上で実行している SERVER1 および SERVER2 の 2 つのサーバー。

  サーバー SERVER1 は、SYSTEM ライブラリのプログラム TESTS1、TESTS2、および TESTS3 を実行します。

  サーバー SERVER2 は、SYSTEM ライブラリのプログラム TESTS4 を実行します。
- SYSTEM ライブラリのプログラム TESTS1～TESTS4 を実行できる、ノード NODE2 上の 1 つのサーバー SERVER1。

上記例内の行のインデントは必須ではありません。すべての行は、任意の位置（1）で開始できます。 このファイルは手動で修正することができます。または、SYSRPC リモートディレクトリメンテナンス機能を使用して生成できます。

13 Security の使用

Natural Security での Natural RPC の使用

Natural RPC は、セキュリティがどちらか（またはどちらにも）アクティブなクライアント／サーバー環境の Natural Security もサポートします。

全般的な情報については、『*Natural Security*』ドキュメントを参照してください。

クライアント／サーバー環境での Natural リモートプロシージャコールの使用の制御方法については、『*Natural Security*』ドキュメントの「*Protecting Natural RPC Servers and Services*」を参照してください。

クライアント側

クライアントは、RPC 要求とともにログオンデータを送信する必要があります。 ログオンデータは、ユーザー ID、パスワード、およびライブラリで構成されています。

- ユーザー ID およびパスワードは、Natural RPC サーバー側でクライアントの認証を実行するために使用されます。
- ライブラリは、要求されたライブラリへの Natural Security によって保護されたログオンを実行するために使用されます。

次の情報は、Natural RPC クライアントのみに適用されます。 Natural Security によって保護された Natural RPC サーバーにアクセスする EntireX RPC クライアントについては、EntireX 開発者キットのドキュメントを参照してください。

ログオンデータ

ログオンデータを Natural RPC サーバーに送信するには、ログオンオプションを使用する必要があります。 「*Natural RPC 環境の運用*」の「*ログオンオプションの使用*」を参照してください。 ログオンデータの各部分は、次のように設定されます。

1. ユーザー ID およびパスワード：

   Natural Security 下でクライアントが実行される場合

   クライアントでの Natural Security ログオンのユーザー ID およびパスワードが使用され、Natural RPC サーバーへ渡されます。

   異なるユーザー ID またはパスワードをサーバー側での Natural Security ログオンに使用する場合は、アプリケーションプログラミングインターフェイス `USR1071N`（下記参照）を使用できます。

   注意: Natural Security ライブラリプロファイルのセッションパラメータ制限の Natural RPC 制限部分で、`USR1071N` の使用を禁止できます。

Natural Security 下でクライアントが実行されない場合

Natural RPC サーバーに渡すユーザー ID およびパスワードを指定するには、最初の RPC 要求が送信される前に、クライアントはアプリケーションプログラミングインターフェイス USR1071N（下記参照）を呼び出す必要があります。

2. ライブラリ：

デフォルトで、クライアントが現在ログオンしているライブラリの名前が使用されます。 別のライブラリ名を Natural RPC サーバーに渡す場合は、アプリケーションプログラミングインターフェイス USR4008N を使用できます。

USR1071N

アプリケーションプログラミングインターフェイス USR1071N は、ライブラリ SYSEXT にあります。 これは、Natural RPC サーバーに渡すユーザー ID およびパスワードを指定するために使用します。

手順 13.1. USR1071N を使用するには

1 サブプログラム USR1071N およびプログラム USR1071P を、ライブラリ SYSEXT からライブラリ SYSTEM または steplib ライブラリまたはサーバー環境の任意のアプリケーションにコピーします。

2 DEFINE DATA PARAMETER ステートメントを使用して、次のパラメータを指定します。

パラメータ	I/O	フォーマット	説明	
USERID	I	A08	使用するユーザー ID。	
PASSWORD	I	A08	ユーザーIDを検証するパスワード。このパスワードは、クライアント側では評価されません。	
MIXEDCASE	I	A01	パスワードの大文字小文字混在オプション（オプション）。	
			Y	大文字と小文字が混在するパスワードを許可します。
			N	パスワードを大文字に変換します。

3 クライアント側の呼び出し元プログラムで、次のステートメントを指定します。

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

コマンド行から USR1071P を呼び出し、表示されるウィンドウにユーザー ID およびパスワードを入力することもできます。

詳細については、ライブラリ SYSEXT の USR1071T メンバを参照してください。

注意: USR1071N を呼び出す 2 つのサンプルが提供されています。USR1071P は、ユーザー ID とパスワードを渡します。USR1071X（拡張バージョン）は、さらにユーザーが各種データを設定／取得できるようにします。

サーバー側

サーバー側に Natural Security がインストールされており、AUTO=ON が指定されていない場合は、ユーザー ID とパスワードを使用した Natural ログオンが必要です。 Natural プロファイルパラメータ STACK を使用して、Natural システムコマンド LOGON を渡すことをお勧めします。AUTO=ON が指定されている場合は、通常どおり *INIT-USER の内容が内部ログオンに使用されます。

ログオンオプションを実施する（ログオンオプションが設定されたクライアントからの要求だけをサーバーに受信させる場合）には、そのサーバーに対してプロファイルパラメータ RPC またはパラメータマクロ NTRPC のキーワードサブパラメータ LOGONRQ を ON に設定します。 ログオンオプションが実施されない場合は、ログオンデータのないクライアント要求は受け入れられ、サーバーライブラリまたはいずれかの steplib で実行されます。 これにより、パブリックかつ保護されたサービスを提供できます。

クライアントがログオンデータを渡す場合は、クライアントからのユーザー ID およびパスワードがサーバー上の対応するユーザーセキュリティプロファイルに対して確認されます。要求したライブラリへログオンし、サーバー上の対応する Natural Security ライブラリおよびユーザープロファイル定義に従ってサブプログラムが実行されます。

サブプログラムの実行後、サーバー上で CALLNAT 要求前に使用したライブラリが再度更新されます。 会話型 RPC の場合は、会話内の最初の CALLNAT 要求はサーバー上のライブラリ ID を設定し、CLOSE CONVERSATION ステートメントはサーバー上のライブラリ ID を会話が開かれる前に使用されたものにリセットします。

Natural Security のライブラリプロファイルの *Natural RPC 制限*の一部として、サーバーセッションオプション Close all databases が提供されています。 これにより、ライブラリに含まれるリモートサブプログラムによって開かれたすべてのデータベースは、ライブラリへ（から）の Natural ログオン（ログオフ）が実行されたときに閉じられます。これは、各クライアントがそれ自身のデータベースセッションを使用することを意味しています。

Close all databases オプションが設定された場合、サーバーによってこのクライアントについて実行されるすべての Adabas アクセスに対して、クライアント固有の ETID を使用することもできます。 この場合、ETID=OFF で Natural RPC サーバーを起動し、ETID を必要とする各クライアントのユーザープロファイルに適切な ETID を定義する必要があります。例えば、ETID *USER を指定します。 この場合、同じ名前の 2 つのクライアントは、Adabas コールによって 2 つの同時要求を発行できないことに注意してください。

パスワードの変更

Natural RPC サービス要求によって、Natural RPC サーバーの Natural Security パスワードを変更できます。 この目的のために、アプリケーションプログラミングインターフェイス USR2074N がライブラリ SYSEXT にあります。

手順 13.2. USR2074N を使用するには

1 サブプログラム USR2074N およびオプションでプログラム USR2074P を、ライブラリ SYSEXT からライブラリ SYSTEM または steplib ライブラリまたはサーバー環境の任意のアプリケーションにコピーします。

2 DEFINE DATA PARAMETER ステートメントを使用して、次のパラメータを指定します。

パラメータ	I/O	フォーマット	説明	
USERID	I	A08	使用するユーザー ID。	
PASSWORD	I	A08	ユーザー ID を検証するパスワード。 このパスワードは、クライアント側では評価されません。	
NEWPASSWORD	I	A08	ユーザー ID の新しいパスワード。 このパスワードは、クライアント側では評価されません。	
NODE-NAME	I	A192	アドレスされるサーバーノードの名前。 物理ノード名の場合は最大 32 文字、論理ノード名の場合は最大 192 文字をノード名に含めることができます。 「*Natural RPC 環境の運用*」の「*EntireX ロケーショントランスペアレンシの使用*」を参照してください。 ライブラリ SYSEXT で提供されるサンプル USR2074P は、最大 32 文字をサポートします。	
SERVER-NAME	I	A192	アドレスされるサーバーの名前。 物理サーバー名の場合は最大 32 文字、論理サービス名の場合は最大 192 文字をサーバー名に含めることができます。 「*Natural RPC 環境の運用*」の「*EntireX ロケーショントランスペアレンシの使用*」を参照してください。 ライブラリ SYSEXT で提供されるサンプル USR2074P は、最大 32 文字をサポートします。	
PROTOCOL	I	A1	サーバーノードをアドレスするトランスポートプロトコル。 有効値：	
			B	EntireX Broker
RC	O	I2	戻り値：	
			0	OK。MESSAGE には確認メッセージが含まれます。
			1	RPC またはサーバーノードからのエラー。MESSAGE にはエラーメッセージが含まれます。

パラメータ	I/O	フォーマット	説明	
			2	インターフェイスからのエラー。MESSAGE にはエラーメッセージが含まれます。
			3	Natural Security エラー。MESSAGE# には Natural エラー番号が含まれ、MESSAGE には対応するメッセージテキストが含まれます。
MESSAGE#	O	N4	返されたメッセージ番号。	
MESSAGE	O	A80	返されたメッセージテキスト。	

3 クライアント側の呼び出し元プログラムで、次のステートメントを指定します。

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

または、ライブラリ SYSEXT からプログラム USR2074P を使用できます。 コマンド行から USR2074P を呼び出し、表示されるウィンドウに必要なデータを入力します。 この場合、パスワードを除くすべての入力は大文字に変換されます。 パスワードに大文字と小文字を混在させかどうかを選択できます。

偽装（z/OS バッチモードのみ）

- 偽装の目的
- 偽装を有効にする手順（サーバー側）
- ユーザー偽装を使用する手順（クライアント側）
- 偽装のルール

偽装の目的

偽装は、Natural RPC サーバー側のオプションの機能で、Natural RPC サーバーが Natural Security で実行されている場合にのみ使用できます。 偽装機能は、*Natural RPC サーバー用の Natural Security プロファイル*で制御されます。 『*Natural Security*』ドキュメントの「*Protecting Natural RPC Servers and Services*」も参照してください。

z/OS バッチモードでの偽装は、z/OS 環境下の Natural RPC サーバーフロントエンドの使用を必要とし、z/OS によって提供される SAF インターフェイスを使用します。

Natural RPC サーバーに対して偽装がアクティブな場合、*ログオンオプション*を使用するクライアント要求は、LOGON データ内でクライアントが渡すユーザー ID（Natural RPC ユーザー ID と呼ばれる）で実行されたオペレーティングシステム側からのものになります。 偽装の前提は、Natural RPC サーバーが実行されているオペレーティングシステムへのアクセスが、SAF 対応の外部セキュリティシステムによって制御されていることです。 ユーザー認証（Natural RPC ユー

ザー ID およびパスワードの検証）は、この外部セキュリティシステムによって実行されます。認証が成功した後、オペレーティングシステムに対してユーザーの ID が確立されます。つまり、ACEE が作成され、現在のクライアント要求が実行される TCB にリンクされます。 後続の認証チェックは、この ID に基づいて実行されます。つまり、SAF 対応の外部セキュリティシステムによって制御されるリソースへのすべてのアクセスは、この ID に対して認可されます。このことは、ワークファイルおよびデータベースへのアクセスに特に適用されます。

偽装によって、Natural Security がオフにされることはありません。外部セキュリティシステムによるユーザーの ID の認証が成功した後、同じ LOGON データを使用して、Natural Security ログオンが行われます。ただし、パスワードの検証は行われません。

偽装を使用して Natural RPC サーバーを起動するには、「*Natural RPC サーバーの起動*」の「*RPC サーバーフロントエンドを使用した Natural RPC サーバーの起動*」を参照してください。

注意: 偽装を使用しない場合、ログオンオプションを使用するクライアント要求は、Natural RPC サーバーを起動したユーザー ID で実行されたオペレーティングシステム側からのものになります。

偽装を有効にする手順（サーバー側）

1. **RPC** サーバーフロントエンドのインストール

   Natural for Mainframes の『インストール』ドキュメントの説明に従います。「*z/OS 環境での Natural のインストール手順*」を参照してください。

   推奨される APF 認可 LINKLIST ライブラリを使用する場合は、結果として生じるロードモジュールが STEPLIB または JOBLIB 連結内に存在しないことを確認する必要があります。

2. **Natural z/OS** バッチニュークリアスの **DB2** インターフェイス DSNRLI へのリンク

   この手順は、Natural for DB2 のユーザーのみに適用されます。

3. **Adabas** バッチリンクルーチン ADALNK のリエントラントバージョン ADALNKR の使用

   「*Natural RPC サーバーの起動*」の「*レプリカ付きメインフレーム Natural RPC サーバーの考慮事項*」を参照してください。

4. **EntireX Broker** スタブ BKIMBTSO の使用

   「*Natural RPC 環境の設定*」の「*EntireX Broker スタブへのアクセスの提供*」を参照してください。

5. 必要な **RPC** サーバー固有のすべての **Natural** プロファイルパラメータの定義

   「*Natural RPC 環境の設定*」の「*RPC サーバー固有の Natural パラメータの設定*」を参照してください。パラメータは、NATPARM パラメータモジュールまたは CMPRMIN データセット内に定義されます。 JCL EXEC ステートメントのパラメータ PARM= は、Natural プロファイルパラメータを指定するために使用されません。

6. **Natural Security** での **RPC** サーバープロファイルの定義

   Natural Security（NSC）で、RPC サーバー（SRVNAME）が使用するサーバー名に RPC サーバープロファイルを定義して、偽装を有効にします。

   『*Natural Security*』ドキュメントの「*Protecting Natural RPC Servers and Services*」にある、*Natural RPC サーバー用の Security プロファイルに関する説明*を参照してください。

7. **SAF** 定義の確認

   （この手順は、Natural for DB2 のユーザーのみに適用されます。）

   SAF リソースクラス DSNR がアクティブな場合は、次の SAF 定義が必要かどうかを確認する必要があります。

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

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

   subsys は DB2 サブシステム ID です。

   DB2 にアクセスする各ユーザーは、グループ *DB2group* のメンバである必要があります。

   詳細については、関連する IBM の DB2 ドキュメントを参照してください。

8. ユーザー出口 NATRPC02 の作成

   （この手順は、Natural for DB2 のユーザーのみに適用されます。）

   NATPLAN のコールを使用して Natural RPC ユーザー出口 NATRPC02 を作成し、必要な DB2 プランを設定します。

   現在の Natural for DB2 バージョンの NATPLAN を使用してください。

   サンプル NATRPC02：

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

9. ロールサーバーの起動

   Natural RPC サーバーによって使用される *subsystem-id* に対してロールサーバーを起動します。

10. **Natural RPC** サーバーフロントエンドの起動

    Natural RPC サーバーフロントエンドを起動します。

「*Natural RPC サーバーの起動*」の「*RPC サーバーフロントエンドを使用した Natural RPC サーバーの起動*」を参照してください。

必要なすべてのロードライブラリを STEPLIB 連結に追加したことを確認してください。 特に次のものが必要です。

- Natural ロードライブラリ
- EntireX ロードライブラリ
- Adabas ロードライブラリ（Adabas リンクルーチン ADAUSER を使用する場合）
- DB2 ロードライブラリ（DB2 にアクセスする場合）

次のメッセージが表示された場合、偽装は正常にアクティブになります。

- ジョブログ：

```
RPC0010 偽装用の認可環境が確立されました
```

- RPC トレースファイル：

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

ユーザー偽装を使用する手順（クライアント側）

クライアントは、RPC 要求と一緒にログオンデータを送信する必要があります。標準の Natural Security（NSC）によって保護された Natural RPC サーバーの場合、これはすでに行われています。 標準の Natural RPC サーバーと異なり、ユーザー ID は有効な SAF ユーザー ID であり、パスワードは対応する SAF パスワードである必要があります。 ユーザー ID およびパスワードは、サーバーが実行されている z/OS システム上の Natural RPC サーバーによってチェックされます。 z/OS 環境下での偽装の場合、ユーザー ID は、定義されているルールに従って NSC によってチェックされます。 パスワードは無視されます。 したがって、NSC パスワードを SAF パスワードに設定する必要はありません。

クライアントの種類によっては、ログオンデータの設定が異なります。

Natural クライアント

1. サービスディレクトリメンテナンス関数またはプロファイルパラメータ RPC またはパラメータマクロ NTRPC の DFS キーワードサブパラメータのいずれかでログオンオプションをオンにします。

   また、USR2007N を使用してオンにすることもできます。

   「*Natural RPC 環境の運用*」の「*ログオンオプションの使用*」を参照してください。
2. SAF ユーザー ID および SAF パスワードを設定します。アプリケーションプログラミングインターフェイス USR1071P を使用します。

クライアントが Natural Security（NSC）環境で実行されており、NSC のユーザー ID およびパスワードが SAF ユーザー ID および SAF パスワードと同じ場合、USR1071P は必要ありません。

EntireX RPC クライアント

1. アプリケーション環境に応じて、Natural ログオンオプションをオンにします。
2. アプリケーション環境に応じて、RPC ユーザー ID および RPC パスワードを SAF ユーザー ID および SAF パスワードに設定します。

偽装のルール

- 偽装は、会話型以外の各 CALLNAT の開始時および各会話の開始時に実行されます。
- Natural RPC ユーザー ID およびパスワードの認証は、外部セキュリティシステムによって実行されます。 FSEC システムファイル上のパスワードは使用されません。
- 認証が成功した後、Natural RPC ユーザー ID はオペレーティングシステムに対して確立されます（ユーザーは偽装されます）。
- 偽装が適切に実行された後：
  1. Natural RPC ユーザー ID に対して Natural Security ログオンが実行されます。パスワードチェックは行われません。
  2. CM で開始されない DDNAME を持つすべてのワークファイルが、Natural RPC ユーザー ID を使用してオープンされます。
  3. すべての Adabas データベースが、Natural RPC ユーザー ID でオープンされます（Adabas 外部セキュリティのみ該当）。
  4. ETID が NSC ユーザープロファイルで指定されている場合、この ETID が Adabas オープン要求で使用されます。
  5. DB2 接続が、Natural RPC ユーザー ID でオープンされます（Natural for DB2 ユーザーのみ該当）。
- 各非会話型 CALLNAT の終了時および各会話の終了時に、Natural RPC ユーザー ID はオペレーティングシステムからログオフされます。
- ログオフ後：
  1. CM で開始されない DDNAME を持つすべてのワークファイルがクローズされます。
  2. すべての Adabas データベースがクローズされます。

偽装（CICS）

以下では次のトピックについて説明します。

偽装の目的

偽装は、Natural RPC サーバー側のオプションの機能で、Natural RPC サーバーが Natural Security で実行されている場合にのみ使用できます。 偽装機能は、*Natural RPC サーバー用の Natural Security プロファイル*で制御されます。『Natural Security』ドキュメントの「*Protecting Natural RPC Servers and Services*」も参照してください。

CICS 環境での偽装は、CICS 環境下で Natural RPC サーバーフロントエンドの使用を必要とし、CICS によって提供されるインターフェイスを使用します。

Natural RPC サーバーに対して偽装がアクティブな場合、ログオンオプションを使用するクライアント要求は、`LOGON` データ内でクライアントが渡すユーザー ID（Natural RPC ユーザー ID と呼ばれる）で実行された CICS 側からのものになります。 CICS 環境での偽装は、CICS オプションを使用して、指定されたユーザー ID で CICS タスクを開始します。 クライアント要求が到着した後、Natural RPC サーバーフロントエンドは、`EXEC CICS START TRANSID()` コマンドの `USERID()` オプションを使用して、新しい CICS タスクを開始します。`USERID` は Natural RPC ユーザー ID です。 ユーザー認証（Natural RPC ユーザー ID の検証）は、一般的には基礎となる外部セキュリティシステムを使用して CICS によって実行されます。 認証が成功した後、CICS タスクに対してユーザーの ID が確立されます。 後続の認証チェックは、この ID に基づいて実行されます。 つまり、CICS によって制御されるリソースへのすべてのアクセスは、この ID に対して認可されます。このことは、CICS リソースおよびデータベースへのアクセスに特に適用されます。

偽装によって、Natural Security がオフにされることはありません。 CICS によるユーザーの ID の認証が成功した後、パスワードの検証と同じ `LOGON` データを使用して Natural Security ログオンが行われます。

偽装を使用して Natural RPC サーバーを起動するには、「**Natural RPC サーバーの起動**」の「**RPC サーバーフロントエンドを使用した Natural RPC サーバーの起動**」（CICS のみ）を参照してください。

注意: 偽装を使用しない場合、ログオンオプションを使用するクライアント要求は、Natural RPC サーバーを起動したユーザー ID で実行されたオペレーティングシステム側からのものになります。

偽装を有効にする手順（サーバー側）

1. **CICS** 環境での **RPC** サーバーフロントエンドのインストール

   Natural for Mainframes の『*インストール*』ドキュメントの「*Natural CICS インターフェイスのインストール*」セクションの該当する手順の説明に従います。

2. **Adabas** 外部セキュリティの **Adabas** リンクルーチンのインストール

   詳細については、関連する Adabas ドキュメントを参照してください（Adabas 外部セキュリティユーザーのみに適用されます）。

3. `NATETB23` ではなく **Broker** スタブ `CICSETB` を使用

   「***Natural RPC 環境の設定***」の「*メインフレームでの **EntireX Broker** スタブへのアクセスの提供*」を参照してください。

   `CICSETB` を Natural CICS インターフェイスニュークリアスにリンクする必要があります。

4. 必要な **RPC** サーバー固有のすべての **Natural** プロファイルパラメータの定義

   「***Natural RPC 環境の設定***」の「***RPC サーバー固有の Natural パラメータの設定***」を参照してください。

   パラメータは、パラメータモジュール `NATPARM` またはデータセット `CMPRMIN` 内に定義されます。

5. **Natural Security** での **RPC** サーバープロファイルの定義

   Natural Security（NSC）で、RPC サーバー（`SRVNAME`）が使用するサーバー名に RPC サーバープロファイルを定義して、偽装を有効にします。

   『*Natural Security*』ドキュメントの「*Protecting Natural RPC Servers and Services*」にある、*Natural RPC サーバー用の Security プロファイルに関する説明*を参照してください。

6. **CICS** 起動パラメータ `XUSER=YES` の場合

   CICS 起動パラメータ `XUSER=YES` が指定されている場合、クライアントユーザーごとにサロゲートユーザーを定義する必要があります。

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

   ここでは次の内容を表しています。

   userid1 は、クライアントのユーザー ID です。

   userid2 は、Natural RPC サーバーフロントエンドが起動されるユーザー ID です。

   詳細については、関連する IBM の CICS ドキュメントを参照してください。

7. **RPC** サーバーフロントエンドに対する **CICS** PROGRAM エントリの定義

   「*Natural CICS インターフェイスのインストール*」の対応する手順を参照してください。

8. RPC サーバーフロントエンドを呼び出すトランザクション ID に対して CICS TRANSACTION エントリを定義します。

   「*Natural CICS インターフェイスのインストール*」の対応する手順を参照してください。

9. DB2TRAN および DB2ENTRY エントリの定義

   （この手順は、Natural for DB2 のユーザーのみに適用されます。）

   RPC サーバーフロントエンドを呼び出すトランザクション ID に対して DB2TRAN および DB2ENTRY エントリを定義します。

10. ロールサーバーの起動

    Natural RPC サーバーによって使用されるサブシステムに対して*ロールサーバー*を起動します。

    （この手順は、NCMDIR マクロパラメータ ROLLSRV が YES に設定されている場合にのみ適用されます。）

11. **CICS** 環境での **Natural RPC** サーバーフロントエンドの起動

    「***Natural RPC サーバーの起動***」の「***RPC サーバーフロントエンドを使用した Natural RPC サーバーの起動（CICS のみ）***」を参照してください。

    RPC トレースファイルに次のメッセージが表示された場合、偽装は正常にアクティブになります。

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

ユーザー偽装を使用する手順（クライアント側）

クライアントは、RPC 要求と一緒にログオンデータを送信する必要があります。標準の Natural Security（NSC）によって保護された Natural RPC サーバーの場合、これはすでに行われています。 ユーザー ID およびパスワードは、定義されているルールに従って NSC によってチェックされます。 標準の Natural RPC サーバーと異なり、ユーザー ID は有効な CICS ユーザー ID である必要もあります。

クライアントの種類によっては、ログオンデータの設定が異なります。

Natural クライアント

1. ログオンオプションをオンにする

   サービスディレクトリメンテナンス関数またはプロファイルパラメータ RPC またはパラメータマクロ NTRPC の DFS キーワードサブパラメータのいずれかでログオンオプションをオンにします。

   また、アプリケーションプログラミングインターフェイス USR2007N を使用してオンにすることもできます。

   「*Natural RPC 環境の運用*」の「*ログオンオプションの使用*」を参照してください。
2. ユーザー **ID** およびパスワードの設定

   ユーザーIDおよびパスワードを設定します。アプリケーションプログラミングインターフェイス USR1071P を使用します。

   クライアントが Natural Security（NSC）環境で実行されており、NSC のユーザー ID およびパスワードがサーバー側のユーザー ID およびパスワードと同じ場合、USR1071P は必要ありません。

EntireX RPC クライアント

1. **Natural** ログオンオプションをオンにする

   アプリケーション環境に応じて、Natural ログオンオプションをオンにします。
2. **RPC** ユーザー **ID** およびパスワードの設定

   アプリケーション環境に応じて、RPC ユーザー ID および RPC パスワードを設定します。

偽装のルール

- 偽装は、会話型以外の各 CALLNAT の開始時および各会話の開始時に実行されます。
- Natural RPC ユーザー ID の認証は、CICS によって実行されます。 パスワードは使用されません。
- 認証が成功した後、Natural RPC ユーザー ID は CICS に対して確立されます（ユーザーは偽装されます）。
- 偽装が適切に実行された後：
  1. Natural RPC ユーザー ID に対して Natural Security ログオンが実行されます。定義されているルールに従ってパスワードチェックが行われます。
  2. すべての CICS リソースは、Natural RPC ユーザー ID を使用してアクセスされます。
  3. すべての Adabas データベースが、Natural RPC ユーザー ID でオープンされます（Adabas 外部セキュリティのみ該当）。

4. ETID が NSC ユーザープロファイルで指定されている場合、この ETID が Adabas オープン要求で使用されます。
5. DB2 接続が、Natural RPC ユーザー ID でオープンされます（Natural for DB2 ユーザーのみ該当）。

- 各非会話型 CALLNAT の終了時および各会話の終了時に、Natural RPC ユーザー ID は CICS からログオフされます。
- ログオフ後：

1. すべての CICS リソースがクローズされます。
2. すべての Adabas データベースがクローズされます。
3. DB2 への接続がクローズされます（Natural for DB2 ユーザーのみ該当）。

EntireX Security での Natural RPC の使用

Natural RPC では、クライアント側およびサーバー側の EntireX Security が完全にサポートされています。

クライアント側での EntireX Security

EntireX Broker へログオン、EntireX Broker からログオフするために、Natural アプリケーションプログラミングインターフェイス USR2071N が提供されています。 EntireX Broker へログオンするには、USR2071N のログオン機能を使用してユーザー ID とパスワードを、選択した EntireX Broker に渡します。 ログオンが成功した後に、返されるセキュリティトークンは Natural によって保存され、後続の個々の呼び出しで EntireX Broker に渡されます。 ログオン機能は Natural アプリケーションで透過的です。

EntireX Security がインストールされているか、または EntireX Broker 属性ファイルに AUTOLOGON=NO が指定されている場合、最初のリモート CALLNAT 実行の前にログオン機能で USR2071N を呼び出す必要があります。

リモート CALLNAT を使用する必要がなくなった場合にはすぐに、ログオフ機能で USR2071N を呼び出すことをお勧めします。

手順 13.3. USR2071N を使用するには

1 サブプログラム USR2071N を、ライブラリ SYSEXT からライブラリ SYSTEM または steplib ライブラリまたはサーバー環境の任意のアプリケーションにコピーします。

2 DEFINE DATA PARAMETER ステートメントを使用して、次のパラメータを指定します。

パラメータ	I/O	フォーマット	説明	
function	I	A08	ファンクションコード。設定可能値は次のとおりです。	
			LOGON	EntireX Broker へのログオン
			LOGOFF	EntireX Broker からのログオフ
broker-id	I	A192	Broker ID *broker-id* は、物理ノード名には 32 文字まで、論理ノード名または論理サービス名については 192 文字までを指定できます。「*Natural RPC 環境の運用*」の「***EntireX** ロケーショントランスペアレンシの使用*」を参照してください。 **注意:** 互換性の理由で、*broker-id* のための A8 フィールドを渡す既存の呼び出し元をサポートするために、*broker-id* は BY VALUE RESULT で定義されます。 ライブラリ SYSEXT で提供されるサンプル USR2071P では、最大 32 文字がサポートされています。	
user-id	I	A08	ユーザー ID。	
password	I	A08	ユーザー ID のパスワード。	
newpassw	I	A08	ユーザー ID の新しいパスワード。	
rc	O	N04	戻り値：	
			0	OK
			1	無効なファンクションコード
			9999	EntireX Broker エラー（*message* を参照）
message	O	A80	EntireX Broker によって返されたメッセージテキスト。	

3 クライアント側の呼び出し元プログラムで、次のステートメントを指定します。

```
CALLNAT 'USR2071N' function broker-id user-id password newpassword rc message
```

CALLNAT ステートメントの「*構文の説明*」も参照してください。

コマンド行から USR2071P を呼び出し、表示されるウィンドウにユーザー ID およびパスワードを入力することもできます。 この場合、パスワードを除くすべての入力は大文字に変換されます。 パスワードに大文字と小文字を混在させかどうかを選択できます。

ロケーショントランスペアレンシ使用時の特別な考慮事項：

論理ノード名を使って LOGON する場合、LOGBROKER キーワードを使用する必要があります。

```
BROKER-ID := 'LOGBROKER=my_logical_node,my_set'
```

論理サービス名を使って LOGON する場合、LOGSERVICE キーワードを使用する必要があります。

```
BROKER-ID := 'LOGSERVICE=my_logical_service,my_set'
```

機能：

LOGON	EntireX Broker LOGON 機能は、渡された *user-id* と *password* で *broker-id* へ実行されます。LOGON コールが成功した後、クライアントは、EntireX Broker *broker-id* と通常どおりに通信できます。 *newpassw* で、クライアントユーザーは EntireX Security 機能を通して自身のパスワードを変更できます。 注意： ■ ログオンが正常に実行された場合、この LOGON に使用されたユーザー ID は、この EntireX Broker 経由で送られる後続のすべてのリモートプロシージャ CALLNAT で指定された EntireX Broker に渡されます。 LOGON が明示されない場合、システム変数 *USER の現在の内容が使用されます。 EntireX Broker 1 に LOGON を発行した場合には同じことが適用されますが、リモートプロシージャ CALLNAT は EntireX Broker 2 経由でルーティングされます。 ■ 複数の EntireX Broker へ同時にログオンすることが可能です。 LOGON ごとに異なるユーザー ID を使用できます。 ■ EntireX Broker への LOGON のためのユーザー ID は、クライアントアプリケーションが実行される Natural ユーザー ID と異なっていてもかまいません。 ■ 元の LOGON がパスワードなしで行われた場合、EntireX Broker タイムアウトが発生した後、内部的なログオンが再度行われます。LOGON に使用されたパスワードは保存されません。 タイムアウトが発生した後、内部的な再ログオンが可能ではない場合、クライアントは明示的に LOGON を再発行する必要があります。 ■ Natural セッション終了時、ログオンが実行されたすべての EntireX Broker に対して暗黙的な LOGOFF が実行されます。
LOGOFF	EntireX Broker LOGOFF 機能は、指定された *broker-id* に実行されます。

サーバー側での EntireX Security

プロファイルパラメータ RPC またはパラメータマクロ NTRPC のキーワードサブパラメータ ACIVERS の値が 2 以上の場合、サーバーは、セッションが開始したとき、LOGON 機能を使用して EntireX Broker にログオンします。 ユーザー ID は SRVUSER によって定義されたユーザー ID と同じです。

EntireX Security がインストールされており、EntireX 信頼済みユーザーの ID 機能が有効ではない場合、必要なパスワードを指定するための代替策が 2 つあります。

- SRVUSER=*NSC の設定
- アプリケーションプログラミングインターフェイス USR2072N の使用

これらの代替策について、次に説明します。

SRVUSER=*NSC の設定

Natural Security がサーバーにインストールされている場合は、プロファイルパラメータ RPC またはパラメータマクロ NTRPC のキーワードサブパラメータ SRVUSER を *NSC に設定して、サーバーの起動時に使用された現在の Natural Security ユーザー ID が Natural Security パスワードとともに LOGON で使用されることを指定できます。 この場合には、ACIVERS の値は最低でも 4 に設定する必要があります。

パスワードを指定するためのアプリケーションプログラミングインターフェイス USR2072N の使用

アプリケーションプログラミングインターフェイス USR2072N を使用すると、プロファイルパラメータ RPC またはパラメータマクロ NTRPC のキーワードサブパラメータ SRVUSER とともに LOGON で使用されるパスワードを指定できます。

手順 13.4. USR2072N を使用するには

1 サブプログラム USR2072N およびオプションでプログラム USR2072P を、ライブラリ SYSEXT からライブラリ SYSTEM または steplib ライブラリまたはサーバー環境の任意のアプリケーションにコピーします。

2 DEFINE DATA PARAMETER ステートメントを使用して、次のパラメータを指定します。

パラメータ	I/O	フォーマット	説明
password	I	A08	ユーザー ID のパスワード。

3 クライアント側の呼び出し元プログラムで、次のステートメントを指定します。

```
CALLNAT 'USR2072' password
```

CALLNAT ステートメントの「*構文の説明*」も参照してください。

4 Natural RPC サーバーがその初期化を開始する前に、呼び出し元プログラムを実行する必要があります。 このことを行うには、サーバーを起動するときに、呼び出し元プログラムの名前を Natural スタックに挿入します。 このために、ライブラリ SYSEXT からプログラム USR2072P を使用することもできます。 この場合、パスワードはデフォルトで大文字に変換されます。 大文字小文字混在オプション Y を 2 番目のパラメータとして渡すことによって、大文字小文字混在でパスワードを入力するオプションがあります。

```
STACK=(LOGON server-library;USR2072P password [Y])
```

Integrated Authentication Framework の使用

Integrated Authentication Framework（IAF）は、Natural RPC サーバー側で使用できるオプション機能です。

Integrated Authentication Framework の目的

Integrated Authentication Framework は、次の条件下で使用できます。

1. Natural RPC サーバーが Natural Security 環境で実行されています。
2. EntireX Broker が IAF サーバーによって保護されています。
3. Natural RPC サーバーと EntireX Broker が同じ IAF サーバーを使用しています。
4. Software AG Security eXtension（SSX）が EntireX によってインストールされている必要があります。
5. Natural RPC サーバーが TSO、z/OS バッチモード、UNIX、または Windows 環境で実行されています。

IAF 機能は、Natural RPC サーバー用の Natural Security プロファイルで制御されます。『*Natural Security*』ドキュメントの「*rotecting Natural RPC Servers and Services*」を参照してください。

Natural RPC サーバーが Integrated Authentication Framework を使用するように設定されると、Natural RPC サーバーは、ログオンデータで渡されるユーザー ID およびパスワードによってクライアント要求を認証しなくなります。 代わりに、クライアントが EntireX Broker にログオン

するときに使用したユーザー ID が、信頼済みユーザーの ID として認証なしで使用されます。次の手順が実行されます。

1. クライアント要求が、EntireX Broker によって IAF サーバーを使用して認証されます。
2. IAF サーバーは、ユーザー ID を含む暗号化および署名されたトークンを返します。
3. EntireX Broker は、IAF トークンを RPC 要求とともに Natural RPC サーバーに渡します。
4. Natural RPC サーバーは、IAF トークンをチェックおよび復号化し、IAF トークンに含まれるユーザー ID を信頼済みとして扱います。
5. Natural Security は、定義された権限ルールを使用して、ユーザー ID をチェックし、要求されたライブラリへのログオンを実行します。 パスワードは使用されません。

この結果、Natural Security ログオンが成功した後は、Natural システム変数 *USER 内の Natural ユーザー ID と EntireX ユーザー ID は同一です。

Integrated Authentication Framework を使用する手順（クライアント側）

クライアントは、標準の EntireX Security 環境で行われるのと同様に、EntireX Broker にログオンする必要があります。 ユーザー ID およびパスワードが IAF サーバーによって認証されることは、クライアントに透過的です。

クライアントは、標準の Natural Security によって保護された Natural RPC サーバーの場合に行われるように、RPC 要求と一緒にログオンデータを送信する必要もあります。

標準の Natural Security によって保護された Natural RPC サーバーとは異なり、ログオンデータ内のユーザー ID およびパスワードは無視され、認証は行われません（上記参照）。 Natural ライブラリのみが Natural RPC サーバーによって評価されます。 したがって、ユーザー ID およびパスワードはログオンデータ内で省略される場合があります。

Integrated Authentication Framework を有効にする手順（サーバー側）

サーバー側で Integrated Authentication Framework を有効にするには、環境に応じて、次の手順を実行します。

- TSO および z/OS バッチモード

- Windows および UNIX

TSO および z/OS バッチモード

1. Natural Security で IAF サービスを定義します。『*Natural Security*』ドキュメントの「*Protecting Natural RPC Servers and Services*」にある「*IAF Support*」を参照してください。
2. IAF サーバーによって使用される CA 証明書を *IAF サポート*のトラストストアフィールドで参照される RACF キーリングに追加します。
3. Natural RPC サーバーを起動する予定の RACF ユーザー ID に、キーリングへのアクセスを許可します。
4. Natural Security で、RPC サーバー（SRVNAME）が使用するサーバー名に RPC サーバープロファイルを定義して、IAF サポートを有効にします。『Natural Security』ドキュメントの「*Protecting Natural RPC Servers and Services*」にある、Natural RPC サーバー用の Security プロファイルに関する説明を参照してください。
5. LE サポート（LE370=YES）で Natural RPC サーバーによって使用される Natural z/OS バッチニュークリアスを生成します。
6. Natural RPC サーバーが使用する Natural プロファイルパラメータで、ACIVERS=9 以上に設定します。
7. CEEOPTS 入力データセットを使用して Natural RPC サーバーを実行するバッチ JCL の POSIX をオンにします。「*z/OS でのバッチサーバーの起動*」を参照してください。
8. Software AG Security eXtension（SSX）ロードライブラリを JCL に追加します。「*z/OS でのバッチサーバーの起動*」を参照してください。

Windows および UNIX

1. Natural Security で IAF サービスを定義します。『*Natural Security*』ドキュメントの「*Protecting Natural RPC Servers and Services*」にある「*IAF Support*」を参照してください。
2. Natural Security で、RPC サーバー（SRVNAME）が使用するサーバー名に RPC サーバープロファイルを定義して、IAF サポートを有効にします。『Natural Security』ドキュメントの「*Protecting Natural RPC Servers and Services*」にある、*Natural RPC サーバー用の Security プロファイルに関する説明*を参照してください。
3. Natural RPC サーバーが使用する Natural プロファイルパラメータで、ACIVERS=9 以上に設定します。

4. 環境変数を修正します。

UNIX：	$SAG/iaf/v*XX*/lib/ ディレクトリを $LD_LIBRARY_PATH（HP-UX システムでは $SHLIB_PATH）環境変数に追加します。 *XX* は現在のバージョン番号です。
Windows：	%ProgramFiles%\Software AG\EntireX\Bin ディレクトリを %PATH% 環境変数に追加します。

14 EntireX Broker サポート

セキュリティ

Natural RPC クライアントおよび Natural RPC サーバーでは EntireX Security がサポートされます。 このことは、ローカルオペレーティングシステムに対する認証および Software AG の Integrated Authentication Framework（IAF）によって実行される認証に当てはまります。 EntireX Broker 属性 AUTHENTICATION-TYPE を参照してください。

Natural RPC クライアントまたは Natural RPC サーバーが ACIVERS=8 で起動された場合は、スタブ出口なしの EntireX Security（メインフレームのみ）および SECUEXIT なしの EntireX Security（すべてのプラットフォーム）がサポートされます。 この場合、Natural は ACI フィールド KERNELSECURITY の設定を取得するのに、KERNELVERS コールを発行してからその他の Broker コールを発行します。 この KERNELSECURITY 設定は、その後のすべての Broker コールに渡されます。 また、この機能により、Natural RPC クライアントは同じ Natural セッションで、セキュリティ保護された EntireX Broker にもセキュリティ保護されていない EntireX Broker にもアクセスできます。

ロギングとアカウンティング

Natural RPC クライアントおよび Natural RPC サーバーは、EntireX Broker 内の RPC プログラムおよび RPC ライブラリのロギングおよびアカウンティングをサポートします。

Natural RPC クライアントでは、EntireX Broker に対して、実行するサブプログラムの名前と、そのサブプログラムの実行に使用されるライブラリの名前を指定します。

Natural RPC サーバーは、実行されているサブプログラムの名前と、そのサブプログラムの実行に実際に使用されているライブラリの名前を返します。

索引