4D Server

はじめに

概要

4D Server はマルチユーザ、クロスプラットフォームデータに対応する4D用のアプリケーションサーバです。

4D Serverではクライアント/サーバアーキテクチャに基づいたマルチユーザデータベースおよびカスタムアプリケーションを作成し、使用することができま す。プラットフォームに依存しないクライアント/サーバアーキテクチャにより、WindowsおよびMacintosh 上の4Dからデータースをシームレスに管理できます。4D Server には大規模システムに向けて、プロフェッショナル仕様の開発用ツール、拡張性、データ保護機能、コネクティビティオプションが用意されています。

4D Server のアーキテクチャは完全に統合されており、クライアントとサーバの両方で1つの4Dプリケーションを使用します。4D Server により、開発者はフロントエンドアプリケーションとバックエンドアプリケーションを個別に設計する手間から解放されます。それだけではありません。4D Server は "管理不要" のサーバです。インストールや使用、管理が容易であり、非常にコスト効率の高いサーバです。

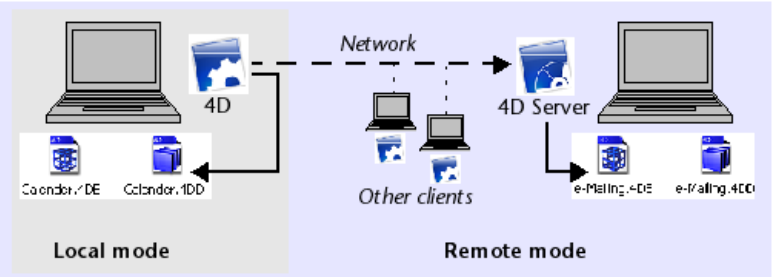
ローエンドのファイル共有ベースのシステムと複雑なSQL ベースのRDBMS との中間に位置するもの、それが4D Server です。4D Server アプリケーションは既存の大規模データベース (Oracle、Sybase、ODBC 対応サーバ等) とスムーズに連携させることができます。4D Server は、あらゆる規模の企業のワークグループから派生するニーズに対応します。

統合されたバックエンドアーキテクチャとフロントエンドアーキテクチャ

4D Server では、フロントエンドアプリケーションとバックエンドアプリケーションは同じものです。クライアントソフトウェアとサーバアプリケーションは、4Dという1 つの製品が2つの役割を果たしています。4D Server アプリケーション自体は、4D Server と4Dリモートモードという2 つの部分に分かれており、それぞれがクライアント/サーバアーキテクチャにおける構成要素となります。

4D Server 部分は、サーバマシンに常駐し、サーバ上のデータベースを保存し、管理します。エンドユーザは自分自身が使用しているマシン (クライアント) からこのデータベースを利用します。

各クライアントマシンには、4Dアプリケーションが常駐し、ローカルモードまたはリモートモードで使用します。ローカルモードでは、データベースまたはユーザのマシンのローカルに保存されている4D アプリケーションと連動します。リモートモードでは、ユーザはサーバ上のデータベースにアクセスし、データの追加、レポートの作成、データベースデザインの変更等のデータベース作業を行います。4D のローカルモードで行う作業はすべて、4D Server と4D リモートモードを使用して行うことができます。



クライアント/サーバ環境で操作するために、他のミドルウェアや開発は不要です。4Dと4D Server では、同一のインタフェースツール、言語、情報管理システムが使用されます。

1 つのプラットフォーム (Windows あるいはMacintosh) 用に作成されたローカルシングルユーザ用アプリケーションはすべて、ワークグループ対応のクライアント/サーバソリューションへと簡単に拡張できます。また逆に、4D Server で作成されたアプリケーションを自動的にローカルモードモードのスタンドアロンアプリケーションに変えることも可能です。

"管理不要" のデータサーバとアプリケーションサーバ

4D Server は、ユーザに重点を置いた4D の特長を受け継ぎました。この結果、4D Server は完全なプラグアンドプレイシステム（PnP）となっています。

4D Server の管理ウインドウ

- **わかりやすくグラフィカルなオンライン一元管理情報**：4D Server の管理ウインドウには、重要な情報が自動的に表示されます。4D Server に割り当てられているメモリ合計、データキャッシュ、接続中のユーザ数と名前、プロセス数とプロセスのステータス、バックアップとリクエストのモニターリ ング、すべてのサーバの稼働状況等の情報です。

- **システム自動構成とスケーラビリティ**: 4D Server は、システムの再構成や再設計を行わなくても、新しいプロトコル、クライアント、プラグインを追加できるように設計されています。

- **クライアントワークステーションに対するダイナミックな自動更新とバージョン管理**：データベースの変更や、プラグインまたはコンポーネントの追加、削除、修正があった場合には、接続されているすべての4D マシンがダイナミックに自動更新されます。更に、カスタムクライアントサーバアプリケーションを構築した場合は、4D Server アプリケーションが更新される際、実行可能な4Dクライアントアプリケーションの新しいバージョンを自動的にダウンロードすることが可能です。

- **標準のTCP/IP プロトコルを使用した非同期自動接続**: 4D Server と4D は、クライアントおよびサーバのプラットフォームに関わらず、TCP/IP プロトコルを使用して透過的に通信を行います。あらゆるオペレーティングシステムにおいて、TCP/IP プロトコルが導入されるようになったため、4D Server でこのプロトコルを使用する際にその他のインストール作業は必要ありません。

- **4D SQL と HTTP 接続の同時セッションおよび状況管理**: 4D Server は、テーブル / プロセス / ユーザの組み合わせごとに現在の作業環境を自動的に作成し、管理します。このセッションベースのアーキテクチャにより、ユーザごとの各プロセスでは、デー タを個別かつ同時に操作することができます。設定を行えば、4D Server の SQL サーバは自動的に内部および外部のSQL クエリを処理します。4D Server のHTTP サーバは、HTTP リクエストおよびSOAP リクエストに応答します。

- **自動レコードロック機能**: 4D Server では、レコードのロックや解放が自動的に行われ、"使用中" のレコードの変更に関わるトラブルを回避することができます。また、レコードロック機能により、ページロックやファイルロックに関連する問題も解消されます。

- **統合ユーザインタフェースメッセージシステム**: デスクトップの領域から誕生した4D Server は、最新の統合開発環境で提供されているユーザインタフェースをすべて備えています。例えば、接続解除やバックアップ等をスケジュール管理するなど、管理側の作業をクライアントに通知することができます。

- **自動起動、自動終了メソッド**: 4D Server では、特定のイベントに応じて5 つのサーバデータベースメソッドが自動的に起動されます。この5 つのメソッドとは、**On Server Startup**、**On Server Shutdown**、**On Server Open Connection**、**On Server Close Connection**、**On Web Connection** です。例えば**On Server Startupデータベースメソッド**はセッション全般で必要となるオブジェクトを自動的に初期化し、ロードします。

4D Server独自の機能

4D Server には、4D の機能に加えて次の機能があります。

- **マルチユーザデータ管理**: 複数のユーザがデータベース作業を同時に実行できます。複数のユーザが同じファイルあるいは異なるファイルのレコードを追加、変更、削除、検索、ソート、印刷することが可能です。データの整合性は、内蔵のレコードロックシステムによって保持されます。

- **マルチユーザ開発**: 複数のユーザがデータベースの開発や設計を同時に行えます。例えば、同時に複数ユーザがファイル定義の編集、およびレイアウト、スクリプト、メソッドの作 成、変更することができます。データベースデザインの整合性は、内蔵のオブジェクトロックシステムによって保持されます。

- **プラットフォームに依存しないクライアント / サーバアーキテクチャ**: このアーキテクチャにより、Macintosh および Windows 上の4Dクライアントに対するデータベースパフォーマンスはシームレスに管理されます。また、異なるプラットフォーム上での同時マルチ開発や、異種混合 ハードウェア環境で動作する4Dクライアントにより入力、変更されるすべてのデータのインタフェースも管理されます。

- **Windows および Mac OS ベースの4D プラグインアーキテクチャ**: Windows 版およびMac OS 4D Server では、サーバマシンにWindows ベースおよびMacintosh ベースの4D プラグインをインストールできます。このアーキテクチャにより、プラットフォームに依存しない4D プラグインの配布を簡単に行えます。クライアントのプラットフォームが何であれ、4D Server および4D によりプラグインはスムーズに処理されます。

- **ビルトイン HTTP サーバ**: 4D ローカルモードと同様に、4D Server および 4D リモートモードにはHTTP エンジンが

組み込まれており、Web 上に4D データベースを公開することができます。データベースはWeb 上に直接公開されるので、データベースシステム、Web サイト、この両者の間のCGI インタフェースを開発する必要はありません。データベースがWeb サイトなのです。また、あらゆるリモート4D マシンをWebサーバへ変換することもできます。組み込みの4D Web サーバおよび4D の詳細については、4D Language ReferenceマニュアルのWebサーバ概要の項を参照してください。

- **SSLによる接続保護** (Secured Socket Layer)：4D Server の接続を暗号化することができます。実際、"クラシック"なクライアント/サーバアーキテクチャはSSLの暗号化サービスを使用できます。詳細は**クライアント/サーバ接続の暗号化**を参照してください。

- **トリガ**: トリガはテーブルに結び付けられたメソッドです。開発者はトリガを呼び出しません。トリガはレコードが操作 (追加、削除、更新) されるたびに、自動で4Dデータベースエンジンが起動します。4D Serverでは、トリガはサーバマシン上で実行されます。4DであろうとODBCで接続するアプリケーションであろうと、すべてのクライアントはトリガによりデータベースルールが強制されます。4Dのトリガに関する詳細は、4D Language Referenceマニュアルのを参照してください。

- **ストアドプロシージャ**: 4Dメソッドを作成して、それがサーバマシン上あるいは複数の指定したクライアント上の独立したプロセスで実行されるよう設定できます。クライアント/ サーバの業界用語でいえばこの機能はストアドプロシージャと呼ばれます。にもかかわらず、4D Serverはストアドプロシージャのコンセプトの先を行くアーキテクチャを提供しています。4D Serverにおいて、ストアドプロシージャはカスタムサーバプロセス (またはクライアントプロセス、後述参照) であり、サーバあるいはクライアントの他の実行中のプロセスとは非同期かつ独立してコードを実行します。通常のクライアント/サーバアーキテクチャでは、 ストアドプロシージャは実行して結果を返します (同期的にあるいは非同期的に)。4D Serverでは、クライアント/サーバセッション中を通して実行されるストアドプロシージャを開始し、クライアントから送信されるメッセージに応答させ ることができます。同時にクライアントと協働しないストアドプロシージャを走らせて、4Dコネクティビティプラグインや ODBCを使用してSQLベースの サーバや他の4D Serverとデータの同期を行うこともできます。同時に起動できるストアドプロシージャの数に制限はありません (ハードウェアとメモリの制限を除く)。4D Serverのストアドプロシージャは独自のプロセス内で実行され、他のプロセスと同様個別のデータベースコンテキスト (カレントセレクションなど) を保守できます。さらに4Dランゲージは、ストアドプロシージャのプロセス変数 (BLOB変数を含む) をクライアントプロセスから読み書きするコマンドを提供します。これによりクライアントとストアドプロシージャ間の洗練された通信が可能になります。実際 ストアドプロシージャを使用して、新しいカスタムサービスを4D Serverに追加できます。詳細は**ストアドプロシージャ**を参照してください。

- **クライアント上で実行されるストアドプロシージャ**: 4D Serverでは、クライアントやサーバから、1つあるいは複数の他のクライアント上でストアドプロシージャを実行できます。これによりワークロードをクライアントやサーバ間で分散したり、クライアント間の通信を行うようなアプリケーションをビルドできます。詳細は**ストアドプロシージャ**を参照してください。

- **サーバパス**: ユーザのパスワードと一緒に、サーバデータベースのパスを保存できます。この機能を使用して、ユーザは.4DLinkドキュメントをダブルクリックしてサーバ上のデータベースに接続できます。詳細は**4D Serverデータベースへの接続**を参照してください。

- **サービスとして登録**: Windowsでは4D Serverをサービスとして起動できます。

- **組み込みのバックアップシステム**: 4D Serverは完全なデータベースバックアップおよび復元モジュールを持っています。このモジュールを使用して、アプリケーションを終了しなくても、動作 中にデータベースをバックアップできます。バックアップは手動あるいは特定の間隔で自動で起動できます。問題が発生した場合、データベースの復元と再起動 も自動で開始できます。

- **物理ミラーによるバックアップ**: 重要なアプリケーションでは、物理ミラーによるバックアップを設定できます。これにより動作中のデータベースに問題が発生した場合でも即座に再起動が可能です。

- **コネクティビティプラグイン**: 4D ODBC Proのような4Dコネクティビティプラグインを使用すれば、4D Serverや4Dから直接メインフレームやORACLEなど他のODBCデータソースに接続できます。これらのデータベースとインタラクティブに情報を 共有できます。さらに4Dは4D Server ODBC Driverを提供していて、ODBCクライアントから4D Serverに接続することができます。

4D Serverアーキテクチャ

クライアント/サーバアーキテクチャを使用して、4D Serverはデータベースを格納したり保存したりするだけでなく、クライアントへのサービスも提供します。これらのサービスはリクエスト/レスポンスのシステムを使用してネットーワーク経由で管理されます。

例えばレコードを検索するために、クライアントマシンはクエリリクエストをサーバに送信します。リクエストを受信するとサーバはクエリ処理をサーバマシン上でローカルに実行し、クエリが完了すると、結果(検索されたレコード) を返します。

4D Serverのアーキテクチャはクライアント/サーバモデルに基づきます。現在ではクライアント/サーバアーキテクチャはより古いファイル共有アーキテクチャをしのぎ、マルチユーザデータベースにおいて最も効率の良いモデルとなりました。

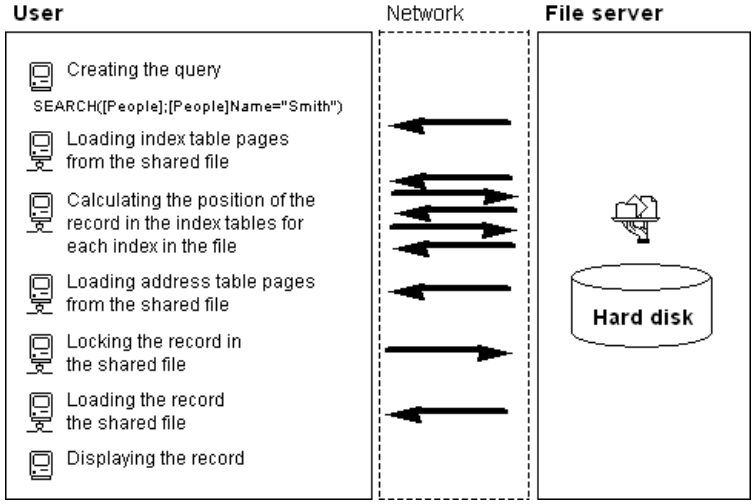
4D Serverのクライアント/サーバ実装はミニコンピュータの世界で使用されているものと同じです。しかし、4D Serverは2つの重大な新しい手法を提供します:

- データベースのすべてのレベルにおけるユーザフレンドリなグラフィカルインタフェース
- 効率と速度を向上させる統合アーキテクチャ

ファイル共有アーキテクチャ

クライアント/サーバアーキテクチャが導入される以前、マルチユーザシステムには、ネットワークアーキテクチャのファイル共有モデルが使用されていました。このモデルでは、すべてのユーザは同一のデータを共有しますが、データ管理は中央のデータベースエンジンによって制御されていません。各クライアントマシンにはデータベースストラクチャとエンジンのコピーを格納する必要があり、一方でサーバにはネットワーク上でファイルを共有するために必要なソフトウェアしかありません。

ファイル共有モデルのもとでは、各ワークステーションがすべてのデータ変更をローカル上で行います。処理ごとに多量の更新を必要とし、ネットワーク付加の増大につながりました。以下の図は、名前が"Smith"である人をデータベースで検索するときにネットワーク上で作成されるトラフィックを例示したものです。



ファイル共有モデルのもう1 つの欠点は、メモリキャッシュを利用してメモリ上にレコードを保持できないということです。ファイル共有モデルでレコードがメモリ内に保持されると、各ユーザが同じレコードを異なるバージョンでキャッシュに格納する可能性があり、データに矛盾が生じてしまうためです。したがって、ユーザはレコードにアクセスするたび、ファイルサーバからレコードをダウンロードする必要があります。これはネットワーク負荷を増大させ、レコードアクセス時間の増加につながります。

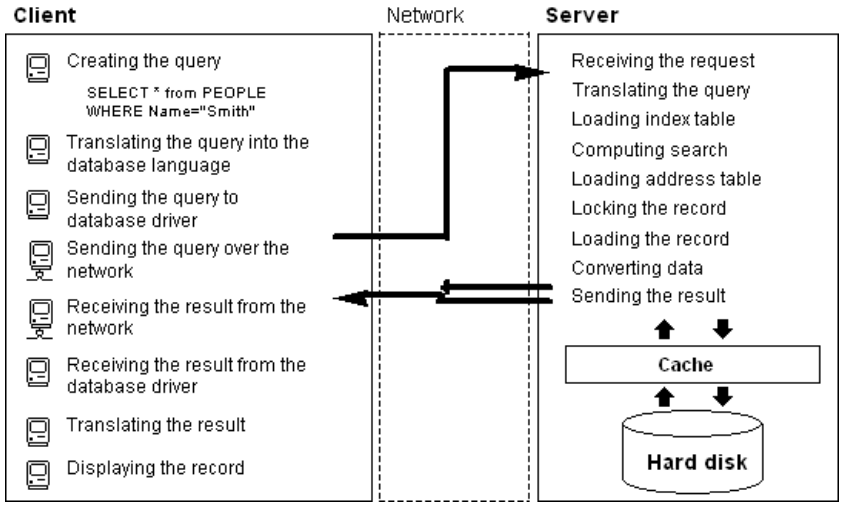
異種混合クライアント/サーバアーキテクチャ

クライアント/サーバアーキテクチャは効率が良く高速なので、ミニコンピュータの世界では大規模データベースシステムで広範囲に使用されています。このアーキテクチャでは、パフォーマンスを向上させるために、サーバマシンとクライアントマシンが作業を分担します。

サーバには中心となるデータベースエンジンがあり、データを格納し、管理します。データベースエンジンは、ディスク上に格納されたデータにアクセスする唯一のソフトウェアです。クライアントがサーバに要求を送ると、サーバは結果を返します。結果はクライアントが変更する特定のレコードであったり、ソートした一連のレコードの場合もあります。

一般に、ほとんどのクライアント/サーバアーキテクチャは、異種混合アーキテクチャと呼ばれていますが、これはクライアントマシン上で実行しているフロントエンドアプリケーションとサーバマシン上で実行しているデータベースエンジンに別々の製品が使用されるためです。このような場合には、クライアントとサーバの間に入って翻訳を行うデータベースドライバが必要です。

例えば、レコードを検索する場合、クライアントはサーバに検索要求を送ります。データベースはサーバ上に格納されているので、サーバはサーバマシン上でローカルにコマンドを実行し、結果をクライアントに返します。次の図は名前が“Smith”であるすべての人をデータベースから検索し、見つかった最初のレコードを表示するようにサーバに要求した場合にネットワーク上で作成されるトラフィックを示しています。



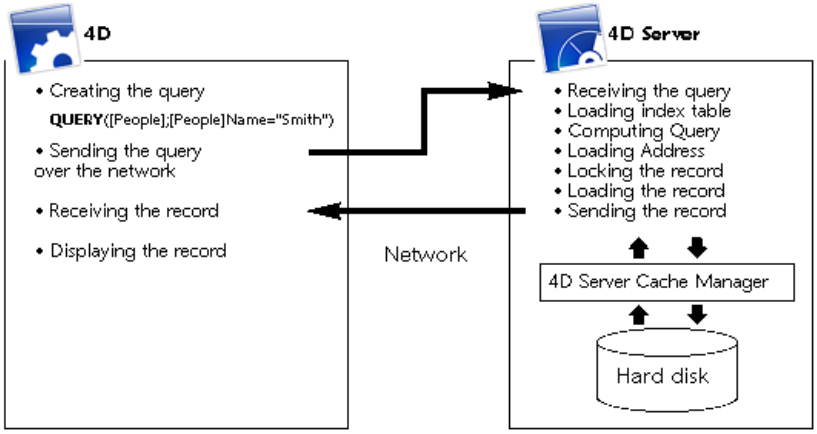
この例により、クライアント/サーバアーキテクチャとファイル共有アーキテクチャでは2 つの点で大きく異なることがわかります:

- **クライアント/サーバアーキテクチャではキャッシュを使用できる**: データに物理的なアクセスを行うのはエンジンだけのため、サーバはディスクに書き込まれるまで、変更レコードを保持するためのキャッシュをメモリ上に持つことができます。データは1 カ所から送出されるので、クライアントは必ず最新版レコードを受け取ることができます。中央のキャッシュメカニズムを使用することにより、データの整合性を保証すると共に、ディスクにアクセスするのではなく、メモリにアクセスするため、データベース処理速度の向上にもつながります。ファイル共有モデルでは、アクセスはすべてディスクアクセスです。
- **低レベルのデータベース処理はサーバ上で実行される**: クライアント/サーバアーキテクチャでは、インデックスやアドレステーブルのブラウジング等、低レベルのデータベース処理はサーバマシンのスピードに合わせてサーバマシン上でローカルに実行されるので、処理速度が大幅に速くなります。ファイル共有アーキテクチャでは、同じ処理を行っても、ネットワークの通信速度とクライアントマシンの制約のため遅くなります。

4D Server の統合クライアント/サーバアーキテクチャ

ほとんどのクライアント/サーバアーキテクチャでは、クライアントソフトウェアとサーバソフトウェアは2 つの異なる製品であり、互いに“通信”するためにはコミニュケーションレイヤが必要です。4D Server では、クライアント/サーバアーキテクチャは完全に統合されています。4D Server と4Dは同一の構造を共有し、直接通信を行うアプリケーションです。

4D Server と4Dは同じ言語を使用するので、問い合わせ言語を翻訳する必要がありません。クライアントとサーバの作業の分担は、透過的であり、4D Server が自動的に管理します。



作業の分担は、1 つの要求が1 つの応答を返す形で構成されています。図に示すように、クライアントには次の役割があります:

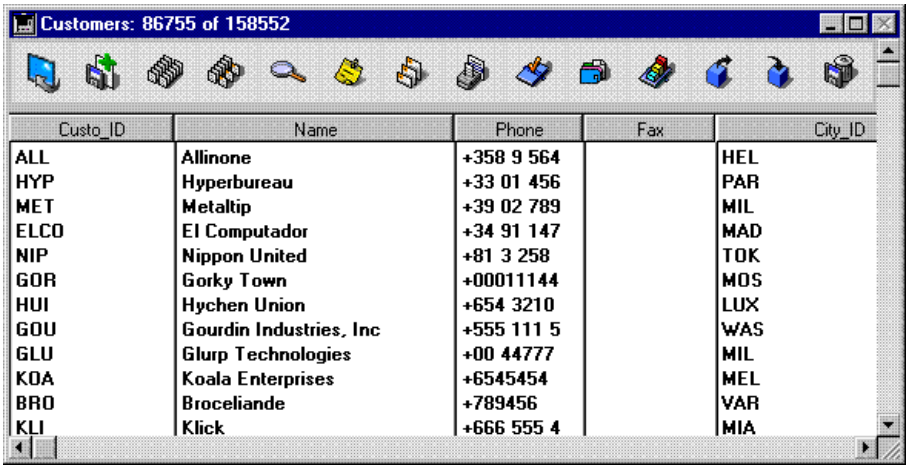
- **リクエスト**: 4Dクライアントマシンは4D Server にリクエストを送ります。これらのリクエストはクエリエディタや並べ替えエディタ等の組み込みエディタ、統合4Dランゲージ、あるいはSQLを使用して行います。4Dには、メソッドを作成し、変更できるエディタが用意されています。また変数や配列等のメソッド要素も管理されます。
- **レスポンスの受信**: 4Dクライアントマシンは4D Server からの応答を受け取り、ユーザインタフェース（フォームに様々なレコードが表示する等）を介してユーザに情報を示します。例えば名前が"Smith"であるすべてのレコードをクライアントが要求した場合に、4Dは4D Server から結果のレコードを受け取り、それをフォームに表示します。

サーバには次の役割があります:

- **スケジューリング**: 4D Server は、同時接続およびクライアントが作成した同時プロセスをすべてスケジュールするためのマルチタスキングアーキテクチャを使用します。
- **ストラクチャとデータオブジェクト**: 4D Serverはフィールド、レコード、フォーム、メソッド、メニュー、リスト等、すべてのデータオブジェクトとストラクチャオブジェクトを格納し、管理します。
- **キャッシュ**: 4D Server は、レコードの他にセレクションやセット等の特定のクライアントに固有のデータオブジェクトを納めるキャッシュを維持します。
- **低レベルデータベース処理**: 4D Server は、クエリやソート等、インデックステーブルやアドレステーブルを使用する低レベルのデータベース処理を実行します。

この作業の分担は、4D Server と4Dが独自の形式で統合されているので、非常に効率良く行われます。4D Server のアーキテクチャの統合は、すべてのレベルに及んでいます:

- **リクエストレベル**: 4Dが4D Server にクエリやソート等のリクエストを送信するとき、4Dは4D Server と同じ内部構造を使用してクエリ処理やソート処理の記述を送信します。
- **ストラクチャまたはデータレベル**: 4Dと4D Serverがデータやストラクチャのオブジェクトをやり取りする場合、どちらのアプリケーションも同じ内部形式を使用します。例えば4Dでレコードが必要な場合、4D Serverはディスクやメモリキャッシュに入っていた形のままでデータを送信します。同様に、4Dがレコードを更新する場合には、4Dが4D Server へデータを送信し、4D Server は受信したそのままのデータをキャッシュに格納します。
- **ユーザインタフェースレベル**: 4Dがレコードのリストを表示する場合、レコードを表示するために使用されるフォームは、クライアント/サーバアーキテクチャの役割を果たしています。例えば、次の図は[Customers]テーブルを検索した結果を示しています。



上記のウインドウは、一度に5フィールドずつ12レコードしか表示できないため、4D Server は12レコードだけを送信します。レコード全体を送る代わりに、4D Server はウィンドウに表示できるだけの数のレコードとフィールドを送ります。ユーザがフォームをスクロールした場合には、必要に応じて4D Server から残りのレコードやフィールドが送信されます。この最適化により、レコードやフィールドは必要な場合にだけ送られ、ネットワーク使用量が削減されます。

10分間で知る4D Server

インストールのチェック

10分間で知る4D Serverは、以下の作業を行うための方法の概略を説明するチュートリアルです:

- サーバデータベースを作成する
- 作成したサーバデータベースにクライアントを接続する
- テーブル、フィールド、フォーム、メニュー、メソッド等のデータベースストラクチャを作成する
- 2番目のユーザを接続し、同時に作業をする

少なくとも2台のコンピュータが必要です:

- 4D Server、4D、およびWebブラウザがインストールされているマシン
- 4Dがインストールされている他のマシン

4D Serverやローカルモードの4Dでの作業を初めて行う場合は、その前にインストール環境を確認することをお勧めします。確認するためには、この節を参照してください。

インストールされた要素

この段落では、4D + 4D Serverの標準インストールを行った後、ディスク上にインストールされている要素の場所を示します。

Windows

要素は**Program Files¥4D¥4D vXX** フォルダにインストールされ、**スタート**メニューに表示されます。

- **4D Server**: このフォルダには4D Serverアプリケーションの他、関連するファイルやフォルダが配置されます。4D Serverを起動するには、**4D Server.exe** ファイルをダブルクリックします。
- **4D**: このフォルダには4Dアプリケーションの他、関連するファイルやフォルダが配置されます。4Dを起動するには、**4D.exe** ファイルをダブルクリックします。

Mac OS

要素は**アプリケーション:4D:4D vXX** フォルダにインストールされます。

- **4D Server**: 4D Serverソフトウェアパッケージ。4D Serverを起動するには、このパッケージをダブルクリックします。
- **4D**: 4Dソフトウェアパッケージ。4Dを起動するには、このパッケージをダブルクリックします。

この演習を行うには、もう一台のマシンにも同様に4Dをインストールします。

次は?

ネットワーク越しに通信を行うマシンで、TCP/IPプロトコルが設定されていなければなりません。

4D Serverと4Dが正しくインストールされていれば、**サーバデータベースの作成**に進んでください。 そうでない場合、上記で示したファイルが失われているような場合は4D Product Line インストールガイドを参照して、それらのファイルのインストールを行ってください。

サーバデータベースの作成

この節では、ネットワーク上のリモートモードの4Dを使用してアクセス可能なサーバデータベースを作成する方法を説明します。最初に4D Serverや4Dを使用する前に、インストールを確認することをお勧めします。**インストールのチェック**インストールのチェックを参照してください。

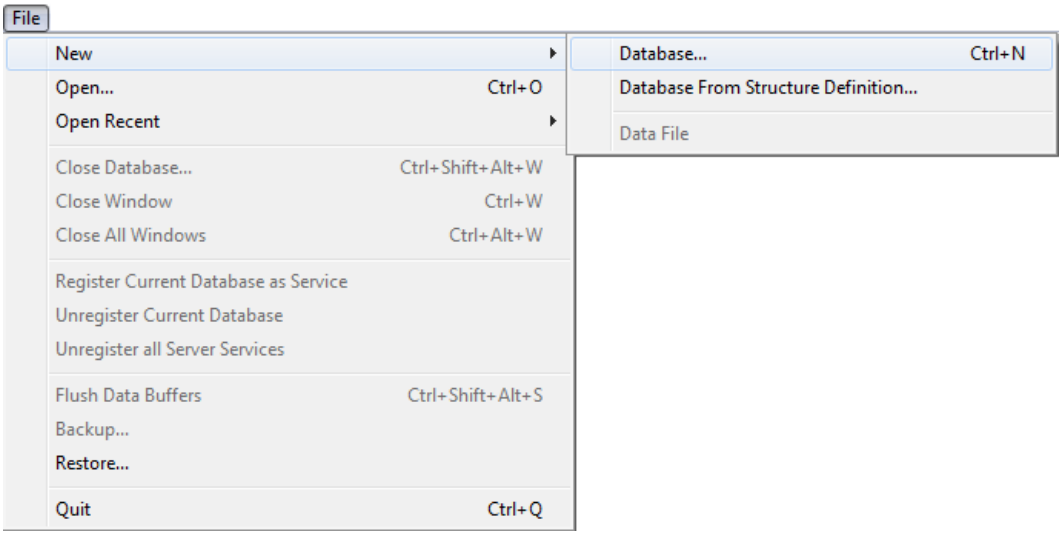
Note: この例題では、インストールガイドで説明されている、アクティベーション済みの4D Serverライセンスをお持ちであると仮定しています。リモートモードで4Dを使用するのにクライアントマシン側でライセンスは必要ありません。ライセ ンスは4D Serverマシン上で管理されます。詳細はインストールガイドを参照してください。

サーバデータベースを作成あるいは開くには、4D Serverを起動します。

1. 4D Serverアイコンをダブルクリックして、4D Serverを起動する

アプリケーションを最初に起動したときは、アプリケーションアクティベーションダイアログが表示されます。その後はブランクのウィンドウが起動のた びに表示されます。この動作はアプリケーションの環境設定で変更できます。このチュートリアルでは新規データベースを作成します。

2. 4D Serverのファイルメニューから、新規>データベース... コマンドを選択する

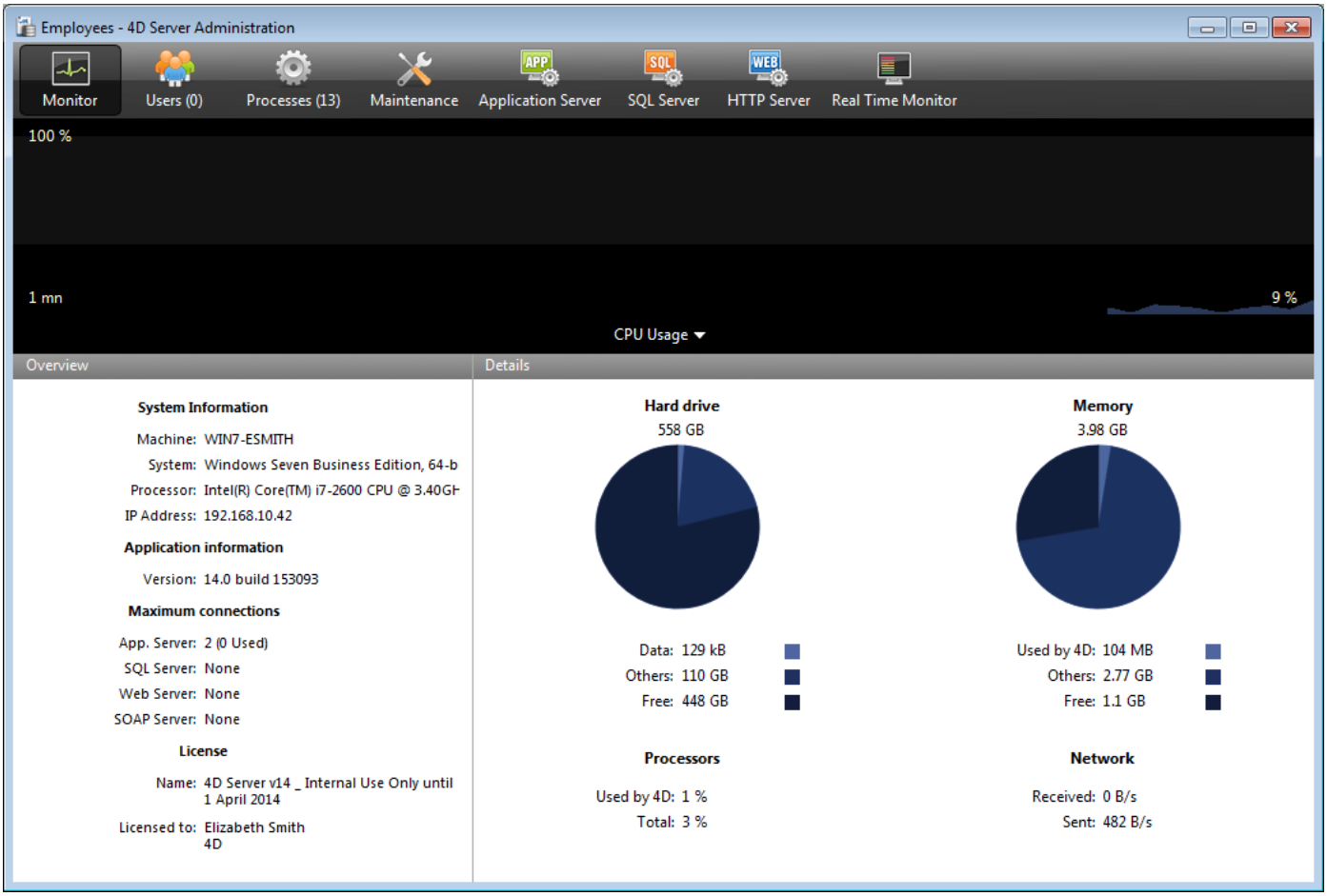


標準のファイルを保存ダイアログボックスが表示され、作成する新規データベースの名前と場所を指定できます。

3. 場所を指定し、データベースの名前を入力する

Employeesとタイプして**保存**をクリックします。

4D Serverは自動でデータベースの動作に必要なファイルやフォルダを作成します。そして管理ウィンドウが表示されます:



4D Serverの**管理ウィンドウ**には複数のページがあり、タブからアクセスできます。**モニタ**ページにはデータベースの動作のほか、システムと4D Serverアプリケーションに関する情報が動的に表示されます。

ユーザと**プロセス**ページには、データベースに接続中のユーザ数と、現在実行中のプロセス数が表示されます。今の時点で 接続ユーザ数は0です。つまりクライアントはだれもデータベースに接続していません。しかしながら、いくつかのプロセスは実行中です。これらのプロセスはデータ ベースエンジン (カーネル) そして4D Serverの組み込みサーバ (アプリケーションサーバ、HTTPサーバ、SQLサーバ) が自動的に作成したプロセスです。

次は?

この時点で、WindowsやMacintosh上の4Dからネットワーク経由でデータベースにアクセスできます。しかしデータベースはHTTP接続の準備はできていません。これらの接続はデフォルトでは許可されていません。

リモート4Dからサーバデータベースに接続するに進んでください。このチュートリアルではまずリモートモードの4Dを使用して接続し、データベースのストラクチャを定義、そしていくつかのレコードをデータベースに追加します。

リモート4Dからサーバデータベースに接続する

この節では以下について説明します:

- 作成したサーバデータベースにリモートの4Dから接続する方法。
- データベースストラクチャの作成。この節にはデータベースにテーブルやフィールドを作成、新規レコードを登録し、既存のレコードを更新するチュートリアルが含まれています。
- 2人目のユーザの接続。
- 2つのリモートクライアントからの同時作業。

データベースへの接続

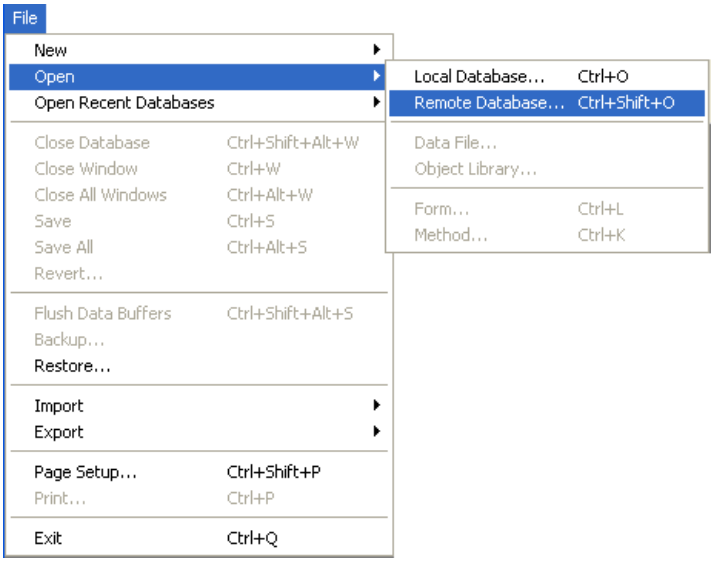
4D Serverでデータベースを作成した (**サーバデータベースの作成**参照)　とても、データベースデザインや実際のデータの更新はクライアントマシンから行われます。この節ではサーバに接続し、サーバデータベースを開く方法を学びます。

1. リモートの4Dアプリケーションアイコンをダブルクリックする

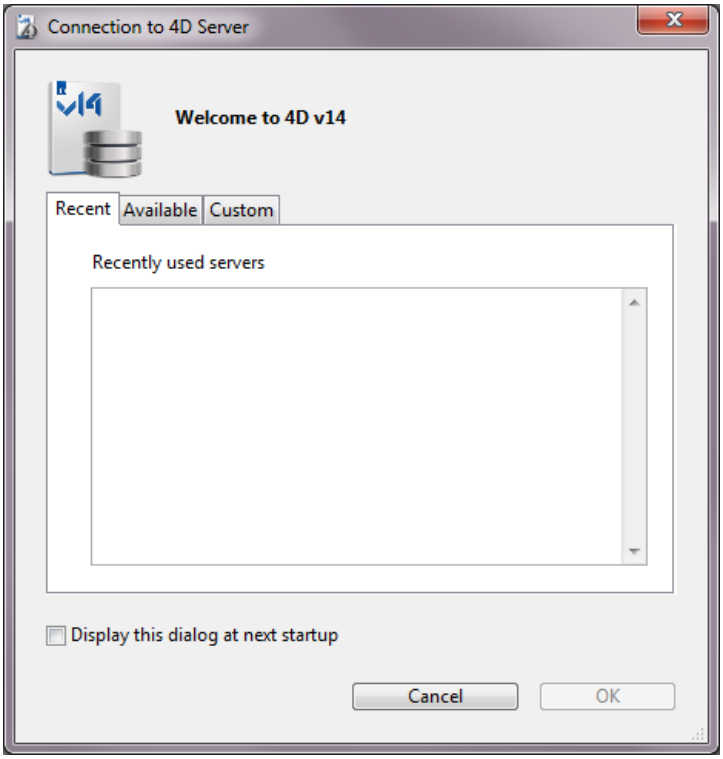
Note: このチュートリアルでは、4D Serverマシンと同じマシンにインストールされた4Dを使用できます。

4Dアプリケーションを始めて起動した場合または起動時の設定を変更していない場合は、Welcomeウィザードダイアログボックスが表示されます:

2. "4D Serverに接続"をクリック
または:
ダイアログボックスが表示されない場合、4Dのファイルメニューから開く>リモートデータベース... を選択

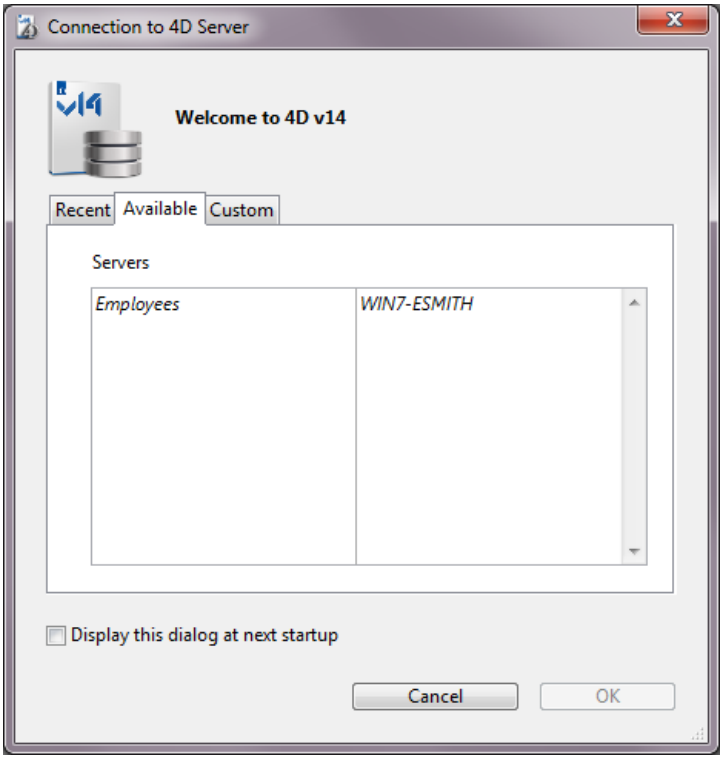


接続ダイアログボックスが表示されます:



3. 利用可能タブをクリックし、ネットワークに公開されている4Dデータベースを表示する

Employeesデータベースがリストに表示されているはずです:



4. Employeesを選択してOKをクリック

データベースがリモートワークステーション上で開かれます。デザインモードが開かれ、データベーストラクチャの作成準備ができました。

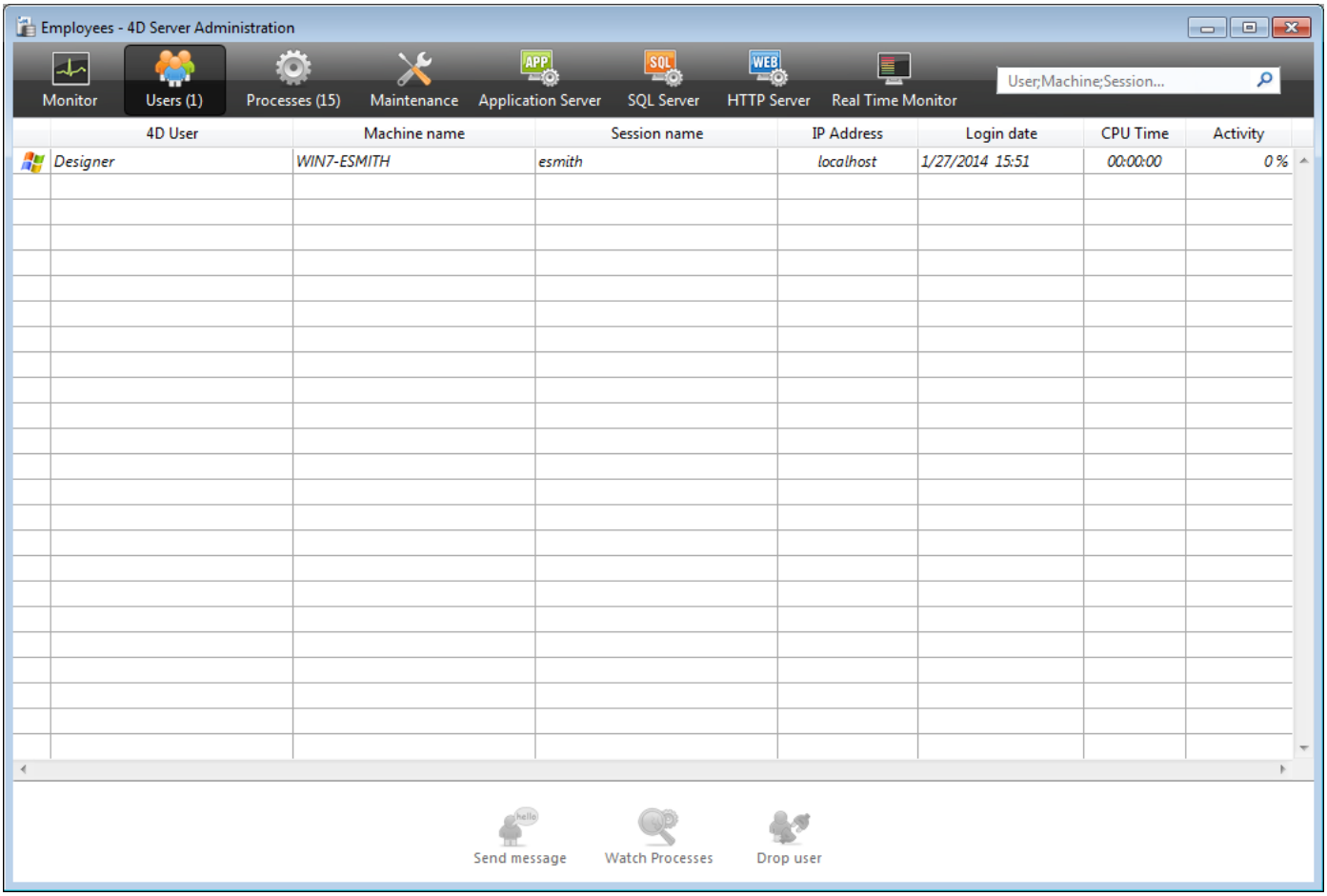
トラブルシューティングTips

4D Serverで作成したデータベース名が表示されない場合、以下の点をチェックします:

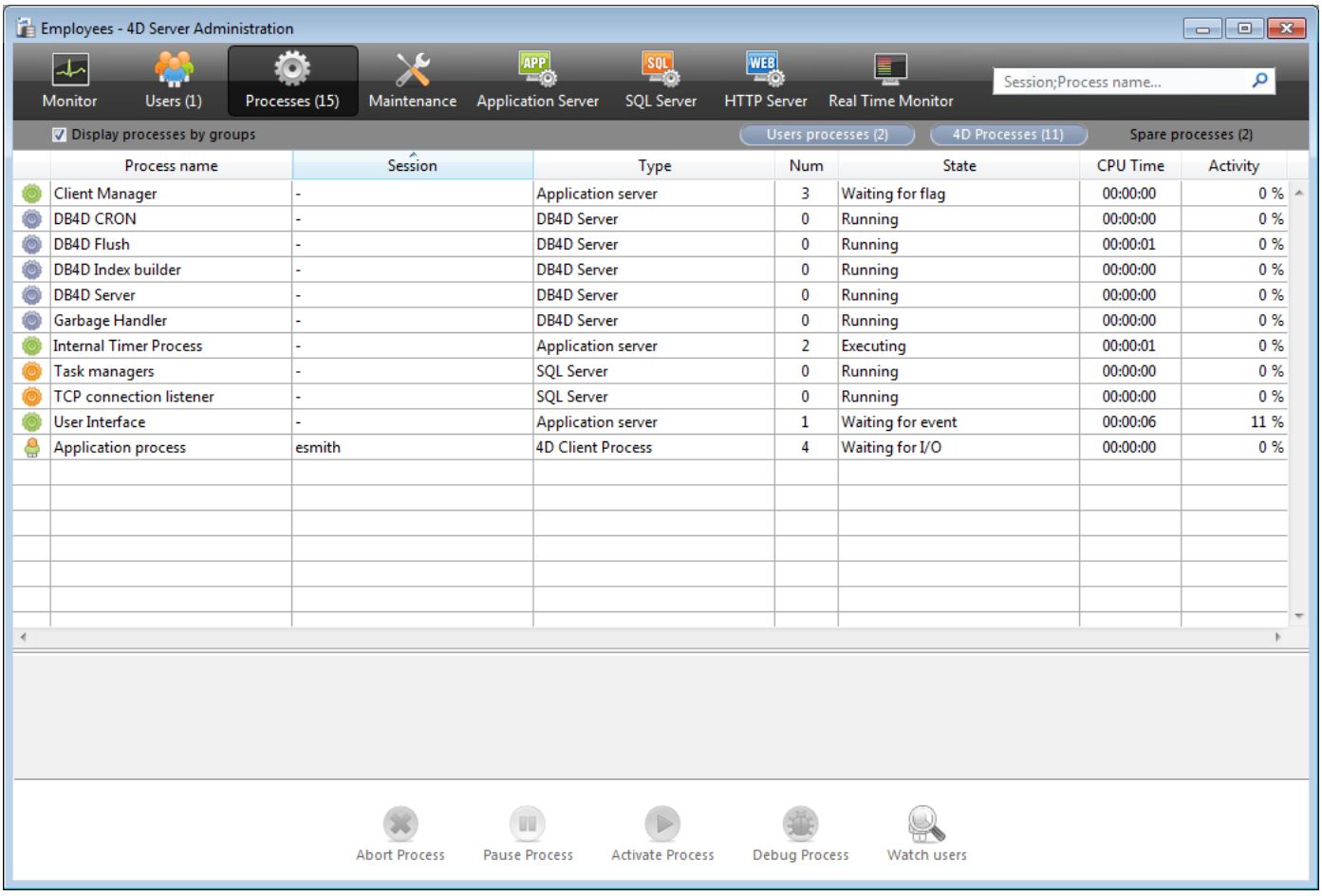
- 4D Serverが起動されているか
- 2台のマシンを使用している場合、それらはネットワークで接続されているか
- TCP/IPプロトコルが両方のマシンで正しく設定されているか
- 接続ダイアログボックスの使い方が不明な場合、**4D Serverデータベースへの接続**を参照してください。

サーバアクティビティ

4D Server管理ウィンドウには、対応するページにネットワークユーザ名が表示され、接続ユーザ数が (1) になっています。



新規プロセスが実行中です:



注: デフォルトでは、上図のように、予備プロセスは非表示となっています。**予備プロセス**ボタンをクリックすることで表示させることができます。

最初のプロセスは4D Server自身のもので、4D Server起動時に作成されます。新しいプロセスは最初にサーバに接続したユーザのものです:

- アプリケーションプロセスはレコード表示ウィンドウとアプリケーションモードを管理します。
- デザインプロセスはデザインモードを管理します。

ユーザが追加されるごとにいくつかのプロセスがプロセスリストに追加されます。

表示されるプロセスリストをフィルタできます。これを行うには**ユーザプロセス**、**4D プロセス**、**予備プロセス**ボタン、および管理ウィンドウ右上の検索/フィルタエリアを使用します。

次は

接続した後は、ローカルモードの4Dと同じ機能を使用してデータベースを操作できます。まずストラクチャを定義する必要があります。**データベースストラクチャの定義**に進んでください。

データベースストラクチャの定義

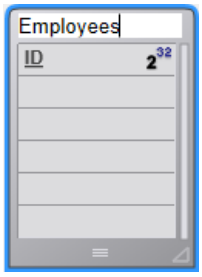
リモートの4D上で、サーバデータベースに接続した後 (**リモート4Dからサーバデータベースに接続する**リモート4Dからサーバデータベースに接続する参照)、デ**ザイン**メニュー内の**データベースストラクチャ**を選択します。

デフォルトで空のストラクチャウィンドウが現れます。これからテーブルを作成します。

[Employees] テーブルの作成

1. ファイルメニューまたは4Dツールバーから新規>テーブル... を選択
または
ストラクチャウィンドウ内で右クリックして、コンテキストメニューからテーブル追加を選択
または
ストラクチャウィンドウの追加ボタン (+) をクリックしてテーブルを選択
テーブルが作成されます。

2. タイトルエリアをクリックしてテーブル名 Employees**を入力**

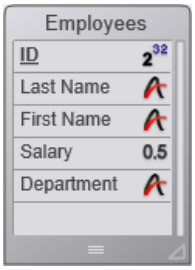


3. フィールドエリアをダブルクリックして、新規フィールドを作成

4. フィールド名をLast Nameに変更する。データ型は文字 (255) のまま
フィールドをダブルクリックしてインスペクタを表示できます。

5. 同様にして[Employees] テーブルに以下のフィールドを追加:

フィールド名	フィールド型
First Name	Alphanumeric (255 文字)
Salary	Real
Department	Alphanumeric (255 文字)



Note: 他の4Dリモートアプリケーションが同時にサーバデータベースで作業していると、あなたが作成したフィールドは少しの時間経過後他のクライアント上に表示されます。変更はサーバ上にリアルタイムで実装されますが、過度のスクリーン更新を避けるため他のスクリーンにすぐには現れません。

[Employees] テーブルのフォームを作成

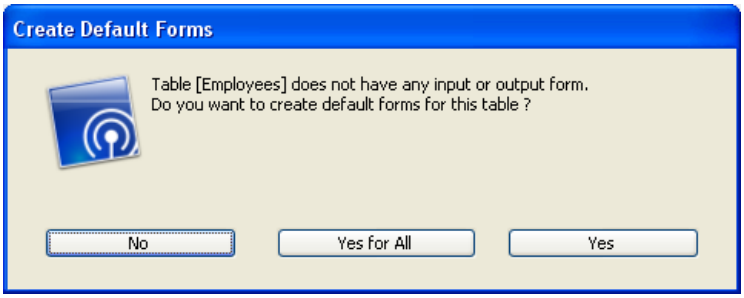
[Employees] テーブルを定義した後、レコードを操作するためにフォームが必要になります。フォームを作成するにはフォームウィザードを使用できます。しかし4Dではデフォルトの入出力フォームを作成する便利なショートカットがあります。

1. 4Dツールバーのテーブルボタンをクリック

または

レコードメニューからカレントテーブル表示を選択

レコードウィンドウが表示されます。4Dはテーブル用のフォームが作成されていないことを検知し、自動でフォームを作成するかどうか尋ねます。



2. はいをクリック

レコード追加および一件表示用の入力フォームと、レコードをリスト表示して更新できる出力フォームが作成されます。

次は

データ操作の準備ができました。**4D Serverでのデータ操作**に進んでください。

4D Serverでのデータ操作

データベースストラクチャの定義データベースストラクチャの定義の節で、[Employees] テーブルを作成し、4Dに自動でそのテーブル用のデフォルトフォームを作成させました。これでレコードを入力する準備が整いました。

レコードの入力

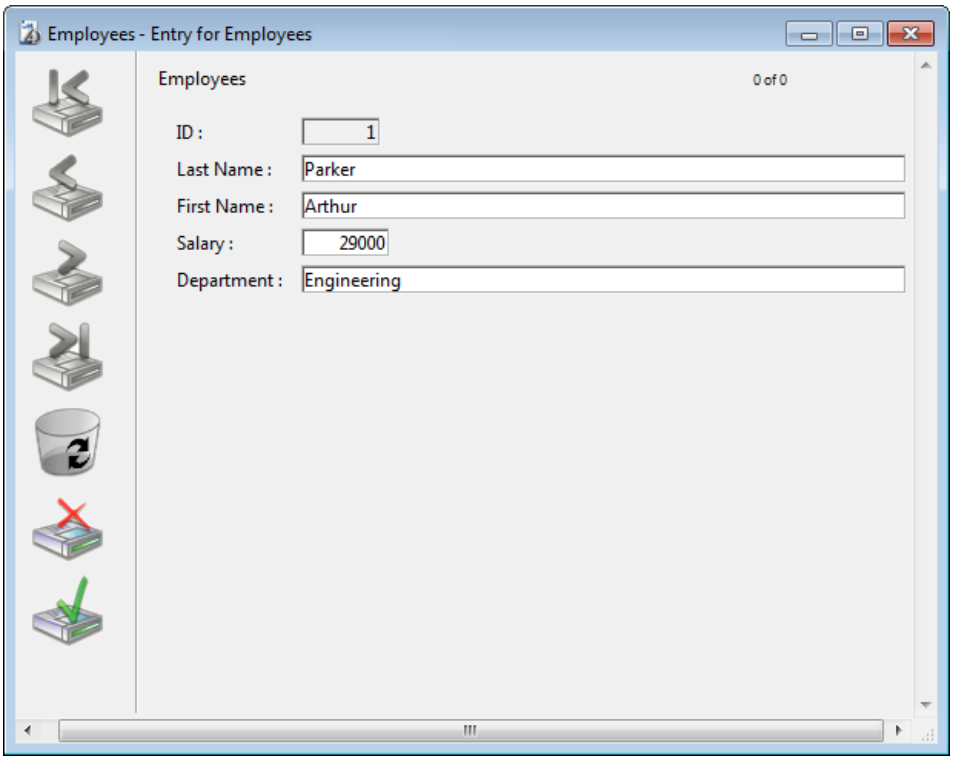
デザインモードで、4Dはレコードを入力、クエリ、印刷、更新するためのツールとエディタをデフォルトで提供しています。またアプリケーションモードで使用する独自のツールを設定することもできます。

1. レコードメニューから新規レコードを選択

空の入力フォームが表示されます。

2. 以下の通りにデータを入力

Tabキーやマウスを使用して、フィールド間を移動できます。



3. フォーム受け入れボタン (一番下のボタン) をクリックしてデータ入力を確定

空の入力フォームが再び表示され、引き続きレコードの追加ができます。

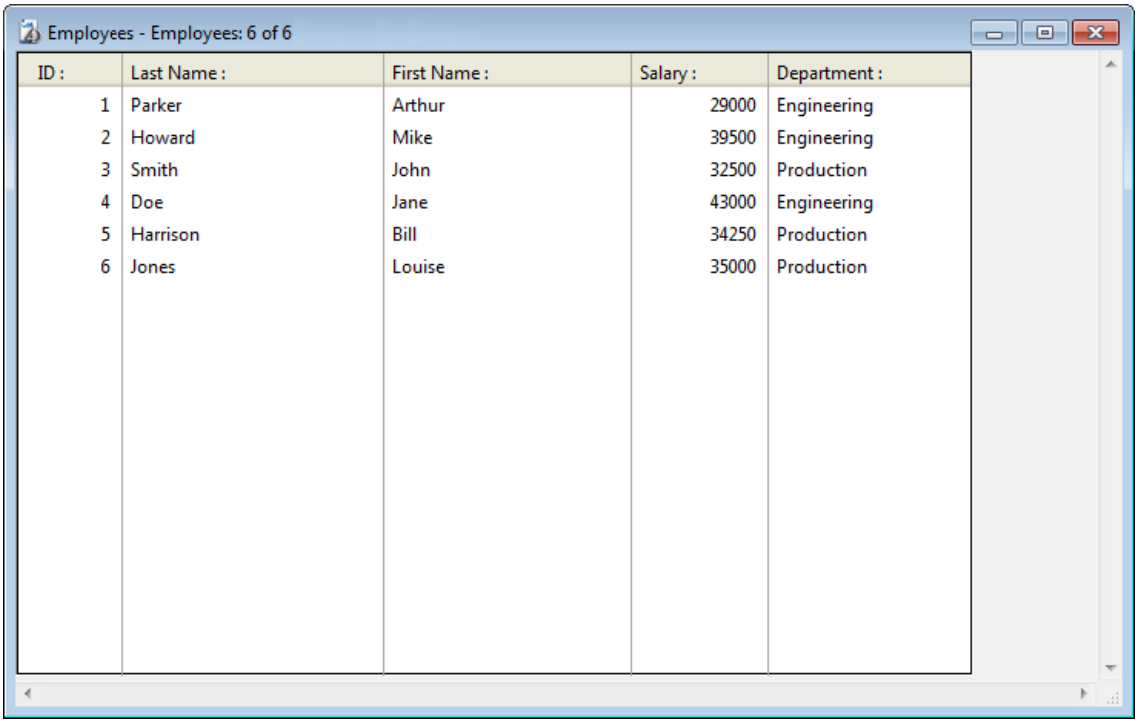
4. 以下のデータのとおりにさらに5つのレコードを作成

Last Name	First Name	Salary	Department
Howard	Mike	39500	Engineering
Smith	John	32500	Production
Doe	Jane	43000	Engineering
Harrison	Bill	34250	Production
Jones	Louise	35000	Production

最後のレコードを入力した後、キャンセルボタン (受け入れボタンの上のXが書かれたボタン) をクリックしてからの入力フォームをキャンセルします。出力フォームに戻ります。

5. 6つのレコードがすべて表示されていなければ、レコードメニューからすべて表示を選択し、必要に応じてウィンドウサイズや列のサイズを変更

以下のように表示されるはずです:

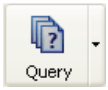


レコードはサーバマシンのデータベースに格納されています。2番目のリモート4Dマシンがサーバマシンに接続すると、追加したレコードが表示されます。また他のクライアントがレコードを追加すれば、**レコード**メニューから**すべてを表示**を選択することで、それらのレコードを含むすべてのレコードを表示できます。サーバに格納されたレコードはすべてのユーザがアクセスできます。

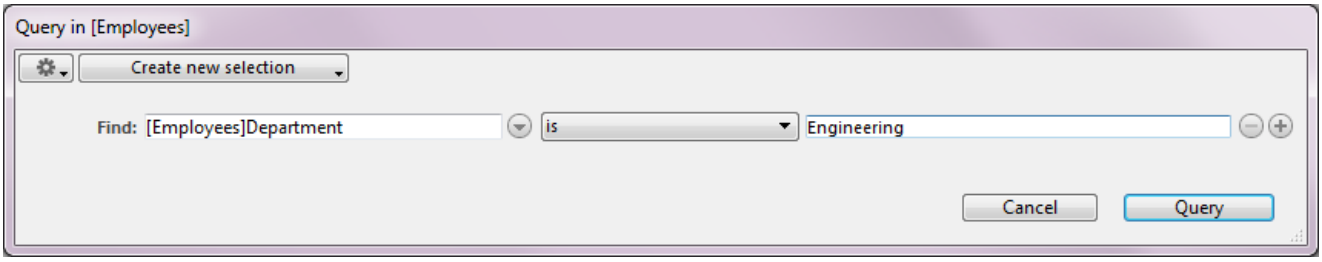
レコードの検索

[Employees] テーブルにレコードを登録したら、レコードの検索、並び替え、印刷などのレコード操作ができます。例としてEngineering部署の従業員を検索してみましょう。

1. ツールバーのクエリボタンをクリック



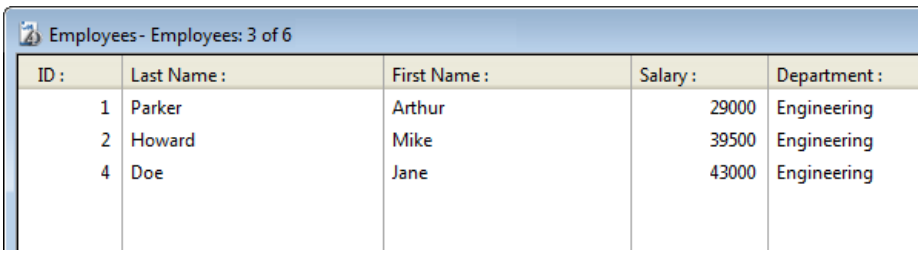
クエリエディタが表示されます:



2. "[Employees]Department" を検索フィールドにし、比較演算子リストから“=”を選択し、値に"Engineering"を入力

3. クエリをクリック

4D Serverにクエリが送信され、4D Serverは4Dに結果を送信します。出力フォームにはEngineering部署に所属する従業員のみが含まれます。



4. 再びすべてのレコードを表示するには、レコードメニューからすべてを表示を選択します。

次は？

ほんの数分でサーバデータベースを作成し、テーブルの作成、レコード追加、そしてデータベースに登録されたデータを使用した検索やチャートの作成を行いました。

次はデータベースにカスタムメニューバーを追加します。**カスタムメニューバーの追加**に進んでください。

カスタムメニューバーの追加

この節では、2つのメソッドとカスタムメニューバーをデザインします。つまりカスタム4Dアプリケーションを作成します。

2つのメソッドを追加

1. ファイルメニューから新規 > メソッド... を選択
新規メソッドダイアログボックスが表示されます。
2. 新規メソッドダイアログボックスに“M_ADD_RECORDS”と入力してOKをクリック
“Method:M_ADD_RECORDS”というタイトルのメソッドエディタが表示されます。
3. 以下のようにM_ADD_RECORDSメソッドのコードを入力

4. 2番目のメソッド“M_LIST_RECORDS”を作成し以下のコードを入力

2つのメソッドが作成されました。カスタムメニューバーを作成し、メニューコマンドにメソッドを割り当てます。

カスタムメニューバーの追加

1. デザインメニュー内のツールボックス > メニューを選択
デフォルトメニューバーを含むメニューバーエディタが表示されます。
2. タイトル"メニューバー番号1"を選択し、ウィンドウ中央の追加ボタンをクリック

3. メニュータイトルに“Tutorial”と入力してEnterを押す
4. タイトル“Tutorial”を右クリックして、メニュー"Tutorial"に項目を追加を選択

5. “Add Records...” と入力してEnterを押す
6. タイトル“Tutorial”を再び右クリックして、“Tutorial”メニューに2つ目の項目を追加
7. “Queries...”と入力してEnterを押す
メニューバー#1は以下のようになります:

8. “Add Records...” メニューコマンドをクリックしてメソッド名コンボボックスから“M_ADD_RECORDS”を選択

9. “Queries...” メニューコマンドをクリックしてメソッド名コンボボックスから“M_LIST_RECORDS”を選択

メニューバー#1は以下のようになります:

10. ツールボックスウィンドウを閉じる
これで終了です。
11. 実行メニューからアプリケーションテストを選択
追加したメニューからアプリケーションを使用することができます:

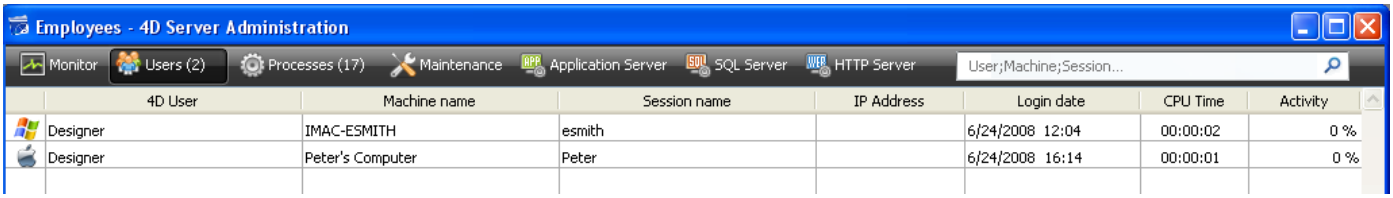
例えば**Tutorial**メニューから**Queries...**を選択すると、 標準の4Dクエリエディタが表示されます。クエリを定義し、検索されたレコードを表示して更新できます。
興味深い点は、あなたはすでに2つのアプリケーションを開発したということです。
この点について詳しく知るために、**4D Serverで同時に作業する**に進んでください。

4D Serverで同時に作業する

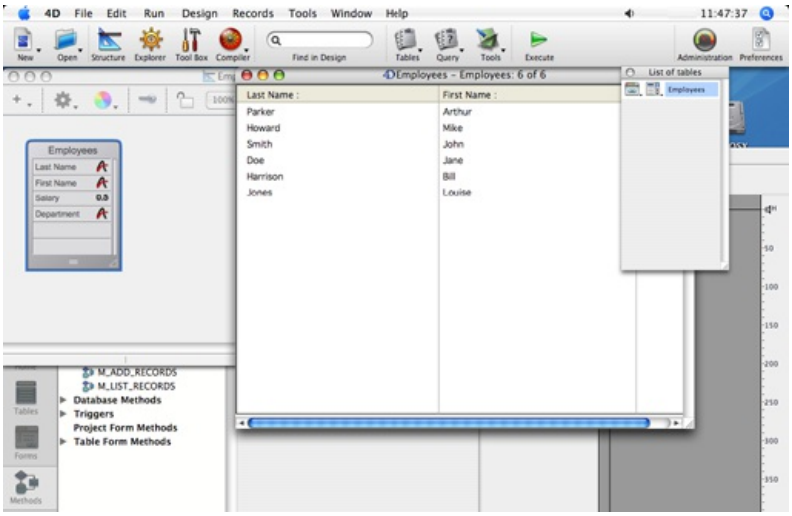
Windowsでこのチュートリアルを行っている場合、このサーバデータベースをそのままMacintosh上でも使用できます。
Macintoshでこのチュートリアルを行っている場合、このサーバデータベースをそのままWindows上でも使用できます。

2番目のユーザでサーバデータベースに接続する

このチュートリアルのために、Windows上のリモート4DとMac OS上のリモート4Dからサーバデータベースに接続します。接続すると、4D Server管理ウィンドウには2番目のユーザが表示されます。 (一番目の列はリモートマシンのOSを示しています):



他のプラットフォーム上で行われたすべてのことが、瞬時にまた透過的にそれぞれのクライアントマシン上で再利用可能です。以下の図はMac OSのクライアントマシンで見たユーザモードです:



6つのレコードと2つのメソッドが存在しています。

レコードを同時に使用する

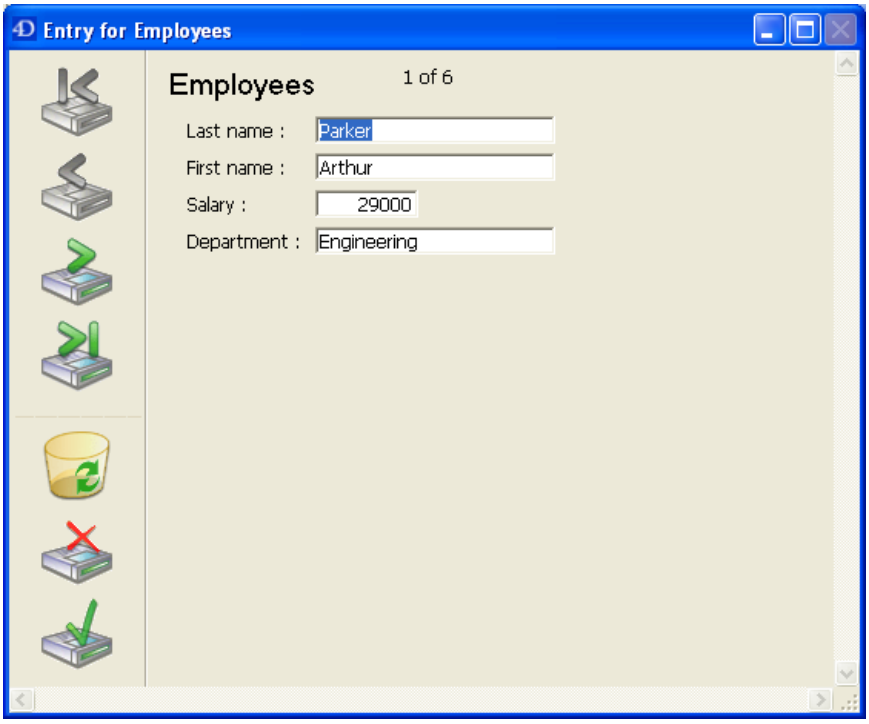
1. 最初のリモートマシン上で"アプリケーションテスト"モードに入り、TutorialメニューからQueries...を選択、“Department = Engineering”のレコードを検索

2. 2台目のリモートマシンでも同様に行う

両方のマシンで、3つのレコードからなるリストを取得します。

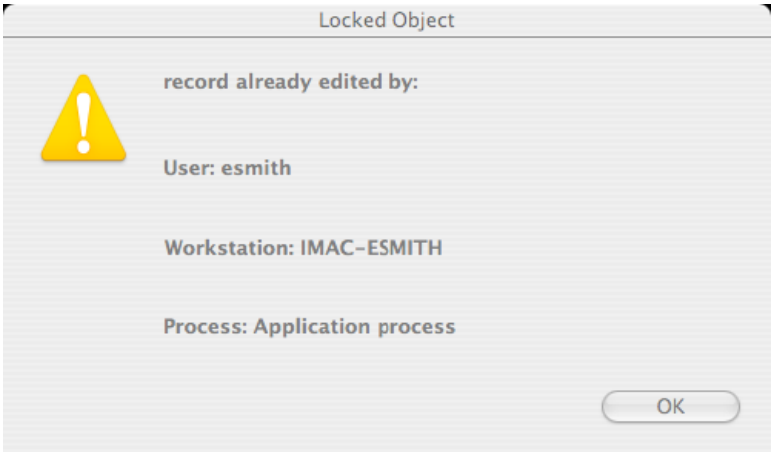
3. 最初のマシン上で“Parker, Arthur”のレコードをダブルクリック

以下のように表示されます:

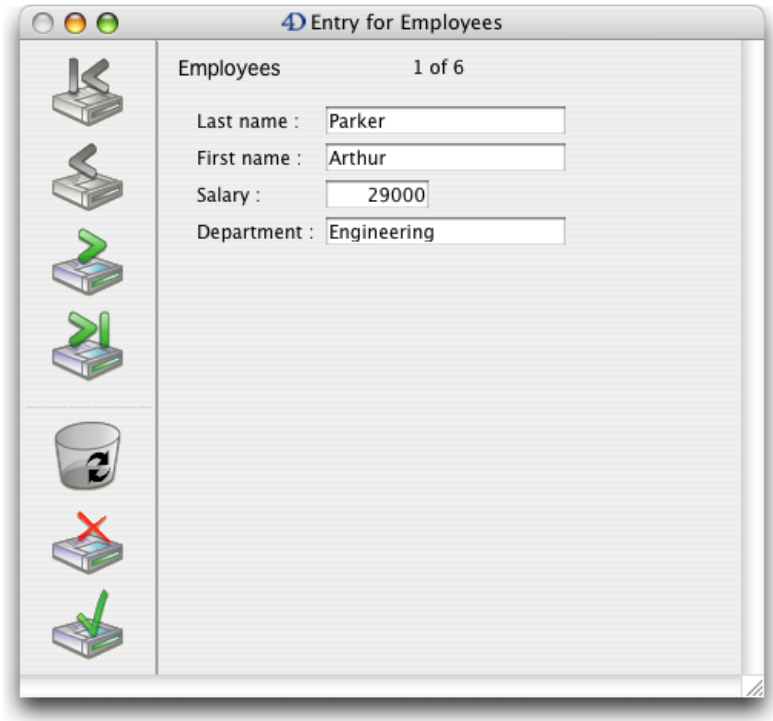


4. 2台目のマシンで同様に行う

4D Serverは組み込みのレコードロックメカニズムを備えており、レコードが既に使用中であることを警告します:

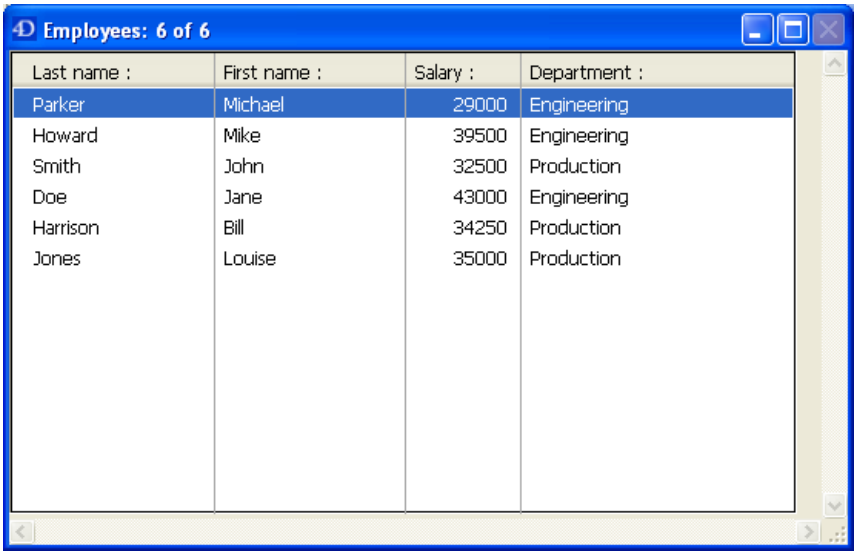


結果、このレコードには読み込みのみのアクセスが許可されます (表示はできますが更新できません)。



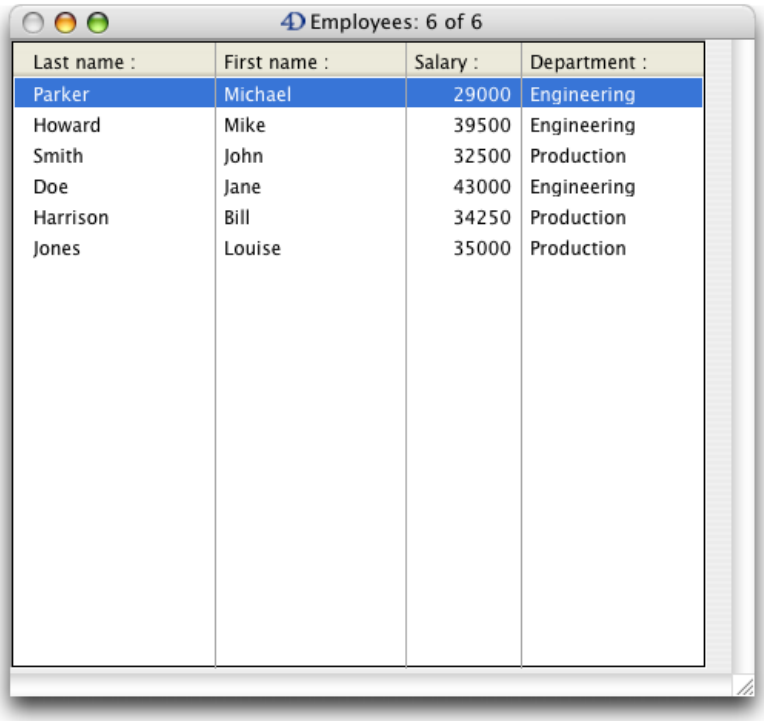
5. 1番目のマシンでfirst nameを“Michael”に変更し、変更を有効にする

リストが更新されます。



6. 2番目のマシン上で、入力フォームのレコード表示をキャンセル

こちらのリストも新しいfirst nameが表示されます。



デザインオブジェクトで同時に作業する

4D Serverはデータサーバであり、アプリケーションサーバでもあります。これが意味することを見てみましょう。

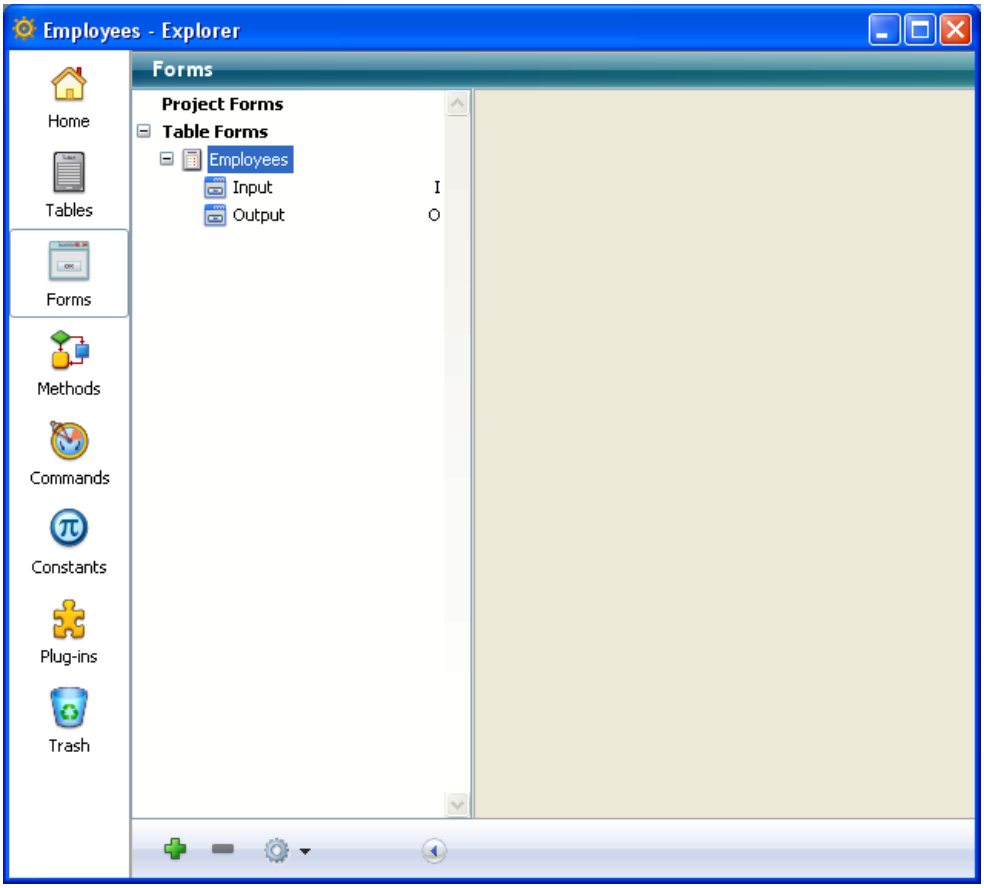
1. 2番目のマシン上でEsc.キーを押し、モードメニューからデザインモードに戻るコマンドを選択

2. 1番目のマシンでも同様に行う

3. 1番目のマシン上でデザインメニューからエクスプローラ > フォームを選択

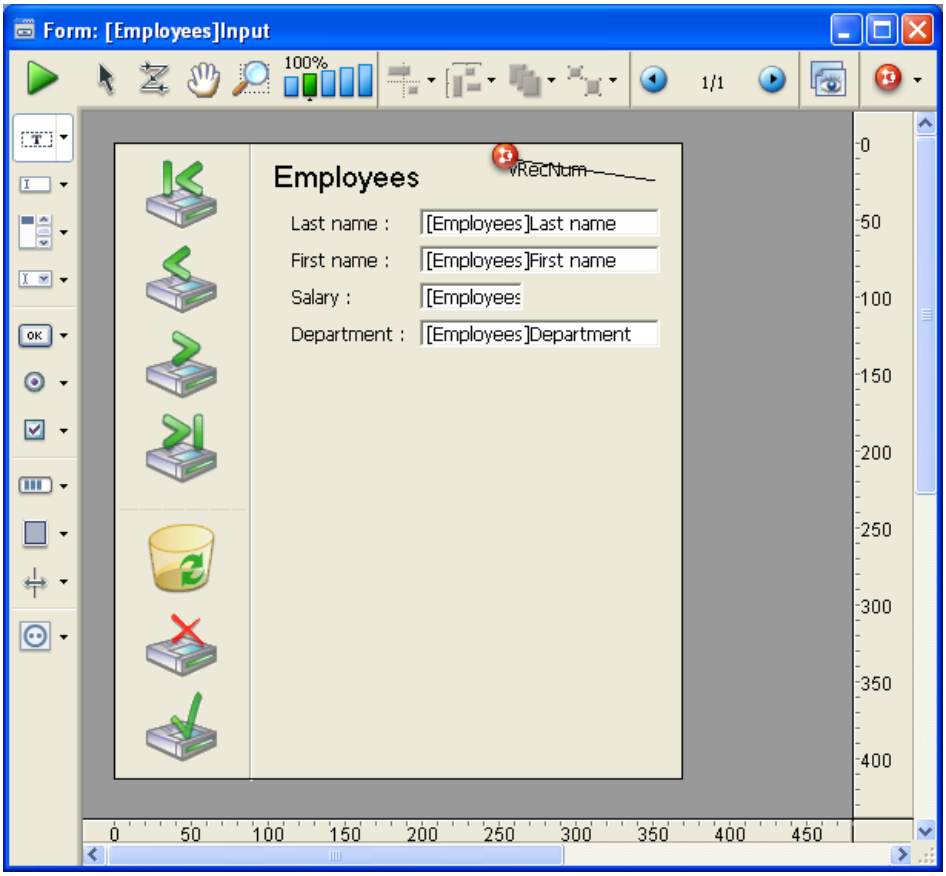
エクスプローラウィンドウが表示されます。

4. テーブルフォームを選択し、Employees テーブルを展開:



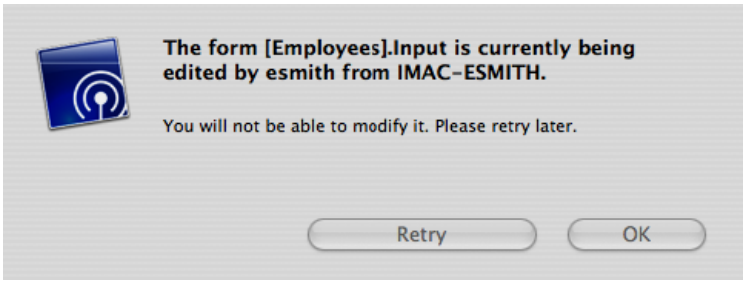
5. Inputフォームをダブルクリック

Inputフォームのフォームエディタが開かれます:



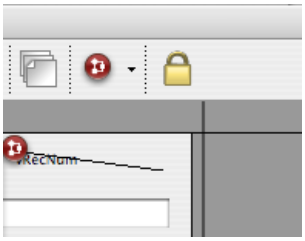
6. 2番目のマシンで同様に行う

他のマシン上ですでにフォームが編集中であるため、4D Serverに組み込みのオブジェクトロックメカニズムがそれを警告します:



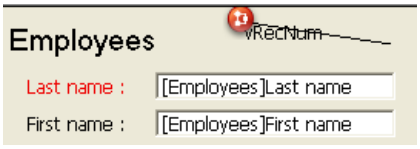
2番目のマシンでは、読み込みのみモードでフォームを開くことができます。オブジェクトを選択して他のフォームにペーストできますが、このフォームを編集することはできません。

フォームの右上には南京錠のアイコンがあり、このフォームが編集不可であることを示しています。



7. 1番目のマシン上で、[Employees]Last Name フィールドのラベル"Last name"を選択

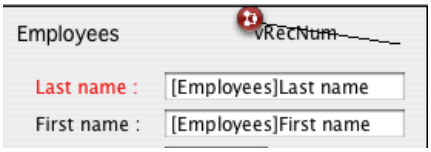
8. オブジェクト>カラー階層リストメニューを使用して、このオブジェクトの描画職を赤に設定



9. ファイルメニューから保存：フォーム：[Employees]Input を選択

10. 2番目のマシン上でそのフォームを閉じ再び開く

他のマシン上で行われた変更がこちらでも適用されます。



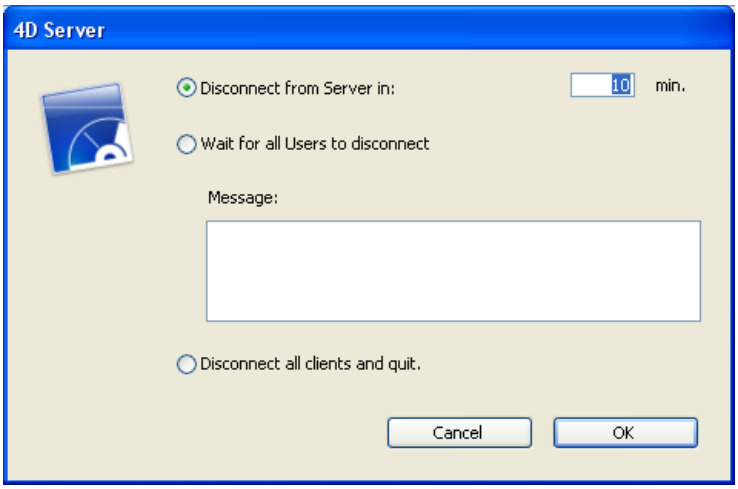
4D Serverを使用すれば他のユーザとともに同時にデータベースを開発できます。

サーバの終了

4Dリモートユーザにレコードやオブジェクトへの同時アクセスが発生していることを知らせる機能に加え、4D Serverはネットワーク経由で、サーバのシャットダウン警告を送信する機能を有しています。

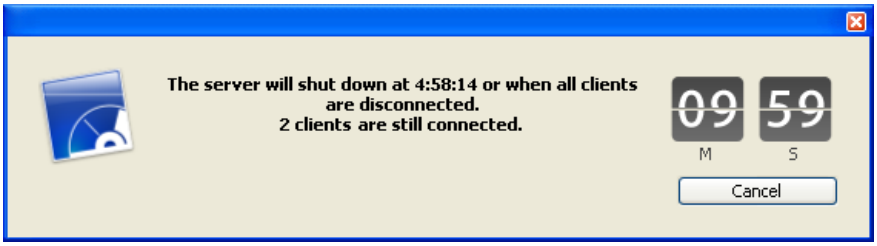
1. サーバデータベースに2つのクライアントを接続したまま、サーバマシン上でファイルメニュー (Windows) または4D Server メニュー (Mac OS) から終了を選択

終了ダイアログボックスが表示されます:



2. OKをクリック

ほとんど同時に、2つのクライアントマシンにサーバが終了中である旨の通知がされます。例えばクライアントでレコードを追加中である場合、ユーザにデータ入力を終了させるための時間が与えられます。



この警告ダイアログはそれぞれのクライアントマシン上で表示されます。

Note: 代わりに、"すべてのユーザの接続解除を待つ"オプションを選択して、できるだけ早く接続を解除するようメッセージを送信することができますし、また"すべてのクライアントの接続を解除して終了"オプションで即座にクライアントの接続を解除することもできます。

3. サーバが終了を待っている間に、リモートマシンの4Dを終了

まとめ

このチュートリアルで、4D Serverをいかに簡単に使用できるかを見ていただきました:

- データベースを一から作成しました。
- テーブルを定義し、4Dにフォームを作成させました。
- いくつかレコードを作成し、操作しました。
- 独自のメニューバーによりアプリケーションをカスタマイズしました。
- WindowsとMacintoshから同時にサーバデータベースに接続しました。
- サーバの終了を行いました。

最後に、1つのアプリケーションを作成する間に、あなたは2つのカスタムアプリケーションを作成しました (WindowsとMacintosh)。さらに、データベースをローカルモードで使用したくなったら、4Dで直接データベースを開くことができます。

4D Serverについてもっと知るには、このマニュアルの紹介の節や、4D Serverについて詳細に説明している他の節を参照してください。

4D環境に関する説明は、以下を参照してください:

- 4Dアプリケーションの利用とデザインについては、4D Design Referenceマニュアル
- 4Dランゲージのコマンドについて学ぶには、4D Language Referenceマニュアル

4D Serverを使用する

新しい4D Serverデータベースの作成

サーバデータベースを作成、あるいは既存のデータベースを開くには、4D Serverアプリケーションアイコンをダブルクリックして、4D Serverを開きます。

4D Serverの**ファイル**メニューから、新しくデータベースを作成したり、既存のデータベースを開いたりできます。

データベースの作成

新規にデータベースを作成するには、**新規>** サブメニューの中から一つを選択します:

:

- **データベース**: 空のデータベース、つまりテーブルやフォームが全く含まれていないデータベースを作成するために使用します。このコマンドを選択すると、標準のファイルを保存ダイアログボックスが表示され、データベースの名前と配置場所を指定できます。
- **テンプレートからデータベースを作成**: 準備されたテンプレートを基にしてデータベースを作成し、それをカスタマイズできます。この機能を利用するためには、テンプレートデータベースを格納し た"4D Templates"フォルダが4D Server.exe (Windows) や4D Server ソフトウェアパッケージ (Mac OS) と同じ階層になければなりません。このコマンドを選択すると、テンプレートデータベース選択ダイアログが表示されます。
- **ストラクチャからデータベースを作成**: XMLフォーマットのストラクチャ定義を使用して新しいデータベースを作成します。この定義は4Dから書き出すことができます。このコマンドを選択すると、標準のファイルを開くダイアログボックスが表示され、使用するXMLファイルを選択できます。

これらのオプションに関する説明は、Design Referenceマニュアルを参照してください。

データベースを開く

既存のデータベースを開くには、標準のドキュメントを開くダイアログボックスを使用するか (**ファイル>開く...** コマンド) 、以前に開いたことのあるデータベースを直接選択します (**ファイル>最近使用したデータベースを開く**コマンド)。
開くコマンドを選択したときに、既にデータベースが開かれていると、まずそのデータベースが閉じられます。クライアントマシンが接続していると、“すべてのユーザの接続解除を待つ”モードが使用されます (**4D Serverを終了する**参照)。

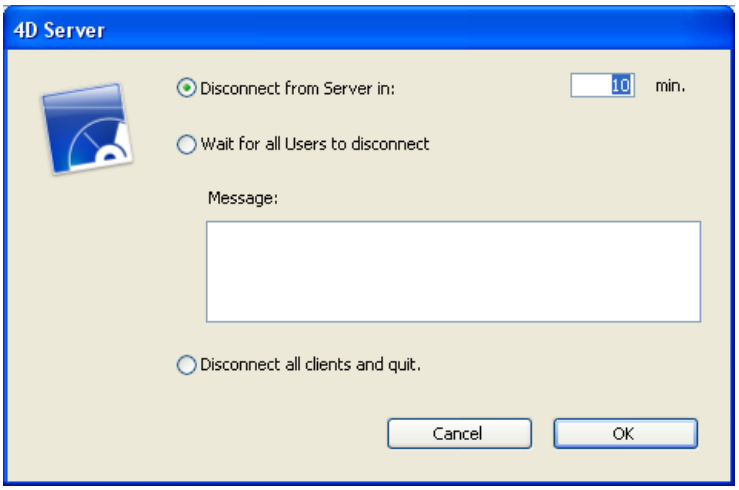
Note: インタプリタまたはコンパイル済みストラクチャ (.4db または .4dc) 、またはショートカットファイル (.4dlink) を4D Serverアプリケーションアイコンにドラッグ&ドロップして、直接既存のデータベースを開くこともできます。

4D Serverを終了する

サーバを終了するには:

1. 4D Serverのファイルメニュー (Windows)または4D Server メニュー (Mac OS) から終了コマンドを選択

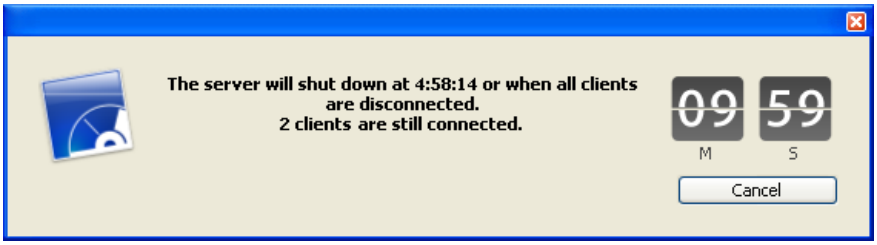
以下のダイアログがサーバマシン上で表示されます:



2. サーバを終了するまでの時間を分単位で入力するか、“すべてのユーザの接続解除を待つ”オプションを選択

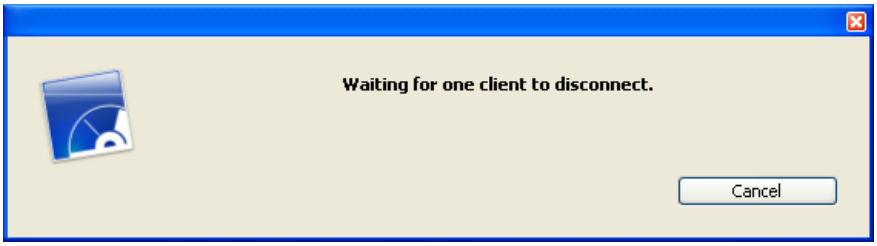
この作業を行うと、サーバへの新規接続は行えなくなります。

- “サーバは終了します:” オプションを選択すると、以下のウィンドウが表示されます:

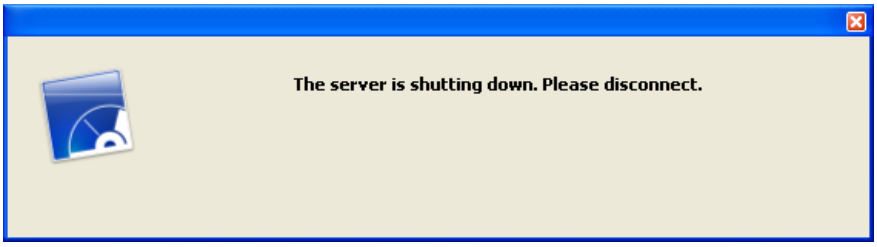


4Dクライアントマシンにも同じウィンドウが表示されます。4Dマシンにはキャンセルボタンは表示されません。このウィンドウは各クライアントマシン上で20秒ごとに更新され、クライアントの終了を促します。タイムリミットに達すると、クライアントマシンが接続されていても、サーバは終了します。

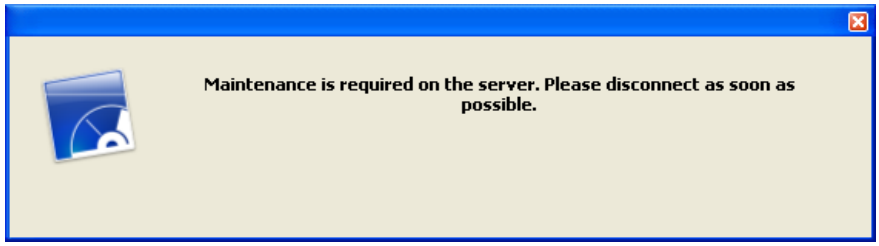
- メッセージを入力せずに“すべてのユーザの接続解除を待つ”オプションを選択すると、以下のウィンドウが表示され、接続中のユーザ数が示されます:



4Dクライアントマシン上には、デフォルトメッセージを使用して以下のダイアログが表示されます:



4D Server終了ダイアログボックスでカスタムメッセージを入力すると、クライアントマシン上でデフォルトメッセージの代わりに入力したメッセージが表示されます:



すべてのユーザが接続を解除次第、サーバは終了します。

- “すべてのクライアントの接続を解除して終了”オプションを選択すると、サーバはすべてのプロセスとすべての接続を終了して、数秒後に終了します。

Notes:

- いずれのケースも、終了ウィンドウを受け入れた際にサーバに接続しているクライアントがいない場合、4D Serverはすぐに終了します。
- 4D Serverシャットダウンウィンドウで**キャンセル**をクリックすると、サーバシャットダウン処理はキャンセルされます。
- **データベースを閉じる...**を使用して、4D Serverを起動したまま、データベースを閉じて、クライアントの接続を解除できます。詳細は**ON EVENT CALL4D Serverメニュー**を参照してください。

Using a 64-bit 4D Server (Windows)

バージョン12より、4D ServerはWindows 64-bit OSをサポートしています。64-bitテクノロジーの主な利点はより多くのRAMがアドレス可能になることです。

ここでは4D Server の64-bitバージョンについて説明します。

必要となる最低Windowsバージョン

6f-bit版の4D Server には、64bit 版オペレーティングシステムが必要になります。4Dウェブサイト上の4D-OS対応早見表をご覧の上、お持ちの4D Server リリースがどのwindowsのオペレーティングシステムと互換性があるかご確認ください。

アーキテクチャー

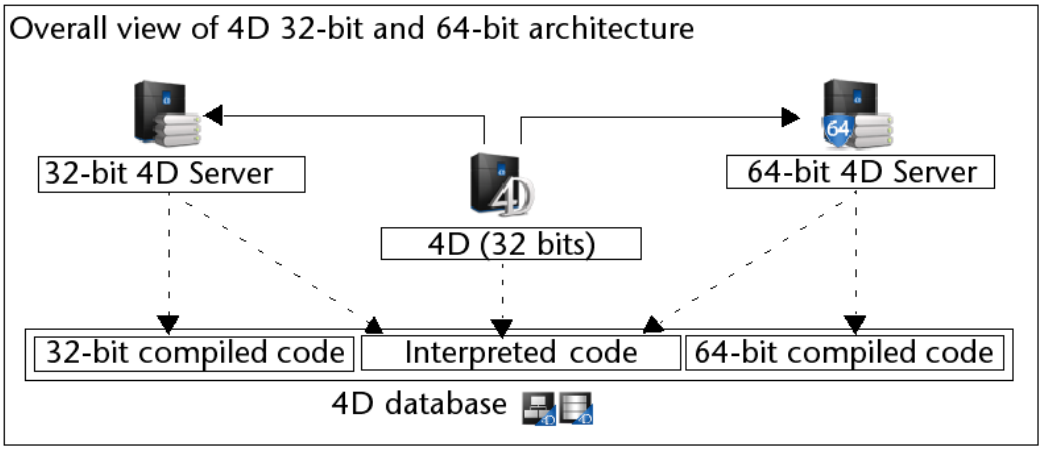
64-bit用の4D Server.exeアプリケーションは特別なバージョンであり、64-bit環境でのみ動作します。 32-bitシステム上では動作しません。

他方、標準の4DServer.exe (32 bits) をWindows 64-bitシステム上で起動した場合、動作しますが、それはエミュレーションモードになります。
クライアント側では、すべての4D マシーン (Mac OS および Windows) から64-bitバージョンの4D Server に接続できます。使用される4Dアプリケーションは標準の32-bitバージョンです (下図参照)。

互換性

インタープリターモードでは、同じ4Dデータベースを64-bit 4D Serverあるいは32-bit 4D Serverで実行できます。どちらのアプリケーションを使用しても、開発手順は同じです。

コンパイルモードでは、64-bit 4D Serverで実行させるために、64-bitプロセッサー用にコンパイルしなければなりません ("64-bit用にコンパイル"参照)。
32-bitのみ用にコンパイルされ、インタープリターコードを含まないデータベースは64-bit 4D Serverで実行できません。



実行時には、以下の相違点に留意してください:

- 64-bit 4D Serverには64-bitモード用にコンパイルされたプラグインのみがロードされます。64-bitプラグインはフォルダーとしてビルドされ、サーバーのPluginsフォルダーに配置されなければなりません (Win4DXフォルダーに配置される.4DXと.RSRファイルに基づく以前のアーキテクチャーはもうサポートされません)。32-bitプラグインは64-bit 4D Serverにロードされませんが、サーバー上のPluginsフォルダーに32-bitプラグインを配置し、リモートマシンに配布することはできます。この場合、サーバーを呼び出すメカニズムは動作しません (例えばサーバー上のテンプレートを読み書きする4D WriteのWR SET AREA PROPERTYコマンドなど)。
  32-bitの.4DXファイルはプラグイン内の/Contents/Windowsフォルダー内に置き、64-bit用のファイルは/Contents/Windows64フォルダーに置きます。
- 64-bit 4D Serverで使用するコンパイル済みの4Dコンポーネントは、64-bit用にコンパイルしなければなりません。
- アプリケーションがロードするBlobに使用されるメモリー量は依然2GBに制限されます。

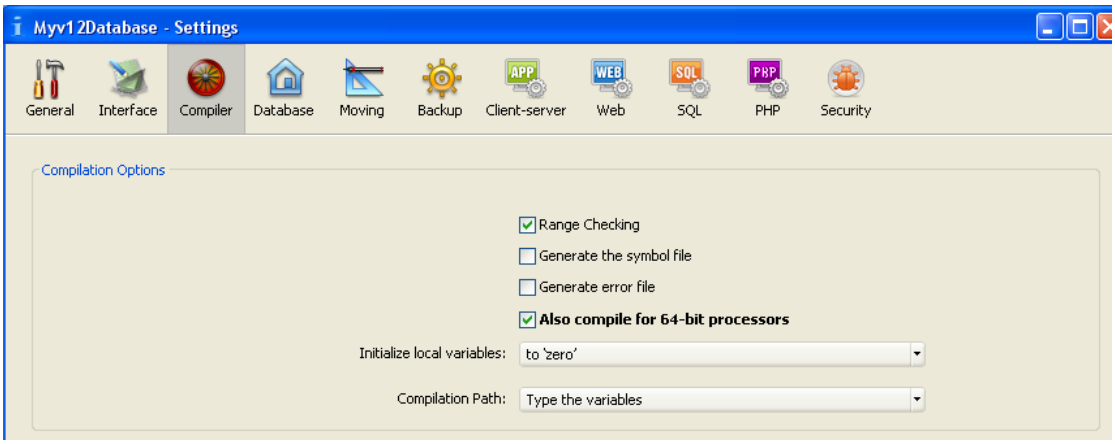
- QuickTimeは64-bit 4D Serverでサポートされません。
  4D社ではPICTフォーマットのピクチャーの利用は推奨しません。ピクチャーが100% Quickdrawであれば64-bitバージョンの4D Serverで扱うことができますが、ピクチャーにQuicktimeが含まれている場合、ロードすることができません。

4D Internet Commands

64-bit 4D Serverで4D Internet Commandsを利用できるようにするためには、バージョン12.1以降の4D Internet Commandsプラグインをインストールしなければなりません。

64-bit用にコンパイル

4D v12アプリケーションは32-bitと64-bitプロセッサー用にコンパイルできるようになりました。これを行うために、新しい**64-bitプロセッサー用にもコンパイルする**オプションがデータベース設定の"コンパイラー"ページに追加されました:



このオプションが選択されていると、コンパイラーは.4DCと.4DBファイルに64-bitコードと32-bitコードを含めます。結果これらのファイルを32-bitあるいは64-bitの4D Serverいずれでも実行できるようになります。デフォルトでこのオプションは選択されていません。

注: データベースを64-bitバージョンでコンパイルするにはUnicodeモードで動作しなければなりません。そうでなければコンパイル時にエラーが生成されます。

キャッシュメモリーのサイズ

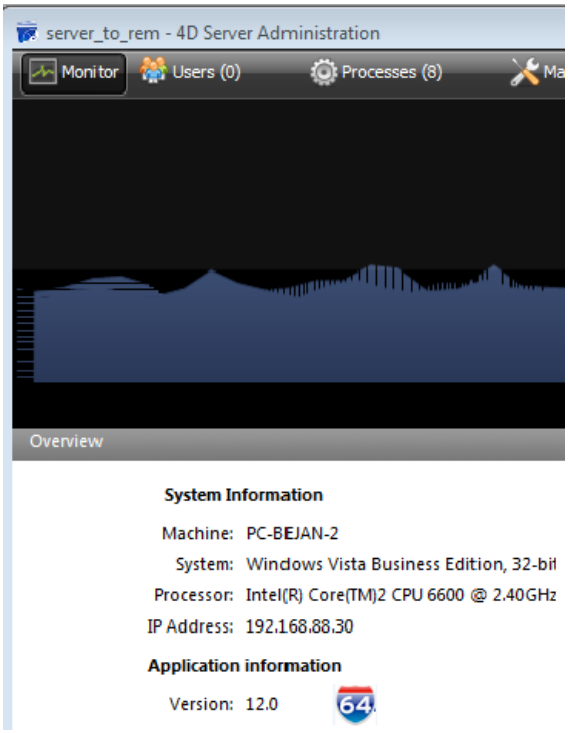
64-bitアーキテクチャーでは1 TB (1000 GB) までのRAMメモリーをアドレス可能になるので、64-bit 4D Serverに割り当てることのできるキャッシュメモリーは事実上無制限となります。

注: 比較すると、32-bitアーキテクチャーにおいては4 GBのRAMに制限されます (OSレベルで)。

データベース設定の"データベース/メモリー"ページで指定したキャッシュ量を確保できない場合、4D Serverは確保可能な最大サイズを割り当て、アプリケーションの起動時にユーザーに知らせます。ユーザーは終了するか、そのままのサイズで続行するかを選択できます。

インターフェース

実行中、64-bit 4D Serverアプリケーションか標準の4D Serverアプリケーションかは、サーバー管理ウィンドウのモニターページに表示されるロゴで見分けることができます:



注: このロゴは4D Serverについてウィンドウにも表示されます。

ランゲージ

変更されたコマンド

以下の4Dランゲージコマンドは新しい64-bit 4D Serverをサポートするために変更されました: **Version type**、**SET DATABASE PARAMETER**、そして**Get database parameter**。詳細はそれぞれのコマンドの説明を参照してください。

プロセススタックのサイズ

64-bitバージョンの4D Server上で走るプロセスのスタックは、32-bitバージョンよりも多くのメモリーを必要とします (約2倍)。**Execute on server**や**New process**コマンドを使用して64-bitバージョンの4D Server上でプロセスを作成する場合、最低128,000 byteを*stack*引数に渡すことを、呼び出し連鎖が大きくなる場合やスタックが足りないというエラーが発生する場合にはさらにそれを増やすよう推奨します。コードが64-bit 4D Server上で実行されるためのものである場合、この引数をチェックするようにしてください。

64-bitサーバーで利用できないコマンド

4D Serverで使用できない標準コマンド(**ストアドプロシージャ**参照) に加え、以下のコマンドは64-bitサーバーで実行できません。これらのコマンドを例えばストアドプロシージャーで呼び出すと、警告ダイアログが表示さ れ、エラー67が返されます。このエラーは**ON ERR CALL**コマンドでインストールされるエラー処理メソッドでとらえることができます。

4D, **クイックレポート**テーマ	**QR New offscreen area**
4D Chart, CT Area テーマ	CT New offscreen area

OS X用4D Serverの64ビット版の使用(暫定版)

4D v15では、OS X用の64 ビット版の4D Server のオペレーショナル・プレビュー版を提供しています。この新製品により、お使いの4D Server アプリケーションで64ビットのAppleマシンの実力を全て引き出す事ができるようになります。64-ビット版のテクノロジーの主な利点は、より多くのRAMメモリーを割り当てることができる、という点です。

このセクションでは、64ビット版の4D ServerをMac OS Xで導入・使用する際の注意点を扱っていきます。

OS Xの必須バージョン

OS X用の64ビット版4D Serverを使用するためには、**10.9** (Mavericks)以上のOSが必要になります。お使いの4D Serverのバージョンがどのでご利用いただけるかは、4D Webサイトにある4D-OS対応早見表を参照して下さい。

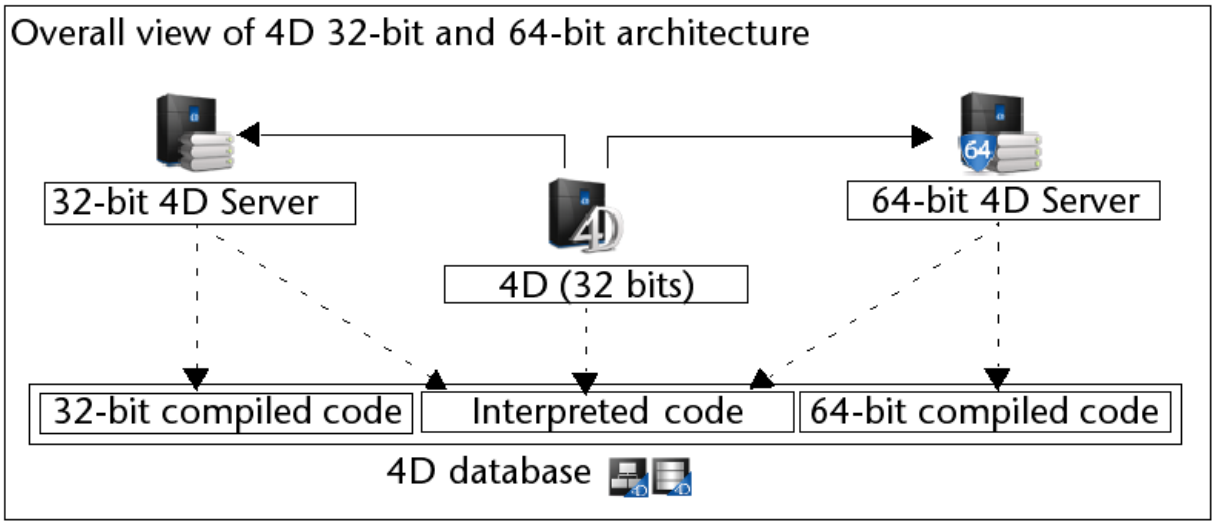
アーキテクチャー

64ビットアーキテクチャー用の4D Server はその環境専用のバージョンです(32ビットOS上では動作しません)。

クライアント側 j からは、4Dアプリケーションは(OS X用・Win用どちらでも)64ビット版の4D Serverにアクセスする事ができます。この場合接続するのに使用する4Dは標準の32ビット版です(以下のダイアグラムを参照して下さい)。

インタープリタモードでは、同じ4Dデータベースを64ビット版の4D Serverでも32ビット版4D Serverでも実行する事ができます。開発は、使用するアプリケーションを問わず、同じように進めることが可能です(ただし以下の制約が付随します)。

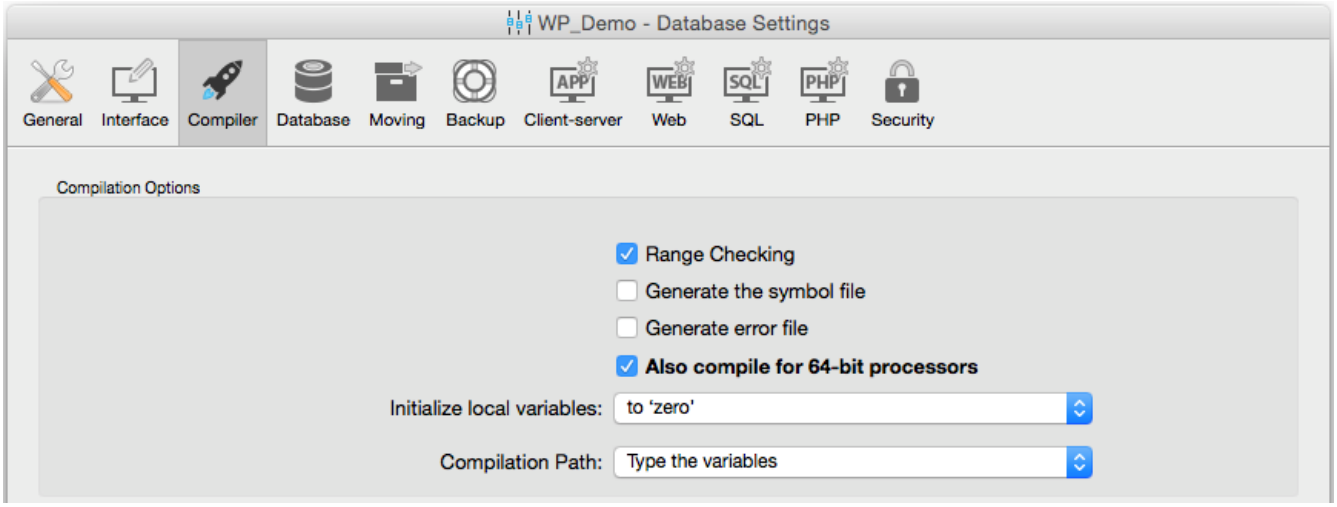
コ ンパイルモードでは、64ビット版4D Serverで実行するためには64ビットプロセッサー用にコンパイルされている必要があります。32ビット用にコンパイルされていて、インタープリタ コードを含まないデータベースは64ビット版4D Serverで実行することはできません。



データベースがプラグインを使用する場合、OS X用64-bit版プラグインがサーバーマシン側においてインストールされていなければなりません。**4D Internet Commands** と **4D Pack** はどちらもOS X 用の64 ビット版でもご利用いただけるという点にご注意ください。

64-bit 用コンパイル

4Dアプリケーションは、32ビット用と64ビット用とにコンパイルすることができます。これをするためには、データベース設定の「コンパイラー」ページにある、**64-bit プロセッサー用にもコンパイルする**オプションをチェックする必要があります:



このオプションがチェックされているとき、コンパイラーは64ビット用のコードと32ビット用のコードを.4DC と .4DBファイルに含めます。それにより、これらのファイルは32ビット版・64ビット版、両方の4D Serverで実行することが出来るようになります。デフォルトでは、このオプションはチェックされていません。

コンパイルされたコードの互換性

OS X 64-bit アーキテクチャーをサポートするために、4D内蔵のコンパイラーが変更されました。その結果、4D v15(またはそれ以降のバージョン)でコンパイルされたデータベースのみが OS X 64-bitで実行可能となります(**注**: コンパイラーは*4D v14 R3*以降変更されています)。つまり:

- 既存の4DデータベースをOS X 64-bitにおいてコンパイルモードで使用したい場合、そのデータベースを4D v15(またそれ以降のバージョン)で再コンパイルする必要があります。
- データベースがコンパイルされたのコンポーネントを使用する場合、そのコンポーネントを4D v15(またそれ以降のバージョン)で再コンパイルする必要があります。

プロセススタックのサイズ

64-bitバージョンの4D Server上で走るプロセスのスタックは、32-bitバージョンよりも多くのメモリーを必要とします (約2倍)。**Execute on server**や**New process**コマンドを使用して64-bitバージョンの4D Server上でプロセスを作成する場合、デフォルト値(0)、または少なくとも512KBを*stack*引数に渡すことを、呼び出し連鎖が大きくなる場合やスタックが足りないというエラーが発生する場合にはさらにそれを増やすよう推奨します。コードが64-bit 4D Server上で実行されるためのものである場合、この引数をチェックするようにしてください。

サポート対象外の機能

以下の機能・技術はカレントのOS X用の64bit版4D Serverではサポートされません:

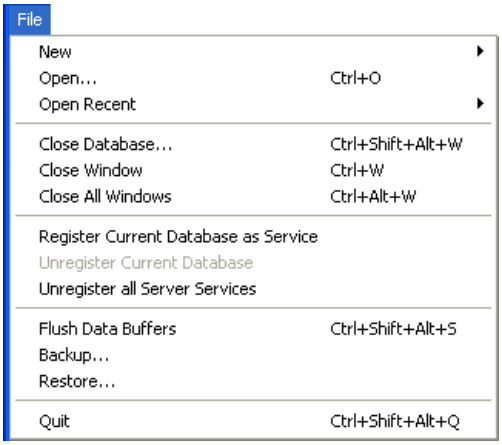
機能/技術	コメント
XSLT と Xalan	**_o_XSLT APPLY TRANSFORMATION**、 **_o_XSLT SET PARAMETER**、 そして **_o_XSLT GET ERROR** は動作しません。代わりに PHP *libxslt* モジュールを使用して下さい。
PICT フォーマット	画像の代わりに、'サポート外の画像フォーマット'画像 + ファイル拡張子が表示されます(**利用不可能なピクチャーフォーマット**を参照して下さい)。PICTフォーマットは4D全体で使用廃止になっています。**AP Is Picture Deprecated**も参照して下さい。
cicn アイコン	**GET ICON RESOURCE**コマンドはサーバーではサポートされていません(*)。
データベース .RSR ファイル	データベース .RSR ファイルは、自動的には開かれません。**Open resource file** を使用する必要があります。
書き込み可能なリソースファイル	**_o_Create resource file** はサーバー上ではサポートされていません(*)。リソースファイルは読み込み専用でのみ開く事ができます。 **注意:** *Mac OS* リソースファイルは、既に*4D v11*から廃止予定となっていました。
_o_Font number	このコマンドはサーバー上ではサポートされていません。(*)
ASCII 互換モード	Unicodeモードのみがサポートされています
旧式ネットワークレイヤー	*ServerNet* のみがサポートされています(**新しい ServerNet ネットワークレイヤー(互換性)**を参照して下さい)
シリアルポート通信	利用不可
読み込み/書き出しダイアログボックス	利用不可
ラベルエディター	利用不可

(*) サーバー側で実行した場合にはエラーが返されます

4D Serverメニュー

4D Serverアプリケーションのインタフェースは以下のメニューで構成されています: **ファイル**, **編集**, **ウィンドウ**, **ヘルプ**。Mac OSではいくつかのコマンドは**4D Server**メニュー (アプリケーションメニュー) に置かれます。

ファイル



新規

この階層コマンドにはサブメニューがあり、サーバマシン上でデータベースやデータファイルを新しく作成するために使用します。
データベースを作成するコマンドについては**新しい4D Serverデータベースの作成**の節で説明しています。

開く... / 最近使用したデータベース

これらのコマンドを使用して4D Serverでデータベースを開くことができます。**最近使用したデータベース**コマンドは、4D Serverが最近開いたことのあるデータベースを含むサブメニューを表示します。このメニューをリセットするには、**メニュークリア**コマンドを選択します。
データベースを開くコマンドについては**新しい4D Serverデータベースの作成**で説明しています。

データベースを閉じる...

このコマンドは4D Serverを終了せずに、現在のデータベースを閉じます。このコマンドを選択するとサーバ終了ダイアログが表示され、接続ユーザの接続解除モードを指定できます (**4D Serverを終了する**参照)。

ウィンドウを閉じる

このコマンドは4D Serverアプリケーションの最前面にあるウィンドウを閉じます。

すべてのウィンドウを閉じる

このコマンドは4D Serverアプリケーションのすべてのウィンドウを閉じます。この場合、データベースが公開されていることを示す要素は**ファイル**メニューの**データベースを閉じる...**のみとなります。

現在のデータベースをサービスとして登録 / 現在のデータベースの登録解除 / すべてのサーバサービスの登録解除

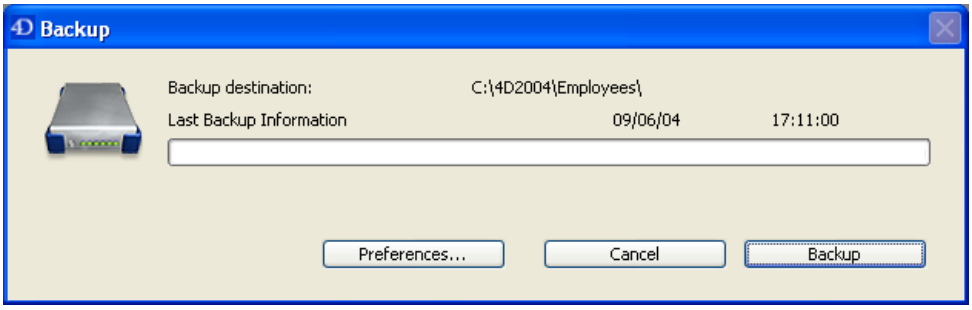
(Windowsで利用可能) これらのコマンドを使用してサービスとして登録するデータベースを管理します。この機能は**データベースをサービスとして登録**で説明しています。

データバッファをフラッシュ

このコマンドはキャッシュ中のデータを強制的にディスクに保存します。4D Serverはデータベース環境設定 (データベース/データ管理ページ) で指定された時間が経過すると自動でキャッシュをフラッシュします。

バックアップ

このコマンドを使用していつでもデータベースのバックアップを起動できます。このコマンドを選択すると、以下のダイアログボックスが表示されます:



- **バックアップ**ボタンは、データベース環境設定で設定された (バックアップするファイル、アーカイブの場所、保持するセット数などの) パラメタを使用して、即座にバックアップを起動します。
- **環境設定**ボタンは環境設定のバックアップテーマを開き、現在のバックアップ設定を確認して、必要であれば編集できます。
- **キャンセル**ボタンはバックアップ処理を中断します。

バックアップ設定に関する詳細は、4D Design Referenceマニュアルを参照してください。

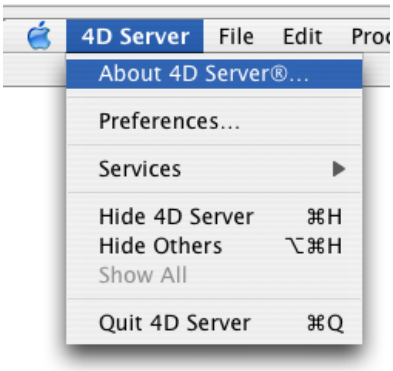
復元...

このコマンドはファイルを開くダイアログを表示し、復元するアーカイブファイルを選択できます。

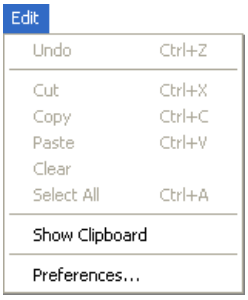
終了

このコマンドを使用して4D Serverアプリケーションを閉じることができます。詳細は **4D Serverを終了する** を参照してください。

Note: Mac OS Xでは**終了**コマンドは**4D Server**メニュー (アプリケーションメニュー) 内にあります。



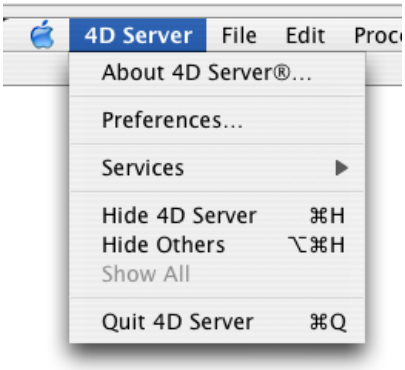
編集



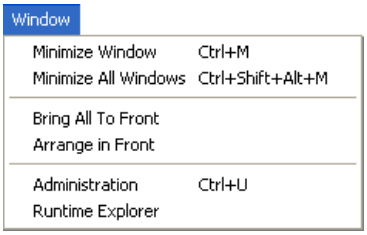
4D Serverの**編集**メニューは標準のコピー/ペーストコマンドや、**クリップボード表示**コマンド等を含みます。

Windowsではこのメニュー内に、アプリケーションの環境設定ダイアログを表示する**環境設定...**コマンドがあります。このダイアログボックスを使用してデータベースの様々な動作を定義できます。このダイアログに関する詳細は4D Design Referenceマニュアルを参照してください。4D Server用の環境設定については **設定 (環境設定)** と **公開 (環境設定)** の節で説明します。

Note: Mac OSでは**環境設定...** コマンドは**4D Server**メニュー (アプリケーションメニュー) 内にあります。



ウィンドウ



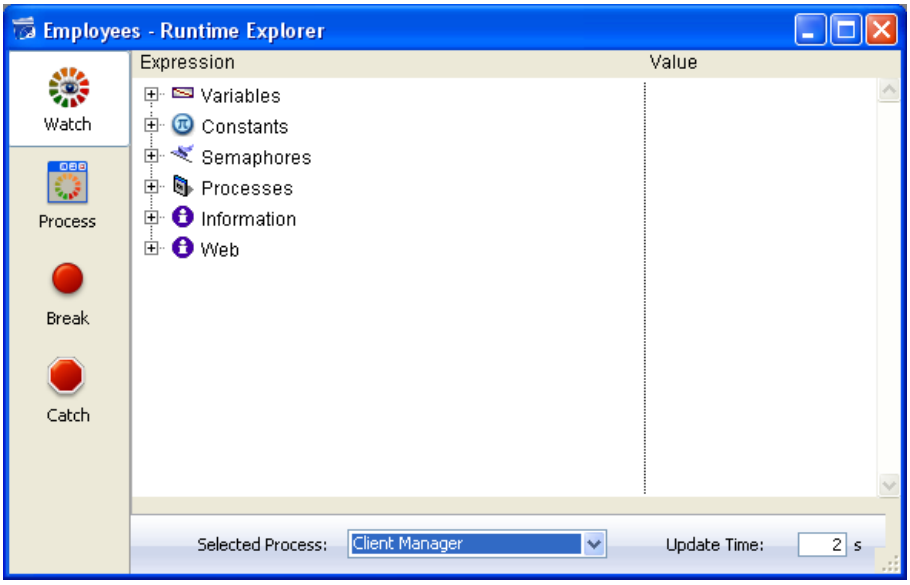
ウインドウメニューの上部にはワークスペースウィンドウを扱うための標準のコマンドがあります (これらのコマンドはプラットフォームより異なります)。
また4D Server特有のウィンドウを表示するためのコマンドも含まれています。

管理

このコマンドは、もしそのウィンドウが閉じられていたり最小化されていれば、4D Server管理ウィンドウを表示します。このウィンドウについては4D Server管理ウィンドウの章で説明しています (**C_POINTERモニタページ**参照)。

ランタイムエクスプローラ

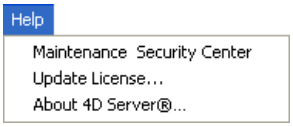
このコマンドは4D Serverのランタイムエクスプローラウィンドウを表示します。



ランタイムエクスプローラでは、データベースの様々なストラクチャ要素の状態を見たり、利用可能なリソースが正しく管理されているかをチェックできます。ランタイムエクスプローラは特に開発時やデータベース検証中に便利です。

ランタイムエクスプローラには4ページあり、それぞれのページにはウォッチ、プロセス、ブレーク、キャッチボタンからアクセスできます。ランタイム エクスプローラは4Dと4D Serverで同じ動作を行います。詳細は4D Design Referenceマニュアル(**ランタイムエクスプローラー**の章)を参照してください。

ヘルプ



Maintenance & Security Center

このコマンドはMaintenance & Security Center (MSC) を開きます。ここにはデータベースの検証、解析、メンテナンス、バックアップ、圧縮を行うのに必要なすべてのツールが集められています。

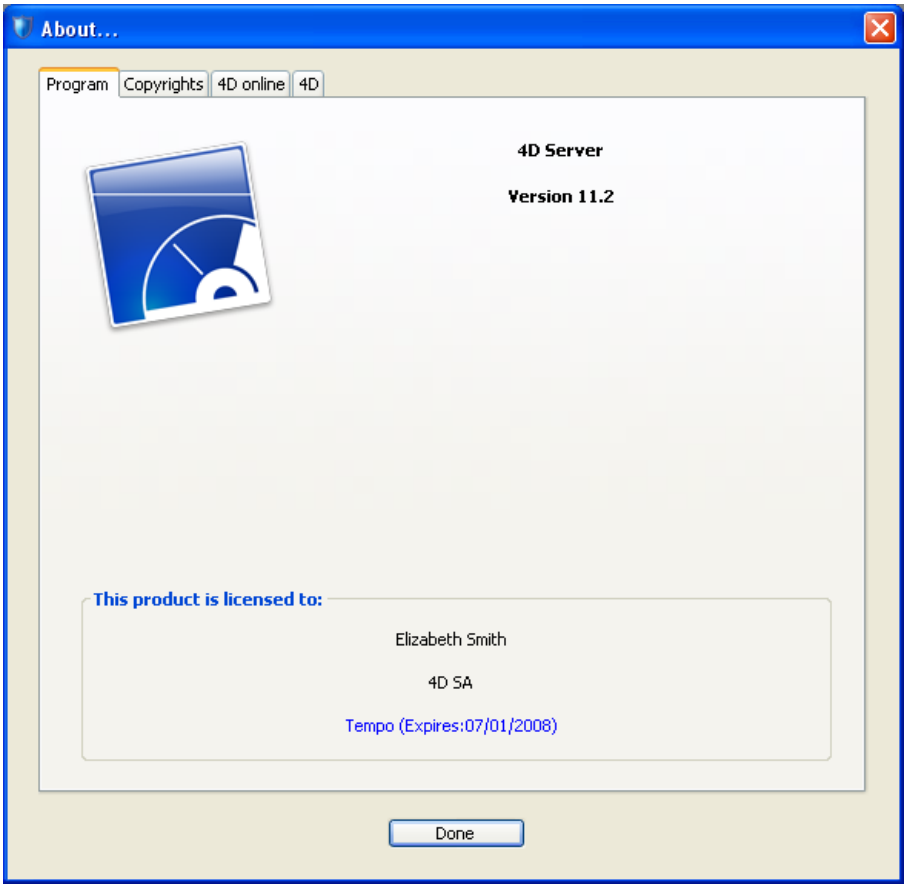
こ のコマンドは4D Serverがデータベースを開いていない時でも使用できます。この場合、このコマンドをメンテナンスモードでデータベースを開くために使用できます (選択すると標準のファイルを開くダイアログが表示され、開くデータベースを選択できます)。メンテナンスモードは圧縮や破損したデータベースを開く際に 使用されます。
MSCに関する詳細はDesign Referenceマニュアルを参照してください。

ライセンス更新...

このコマンドは4D環境で追加のライセンスをアクティベートするために使用するウィンドウを表示します。
このダイアログに関する詳細は、4Dインストールガイドを参照してください。

4D Serverについて...

このコマンドは4D Serverについて... ウィンドウを表示し、タブでアクセス可能なページに様々な情報を表示します:

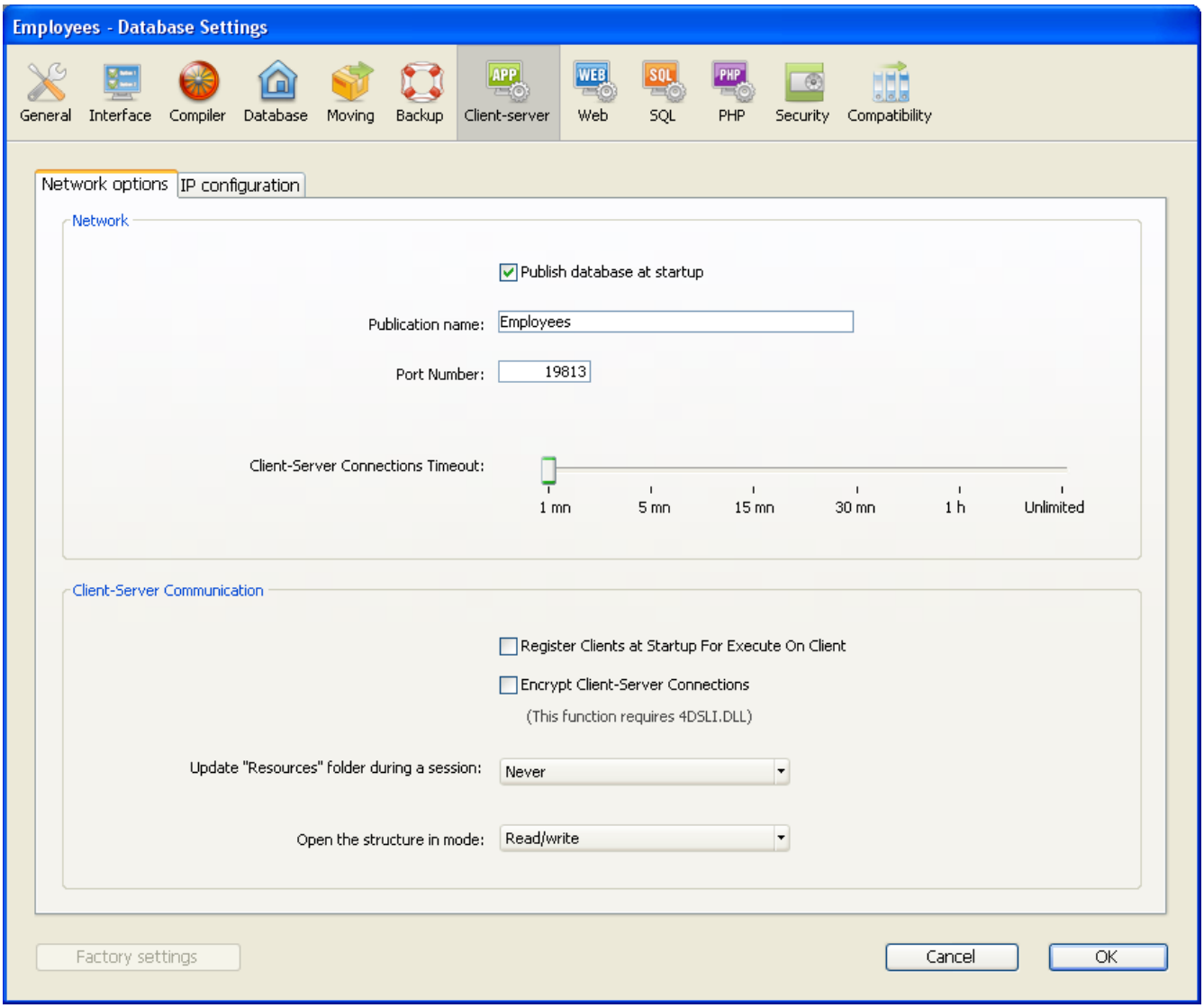


- プログラム: 4D Serverのバージョンとライセンス
- Copyrights: 法的な表示
- 4D オンライン: オンラインでアクセス可能な4Dに関する追加のリソース
- 4D: 世界中の4D SAS

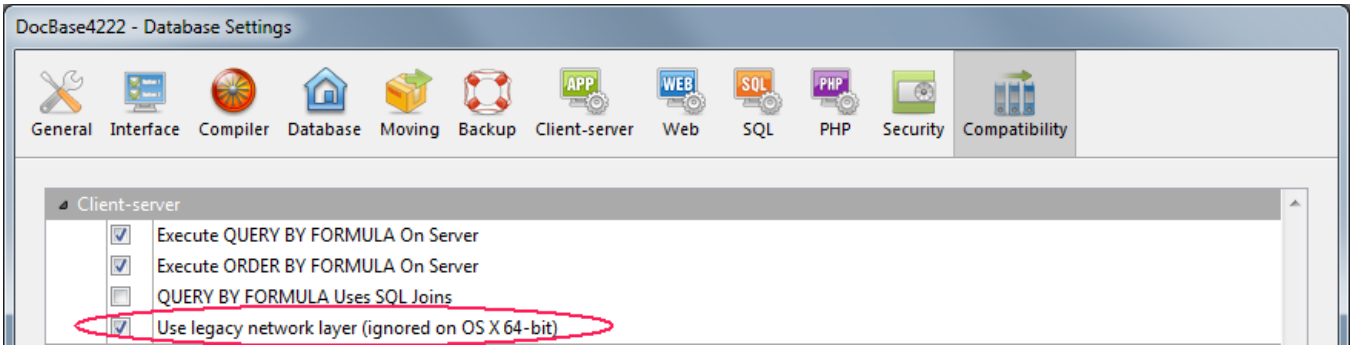
Note: Mac OSでは**4D Serverについて** コマンドは**4D Server**メニュー (アプリケーションメニュー) 内にあります。

設定 (環境設定)

ネットワークやクライアント-サーバー間の通信に関する様々なパラメーターを、データベース設定の**クライアント-サーバ**ページ、"ネットワークオプション"タブで設定できます。 (リモートモードの4Dおよび4D Serverからアクセス可能):



さらに、4D Server v14 R5以降、互換性オプションのタブから、旧式のネットワークレイヤーをいつでも有効化/無効化できるようになりました:



ここではこれらの引数について説明します。

ネットワーク

起動時にデータベースを公開する

このオプションを使用して、4D Serverデータベースが起動されると自動で公開データベースのリストに表示されるようにするかどうかを指定できます。

- このオプションが選択されていると (デフォルトオプション)、データベースは公開され、公開データベースの一覧に表示されます (リモート4Dの**TCP/IP**ページ)。
- このオプションがチェックされていないと、データベースは公開されず、公開データベースの一覧には表示されません。接続を行うには接続ダイアログボックスの**カスタム**ページに手作業でデータベースのアドレスを入力しなければなりません。

備考: この設定を変更したら、設定を有効にするためにサーバデータベースを再起動する必要があります。

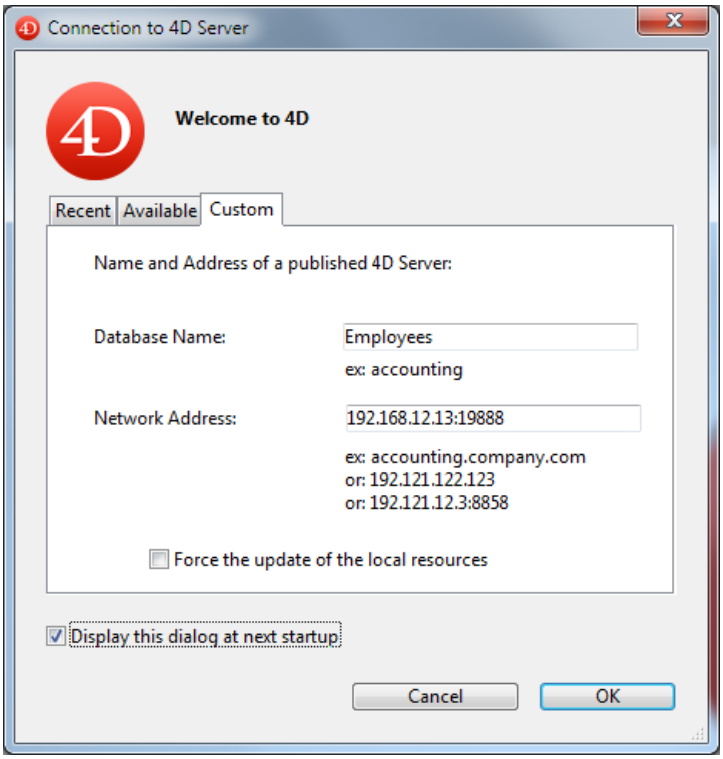
公開名

このオプションでは、4D Serverがデータベースを公開する際に使用する名前を設定できます。この名前は例えば接続ダイアログボックスの利用可ページに表示されます (**4D Serverデータベースへの接続**参照)。
デフォルトで4D Serverはデータベースストラクチャファイル名を使用します。これを好きな名前に変更できます。

備考: このパラメタはカスタムクライアント-サーバアプリケーションでは使用されません。クライアントアプリケーションは接続ダイアログを経由せずにサーバに直接接続します。しかしエラーが発生すると、このダイアログが表示されます。このケースでは、サーバアプリケーションの公開名はコンパイルされたデータベースの名前です。

ポート番号

このオプションでは4D Serverのデータベース公開ポート番号を変更できます。この情報はデータベースのストラクチャ及びそれぞれのクライアントマシンに格納されます。4D Serverとリモートモードの4Dが使用するデフォルトのTCPポート番号は19813です。
TCPプロトコルを使用して、1台のマシン上で複数の4Dアプリケーションを同時に使用したい場合にこの値の変更が必要です。この場合、アプリケーションごとに異なるポート番号を割り当てなければなりません。
4D Server または 4Dからこの値を変更すると、データベースに接続しているすべての4Dマシンに変更が通知されます。接続していないクライアントを更新するには、次回の接続時、接続ダイアログボックスのカスタムページにサーバマシンのIPアドレスに続けてコロン、そして新しいポート番号を入力します。例えば新しいポート番号が19888であるとき:



備考: 4Dクライアントと同じポート番号で公開されているデータベースだけが、接続ダイアログの利用可ページに表示されます。

4D Serverとポート番号

4D Serverは3つのTCPポートを使用して内部サーバとクライアントの通信を行います:

- **SQLサーバ**: デフォルトで19812 (環境設定の"SQL/設定"ページで変更可)。
- **アプリケーションサーバ**: デフォルトで19813 ("クライアント-サーバ/設定"ページで変更可)。

- **DB4Dサーバ**(データベースサーバ): デフォルトで19814。このポート番号を直接は変更できません。常にアプリケーションサーバのポート番号+1です。

4D クライアントが4D Serverに接続するとき、アプリケーションサーバのポート番号 (19813または接続ダイアログボックスのIPアドレス欄でコロンの後ろに指定された番号) を使用して接続します。サーバごとのポートを使用する他のサーバへの接続は自動です。再度ポート番号を指定する必要はありません。
ルータやファイアウォール越しに接続する場合、3つのポートを明示的に開く必要があります。

クライアント-サーバ接続タイムアウト

このサーモメータを使用して、4D Serverとクライアントマシン間で (一定時間活動が行われないときに接続を閉じる) タイムアウトを設定できます。
制限なしオプションはタイムアウトを設定しないことを意味します。このオプションが選択されると、クライアントのアクティビティコントロールは行われなくなります。
タイムアウトが選択されると、指定された時間リクエストを受け取らない時、サーバはそのクライアントとの接続を閉じます。

クライアント-サーバ通信

Execute On Clientのために起動時にクライアントを登録

このオプションが選択されていると、データベースに接続するすべての4Dリモートマシン上で、リモートからメソッドを実行できます。このメカニズムについては**クライアントマシン上でのストアドプロシージャ**で説明しています。

クライアント-サーバ接続の暗号化

このオプションを使用して、サーバーマシンと4Dリモートマシン間通信のSSLモード (保護モード) を有効にできます。このオプションについては**クライアント/サーバ接続の暗号化**で説明しています。

"Resources"フォルダをセッション中に更新

この設定は、セッション中にデータベースの**Resources**フォルダが更新されたとき、接続された4Dマシンの**Resources**フォルダのローカルインスタンスの更新モードをグローバルに設定するために使用されます(Resourcesフォルダはセッションが開かれるたびにリモートマシン上で自動的に同期されます)。3つの選択肢があります:

- **しない**: ローカル**Resources**フォルダはセッション中に更新されません。サーバから送信される通知は無視されます。ローカルの**Resources**フォルダは**ローカルリソースを更新**コマンドを使用して手動で更新できます。
- **常に**: ローカル**Resources**フォルダの同期は、サーバから通知が送信されると、セッション中に自動で行われます。
- **その都度指定**: サーバから通知を受け取ると、クライアントマシン上でダイアログボックスが表示されます。ユーザはローカル**Resources**フォルダの同期を受け入れあるいは拒否できます。

Resourcesフォルダはデータベースインタフェースで使用されるカスタムファイルを格納するものです (翻訳ファイルやピクチャなど)。自動又は手動メカニズムを使用して、このフォルダの内容が更新されたときに各クライアントに通知できます。
詳細は**リソースフォルダの管理**を参照してください。

ストラクチャを開くモード

このオプションは、クライアント側で通信をするときの介しモードをデータベースストラクチャに設定します。デフォルトは「読み／書き」モードです。「読み込みのみ」モードに設定すると、ストラクチャが更新されないようになります。

新しい ServerNet ネットワークレイヤー(互換性)

4D v14 R5 以降、4Dアプリケーションには4D Server と4Dマシン(クライアント)間の通信を管理するための、*ServerNet* という名前の新しいネットワークレイヤーが含まれるようになります。*ServerNet* は現代的で強固なAPIに基づいており、維持が簡単で、最新のネットワークテクノロジーを簡単に導入できる一方、高いレベルのパフォーマンスを発揮することができます。

以前の"旧式"のネットワークレイヤーは、既存のデータベースの互換性を確保するためにサポートはされます。新規に作成されたデータベース内では*ServerNet* ネットワークレイヤーが自動的に採用されます。

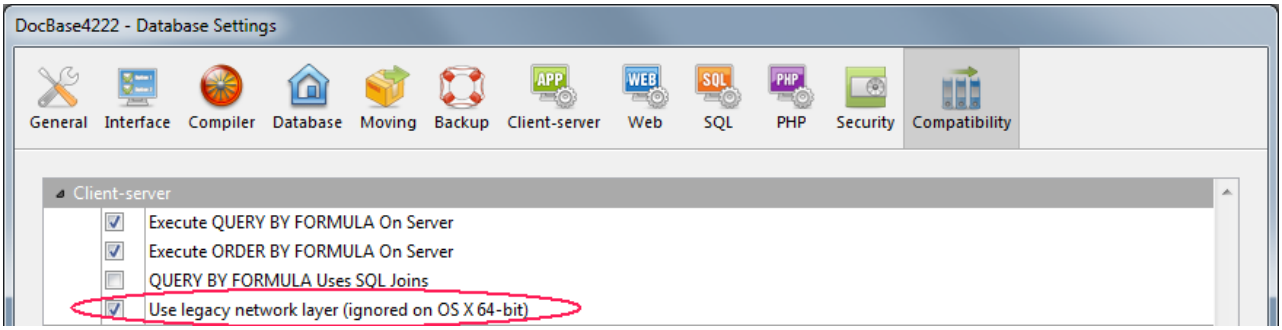
ServerNet にはそれを有効化/無効化するためのオプションがあります。

実装についての注記: *ServerNet* ネットワークレイヤーは 4D v15の "プレビュー" リリースで提供されています。

旧式ネットワークレイヤーの有効化または無効化

新しい互換性オプションによって、4D Serverにおける旧式ネットワークレイヤーをいつでも有効化または無効化することができます。以下のどちらかの方法を使用して下さい:

- **SET DATABASE PARAMETER** コマンドにUse legacy network layer 定数を使用する。
- データベース設定ダイアログボックスにおいて**旧式ネットワークレイヤーを使用**オプションを使用する(**互換性ページ**を参照して下さい):



注: その題名に書いてあるように、このオプションはOS X用の4D Server 64-bit版では無視されます。このプラットフォームでは*ServerNet* のみがご利用になれます。

デフォルトでは、このオプションは以下の様に設定されています:

- 4D v14 R5以降で作成されたデータベースにおいては自動的に*ServerNet* レイヤーが有効化されています。
- 変換された既存のデータベースにおいては自動的に無効化されています。

組み込み4Dクライアントを移行する

既存のサーバーアプリケーションにおいて*ServerNet* レイヤーを有効化した場合、適合する4Dクライアントアプリケーションのみが接続することができます:

- 4D v14 R4 以降のバージョンのクライアントは何も変更しないまま接続することができます。
- それ以前のバージョン(v14.x とR4より前のv14'R'リリース)のクライアントはサーバーに接続する前にアップグレードをしなければなりません。

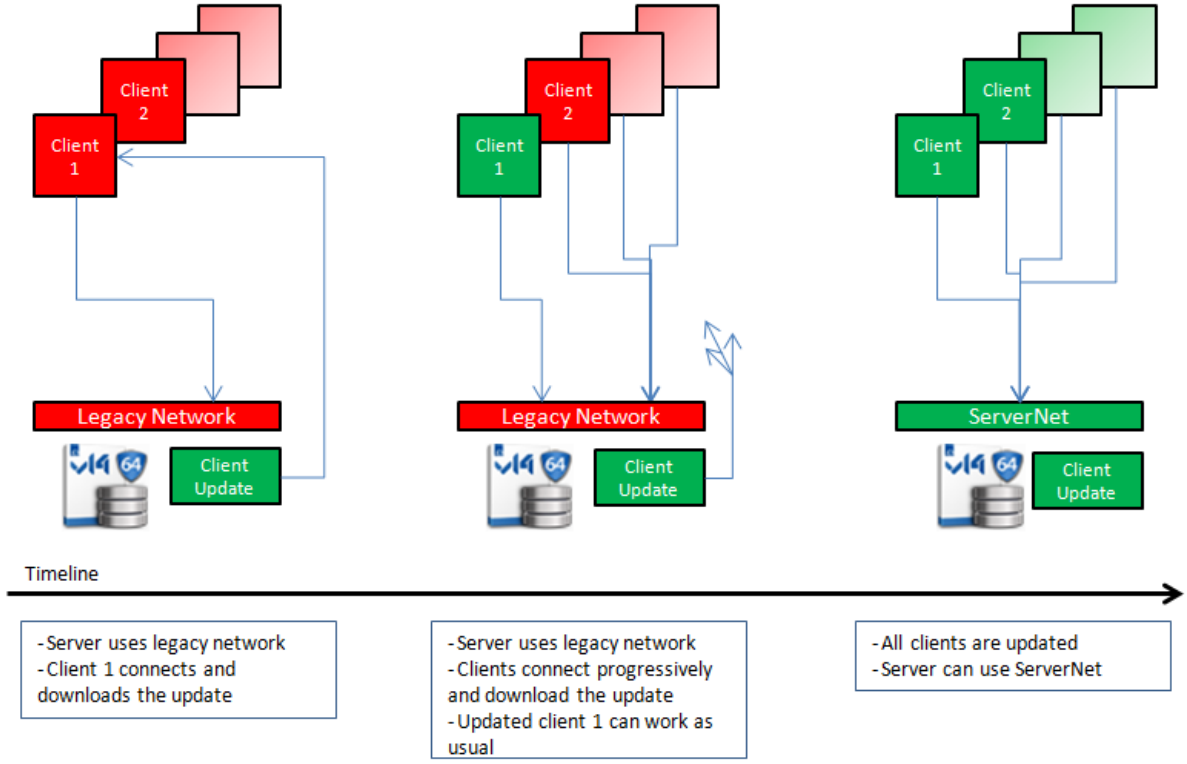
ご自分のアプリケーションがv14 R4以前のバージョンの組み込みクライアントで動いていて、4D Serverの自動機構を使用してアップデートされたクライアントアプリケーションをネットワーク越しに配付したい場合、移行戦略を練る必要があります。 この戦略は以下の原則に則って練る必要があります:

- 互換性のないクライアントは旧式ネットワークレイヤーを使用する4D Server にしか接続することができません。
- アップデートされたクライアントはプロトコルを動的に適応させることができるので、サーバーが使用しているネットワークレイヤーに関わらず4D Server v14 R5以降に接続することができます。

移行戦略は、以下の様な段階を踏む必要があります:

1. 4D v14 R5以降を使用した、アップデートされたクライアントアプリケーションをビルドします。
2. v14 R5 以降の4D Server を、"旧式ネットワークレイヤーを使用"ネットワーク引数を有効化して実行します。
   この設定により、全てのクライアントが接続することができます。
   注: OS X用4D Server の64-bit 版はこのオプションをサポートしていないことに注意して下さい。
3. 全てのクライアントが接続し、新しいバージョンをダウンロードし終わるまで一定時間待ちます。
   これには1日、1週間、あるいはそれ以上の時間がかかる可能性があります。この移行期間中、以前のバージョンのクライアントも、アップデートされたクライアントも、旧式のネットワークサーバーに接続することができます。
4. 全てのクライアントのアップデートが完了したら、旧式のネットワークレイヤーを無効化し、4D Serverを*ServerNet* へと切り替えることができます。

この戦略を図に表すと、以下の様になります:



クライアントリクエストのログを取る

移行プロセスの間、"Diagnostic log recording"ファイルを有効化することが推奨されます。このファイルが有効化されると、4D Serverはそれぞれのクライアントのアップデートリクエストをこのファイルに記録するので、プロセスをモニターすることが出来るようになります。この ログファイルはデフォルトでは有効化されていません。**SET DATABASE PARAMETER** コマンドを、 Diagnostic log recording 定数を1に設定して呼び出す必要があります。

それぞれのアップデートリクエストに対して、以下の情報が記録されます:

- クライアントのIPアドレス
- クライアントのバージョン
- "Update client" イベント

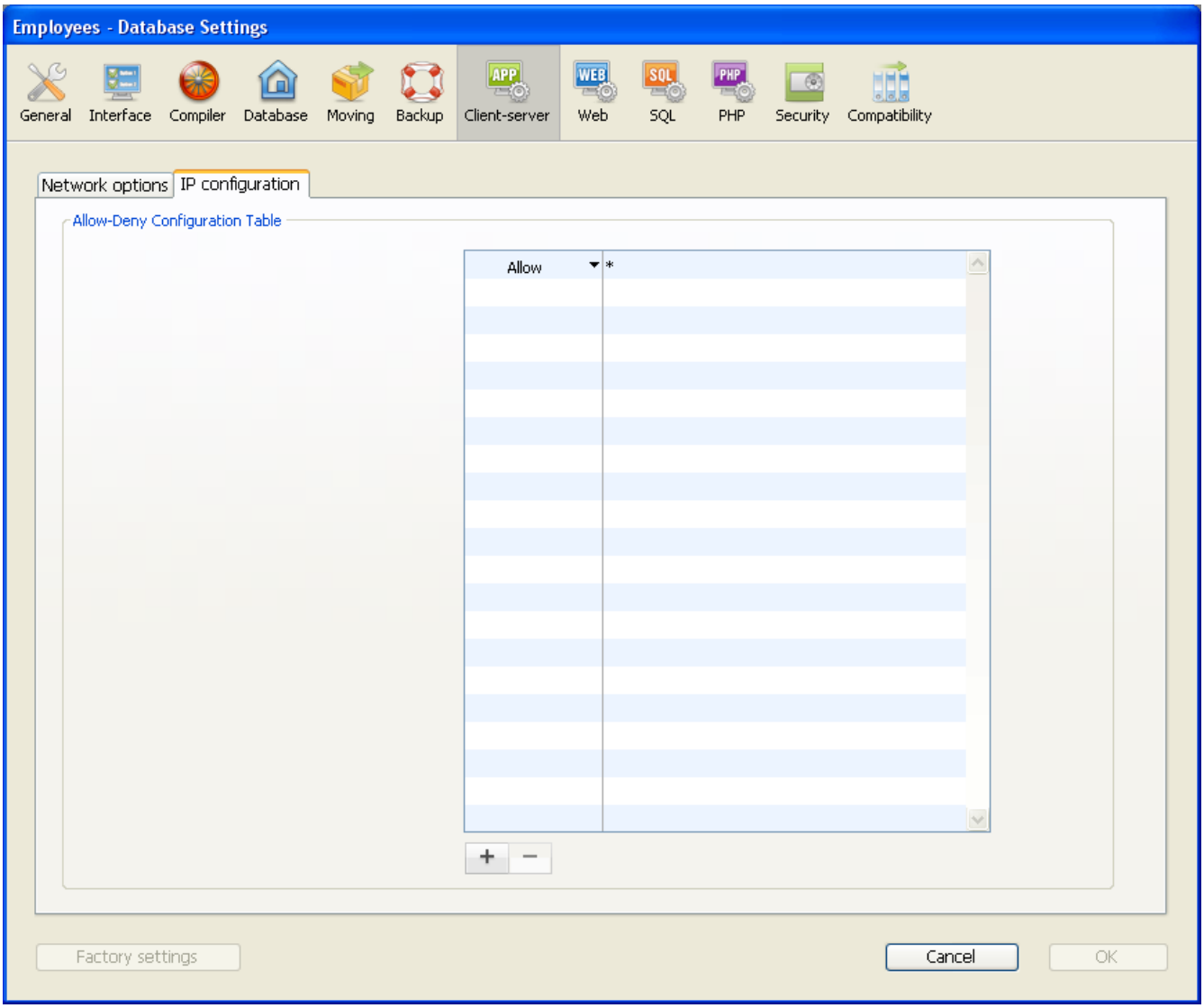
ログファイルをモニタリングすることは、サーバーを*ServerNet* ネットワークレイヤーに切り替えた後も、全てのクライアントが適切にアップデートされたかどうかを確認するために有用です。互換性のないクライアントが接続しようとした場合、サーバーは以下の情報を記録します:

- クライアントのIPアドレス
- クライアントのバージョン
- "Fail to connect" イベント

この場合、例えばクライアントを手動でアップデートするかどうか等を自分で判断することができます。

公開 (環境設定)

データベース設定の**クライアント-サーバー**ページ、"IP設定"タブを使用して、4D Serverデータベースのネットワーク設定に関するパラメターを設定できます (リモート4Dマシンおよび、4D Server両方からアクセス可能):



これらのパラメタについて説明します。

許可-拒否設定表

この表を使用して、4Dリモートマシンの IPアドレスに基づき、データベースへのアクセスコントロールルールを設定できます。このオプションを使用して、例えば戦略アプリケーションなどのセキュリティを高めることができます。

Note: この設定表でWeb接続はコントロールされません。

設定表の動作は以下のとおりです:

- “許可-拒否”列ではポップアップメニューを使用して適用するルールを選択します (許可または拒否)。 ルールを追加するには、追加ボタンをクリックします。新しい行が表に追加されます。削除ボタンで選択した行を削除できます。
- “IPアドレス”列で、ルールに関連するIPアドレスを指定します。アドレスを指定するには、選択した行のセルをクリックし、以下の形式でアドレスを入力します: 123.45.67.89.
  * (アスタリスク) 文字をアドレスの末尾に使用して、範囲を指定することもできます。例えば192.168.* は192.168で始まるすべてのアドレスを示します。
- 表中のルールの順番は重要です。2つのルールが矛盾する設定の場合、より上に設定されているルールが適用されます。

行の順番は列のヘッダをクリックしてソートを行ったり、ドラッグ&ドロップで移動したりして変更できます。

- セキュリティのため、ルールにより明示的に許可されたアドレスのみが接続を許可されます。言い換えれば、表に拒否ルールしか定義されていない場合、許可ルールに適合するアドレスがないため、すべてのアドレスからの接続が拒否されます。特定のアドレスからの接続のみを拒否したい場合 (そして他を許可したい場合)、許可 * ルールを表の最後に追加します。例えば:
  - 拒否 192.168.* (192.168で始まるアドレスを拒否)
  - 許可 * (他のアドレスはすべて許可)

  デフォルトでアクセス制限はありません。最初の行には * (すべてのアドレス) に対して許可が設定されています。

クライアント/サーバ接続の暗号化

4D Server と4Dリモートワークステーションを使って、暗号化モードで通信するクライアント/サーバ接続を構成することができます。
セキュアなクライアント/サーバ通信はSSL (Secured Socket Layer) プロトコルに基づきます。

SSLプロトコルとクライアント/サーバ接続

SSLプロトコルは2 つのアプリケーション間、主にWebサーバーとブラウザーでのデータのやり取りを保護する目的で設計されました。SSLプロトコルは送信者と受信者の認証を行い、やり取りする情報の秘匿性および整合性を保証する目的で作られています。SSLプロトコルの詳細については4D Language Referenceの**SSLプロトコルの使用**を参照してください。

4D Server と4Dクライアントでは、SSLプロトコルで補強されたセキュリティ通信ができます。キーが生成され、4D Serverによって、認証と整合性のメカニズムが透過的に処理されます。ユーザーが追加の設定をする必要はありません。

注: クライアント/サーバーの暗号化を行うと接続が遅くなります。

設定

ネットワークレベルでは、SSL プロトコルはTCP/IP層（ローレベル）とハイレベルなプロトコルの中間に位置します。

“クラシック”なクライアント/サーバアーキテクチャーでSSL を使用するためには、4D Server と各4D クライアントマシン上に *4DSLI.DLL* ファイル (Windows) または *4DSLI.bundle* bundle (Mac OS) が適切にインストールされていることを確認してください。これは、SSL管理専用のSecured Layer Interfaceです。このファイルは以下の場所に置かれます:

- Windows: 4Dおよび4D Serverアプリケーションの実行可能ファイルと同階層
- Mac OS: 4Dおよび4D Serverソフトウェアパッケージの*Native Components* サブフォルダー内

このファイルはデフォルトでインストールされています。

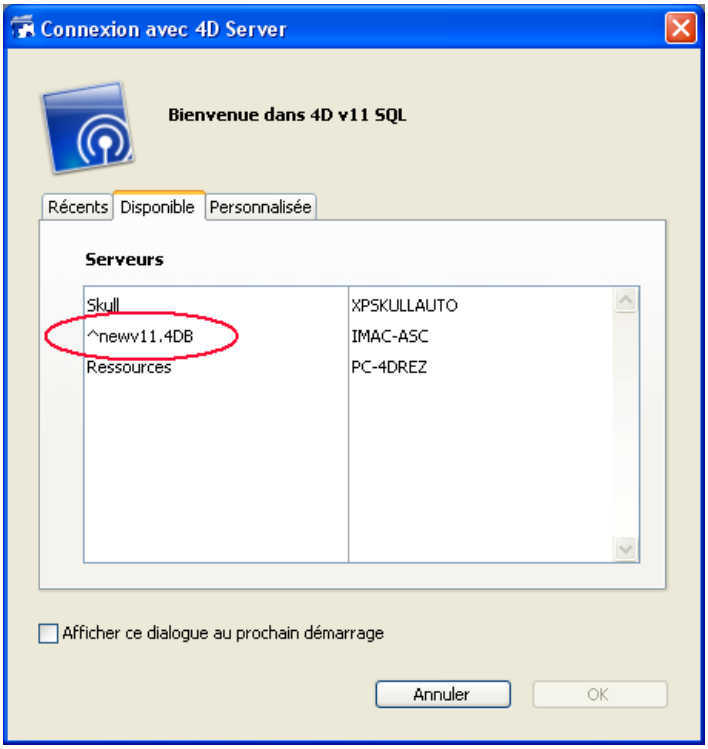
デフォルトで、データベース設定ダイアログボックスの"クライアント-サーバー/ネットワークオプション"ページで、クライアント-サーバー通信エリア内の**クライアント-サーバー接続の暗号化**チェックボックスにチェックします (**設定 (環境設定)**参照)。

デフォルトで、このボックスにチェックは入っていません。
この設定を反映するために、4D Server を終了して再起動してください。

すべての4Dリモートワークステーションが暗号化モードで接続されます。

4Dクライアント接続の暗号化モード

接続ダイアログボックスの利用可ページで、SSLモードで公開されているデータベースの名前の前にはキャレット (^) が表示されます:



注: 接続ダイアログボックスの利用可ページに表示されないデータベースに接続するために、ユーザは**カスタム**ページでIPアドレスと任意の名前を入力します (**4D Serverデータベースへの接続**と**公開 (環境設定)**参照)。この場合セキュアモードで公開されているデータベースに接続するにはデータベース名の前に^をつけます。そうでなければ接続は拒否されます。

リソースフォルダの管理

データベースの*Resources*フォルダを使用して、サーバマシンとすべてのクライアントマシンでカスタムデータ (ピクチャ、ファイル、サブフォルダ等) を共有できます。*Resources* フォルダはサーバマシン上でデータベースストラクチャファイルと同階層になければなりません。

*Resources*フォルダに関連付けられたすべての参照メカニズムは、クライアント/サーバモードでもサポートされています (.lprojフォルダ、XLIFF、ピクチャなど)。この点に関する詳細は4D Design Referenceマニュアルを参照してください。

クライアントはそれぞれこのフォルダのローカルコピーを持ちます。ローカルフォルダの内容は、クライアントがサーバに接続するたびに自動でサーバのそれと同期されます。

さらに、サーバデータベース上の*Resources*フォルダの内容が開発者によりセッション中に更新されたとき、クライアントマシンはその通知を受け取ることもできます。この通知は以下をトリガとして行われます:

- クライアントから最新の更新が行われてから2分後、サーバが自動で行う (この遅延は多数のファイルがコピーされた場合の不適切な通知を避けるためのものです)。
- または更新を行ったクライアントマシン上のリソースエクスプローラのアクションメニューから**クライアントに通知**コマンドを選択する。
- または**NOTIFY RESOURCES FOLDER MODIFICATION** コマンドを使用する。このコマンドは*Resources*フォルダの内容がストアドプロシージャを使用してサーバ上で更新された場合に便利です。

クライアント側では、環境設定のセッション中に"Resources"フォルダを更新の設定に基づき、この通知に対する処理が決定されます。この設定は**SET DATABASE PARAMETER**コマンドを使用して、個々に設定することもできます。3つの選択肢があります: **同期しない**, **自動で同期する** そして **その都度指定**。詳細は**設定 (環境設定)**設定 (環境設定)の節と**SET DATABASE PARAMETER**コマンドの説明を参照してください。
最後に、それぞれのクライアントマシンはいつでもリソースエクスプローラのアクションメニューの**ローカルリソースを更新**コマンドを使用してサーバと同期できます。リソースエクスプローラに関する詳細は4D Design Referenceマニュアルを参照してください。

互換性メモ: 以前のバージョンの4D Serverでは、カスタムデータの交換はストラクチャファイルと同階層に置かれる"Extras"フォルダを使用して行っていました。このフォルダは現在は廃止され、使用は推奨されません。ただし既存のアプリケーションの互換性を保つ目的で当面4D Serverによりサポートされます。

データベースをサービスとして登録

Windowsにおいて、4D Server をサービスとして起動できます。

互換性に関する注意: 4D Server v12より、Mac OS上でのサービスとしての起動はサポートされません。

サービスとして登録された4D Serverアプリケーションはマシンの開始時に自動で、ユーザセッションが開かれる前に、カレントのデータベースを使用して起動されます。またユーザがセッションを終了したときも、データベースは閉じられません。この動作により、マシンの再起動が必要になるような出来事の場合でも、4D Serverが実行されるようになり、メンテナンスをリモートで行うことができます。

Note:

- 64-bit Windowsプラットフォームでは、サービスとして登録された4D Serverアプリケーションはインタフェースなしで実行されます (サーバ管理ウィンドウは表示されません)。
- サービスを管理するメカニズムについては、OSのドキュメントを参照してください。

4D Serverデータベースをサービスとして登録するには、4D Serverの**ファイル**メニューから**現在のデータベースをサービスとして登録**を選択します。マシンが次回に開始されるとき、4D Serverは自動で起動され、カレントのデータベースが開かれます。いくつでもデータベースを登録できます。それぞれのデータベースは一つのインスタンスとしてのみ登録できます。

Note: Windows では、サービス管理機能へのアクセスが制限されている場合、このメニューは選択不可になります。この場合、このコマンドを使用できるようにするには:

- 管理者として4D Serverを実行します。 (これを行うにはアプリケーションアイコン上で右クリックし、コンテキストメニューから**管理者として実行**コマンドを選択します)。 または
- "ユーザー アカウント"コントロールパネルからユーザーアカウント制御設定の変更をクリックし、"通知しない"を選択します。

警告: セッションを開く際には有効なアカウントを使用してください。それ以外の場合にはエラーメッセージが表示されます。デフォルトでは、4D Serverは"ローカル システム アカウント"で実行されており、必ずしもアプリケーションをを使用するために必要な設定になっていないかもしれません。特に、印刷を可能にしたい場合に は、セッションを、デフォルトの印刷設定を持ったアカウントで開く必要があります。アカウントを変更するためには、**コントロールパネル>システムとセキュリティ>管理ツール**>サービスを開いて下さい。サービスのリストの中で、4D Serverを右クリックし、プロパティのオプションを選択し、ログオンタブの中でサーバーを実行する際のアカウントを指定します(この設定は次回の起動時に使用されます)。

データベースの登録を解除するには、4D Serverの**ファイル**メニューから**現在のデータベースの登録解除**を選択します。このコマンドはデータベースがサービスとして登録されていないときは選択できません。

すべての4D Serverデータベースの登録を一度に解除するには、4D Serverの**ファイル**メニューから**すべてのサーバサービスの登録解除**を選択します。このコマンドは4D Serverサービスがどれも有効化されていないときには選択できません。

アプリケーションが起動時にサービスとして起動されているときは、4D Serverからサービスの登録状態を変更することはできません。この場合これらのメニューを選択することはできません。サービスを停止するには、サービスコントロールパネルを使用します。

論理ミラーの設定

4D Serverでは論理ミラーを使用したバックアップシステムの設定を可能とするソリューションを提供します。このソリューションでは新しい2つのコマンド**New log file**と**INTEGRATE LOG FILE**を使用します。

論理ミラーとは

論理ミラーは洗練されたバックアップモードで、クリティカルあるいはハイロードのデータベースで主に使用されます。

論理ミラーは、あるマシンで動作中のデータベースのコピーを別のマシン上に作成し、定期的に更新することで構成されます。両マシンはネットワークを経由して通信を行い、動作中のマシンはデータベースに対して行われたすべての更新を、ログファイルの形で定期的にミラーマシンに送信します。

動作中のデータベースに影響する事故が発生した場合、ミラーデータベースを使用して素早くまたデータを失うことなく、元に戻すことができます。さらに、動作中のデータベースはバックアップによりブロックされることはありません。

なぜ論理ミラーによるバックアップを選択するのか

論理ミラーの使用は特定のニーズに応えるものです。定期的なバックアップとログファイルの使用に基づく標準の方法は、ほとんどの場合簡単で信頼性があり、コストのかからない方法です。データベースは定期的 (通常24時間ごと) にバックアップされます。バックアップ中、データベースには読み込みのみモードでアクセスできます。この一時的な書き込み不可の時間は、2GBを超えるような大きなデータベースであってもとても短いもので、5分もかかりません。この動作を、データベースが利用されていない時間帯に行うよう設定することも可能です。

にもかかわらず、特定の種類の組織、例えば病院などでは、クリティカルなデータベースを24時間完全に動作させなければなりません。データベースをたとえ短時間でも読み込みのみモードにすることはできません。この場合、論理ミラーの設定が適切なソリューションとなります。

Note: ミラーデータベースは**データ**に対して行われた変更のみを反映します。このバックアップモードは開発途中のデータベースには適しません。しばしばストラクチャに対する変更が行われ、それはミラーを無効なものとし、ミラーデータベースストラクチャの交換を必要としてしまいます。

どのように動作するか

論理ミラーを使用したバックアップシステムの設定は、2つの新しいコマンド**New log file**と**INTEGRATE LOG FILE**を使用して行います。これらのコマンドは4D Language Referenceマニュアルで説明しています。

以下の実装を行います:

- データベースはメインの4D Serverマシン (動作マシン) にインストールされ、その全く同じコピーが4D Serverミラーマシンにインストールされます。
- 起動時にアプリケーションのテストを行い (例えば4D Serverアプリケーションのサブフォルダ内で特定のファイルが存在するかどうかをチェックする)、動作マシンかミラーマシンかの区別をして、適切な動作を実行します。
- 動作マシンの4D Serverで**New log file**コマンドを使用して、定期的にログファイルを分割します。メインサーバ上ではバックアップを行いませんので、データベースは常に読み書き可能です。
- 分割されたログファイルをミラーマシンに送信して、**INTEGRATE LOG FILE**コマンドを使用してミラーデータベースに統合します。

このシステムの設定には、コードのプログラムが必要です。特に:

- メインサーバ上で**New log file**コマンドの実行を管理するタイマー。
- 分割されたログファイルを動作マシンからミラーマシンに送信するシステム (4D Internet CommandsによるFTP 、メッセージシステム、Webサービス等)。
- ミラーマシン上で、分割されたログファイルの到着を検知し、**INTEGRATE LOG FILE**コマンドを使用してそれを統合するプロセス。

- メインサーバとミラーサーバとの間の通信とエラー処理システム。

警告: 論理ミラーを使用したバックアップシステムを使用するときは、動作マシン中のマシン上で標準のバックアップを行うことはできません。これら2つのバックアップモードを両方使用すると、動作中のデータベースとミラーデータベースの間で非同期が発生します。動作データベースでは自動マニュアル問わず、バックアップが実行されないようにしてください。他方、ミラーデータベースのバックアップは可能です (後述参照)。

ミラーデータベースのバックアップ

ミラーマシン上の4D Serverで、データベースのバックアップを実行できます。

ミラーマシン上では、どのような方法でもバックアップを実行できます**。ファイル**メニューのコマンドを使用した手動によるバックアップ、データベース設定で設定した定期的なバックアップ、ランゲージコマンドを使用したプログラムによるバックアップ。

注: ミラー・マシンでログファイルを有効にすることはできません。さらに、このマシンで**ログファイルを使用**オプションを選択していないことを確認してください。

動作マシンとの非同期のリスクを避けるため、動作マシンからのログファイルの統合時とミラーデータベースのバックアップ、どちらかの基本的な処理を行うとき、4Dは自動でミラーマシンをロックします。

- ログファイルの統合処理中、バックアップを行うことはできません。**BACKUP**コマンドを使用すると、エラー1417が生成されます (4Dランゲージリファレンスの**バックアップマネージャエラー (1401 -> 1421)**参照)。
- バックアップの実行中は、すべてのプロセスが停止され、ログファイルの統合を起動することはできません。

4D v14以降、カレントのログファイルをミラーマシンでも有効化することが出来るようになりました。これはつまり"ミラーのミラー"、またはそれのさらならミラーを設定できるようになった、という事です。これは**INTEGRATE MIRROR LOG FILE**によって実現可能になったものです。詳細な情報に関しては、このコマンドの説明を参照して下さい。

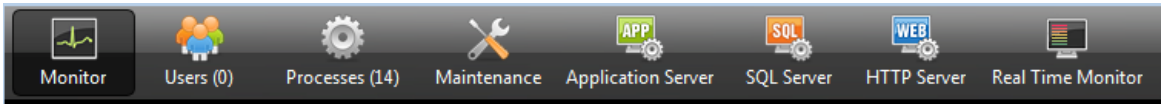
論理ミラーのオペレーションシナリオ

それぞれの4D Serverマシンの視点から見た以下のシナリオで、ミラーを使用したバックアップシステムの設定を示します:

Step	本番環境	ミラーマシン
1	アプリケーションを開始。データファイルをバックアップ。(まだそうしていなければ) ログファイルを有効にする。4Dは MyDatabase.journalファイルを作成。 安全のために、このファイルは別のハードディスクに格納します。 アプリケーションを終了。 (ログファイルを含む) すべてのデータベースファイルをミラーマシンにコピー。	
2	アプリケーションを再起動 (フルバックアップがプログラムされていないことを検証)。実行を開始。	ミラーアプリケーションを開始。4D Serverはカレントログファイルを要求: 本番環境から転送したMyDatabase.journalファイルを選択。ミラーのログを使用したくない場合は、環境設定内のバックアップ/設定ページのカレントログファイルを無効にしてください(**ログファイルを使用**オプションがチェックされていないことを確認)
3	ミラーの更新を決定する (例えば特定の時間経過後)。 **New log file**コマンドを含むメソッドを実行。保存されるファイル名は MyDatabase[0001-0001].journal。 (4DICやWebサービスなど) プログラムを使用してMyDatabase[0001-0001].journalファイルをミラーマシンに送信。 データベース実行。	
4		統合待ちのファイルを検知。**INTEGRATE LOG FILE**コマンドを含むメソッドを実行してMyDatabase[0001-0000].journalファイルを統合。
5	マシン上で事故発生。データファイルが利用不可になる。ミラーマシンへの移行を決定。 MyDatabase.journalカレントログファイルをミラーマシンのいつもの送信先フォルダにコピー。	
6	事故を分析して、修復。	統合待ちのファイルを検知。**INTEGRATE LOG FILE**コマンドを含むメソッ ドを実行してMyDatabase[0001-0001].journalファイルを統合。 念のため、環境設定のバックアップ/設定ページでカレントログファイルを作成する。 データベースが動作。
7	マシンが復旧。データベースファイルをミラーデータベースのものと入れ替える。アプリケーションを開始。4D Serverがログファイルを要求。ミラーデータベースから転送したログファイルを選択する。	データベースを終了。Step2に戻る。

4D Server管理ウィンドウ

4D Serverには、包括的で使いやすい管理ウィンドウが備わっています。
このウィンドウは公開されたデータベースのための解析・管理ツールを提供します。このウィンドウには複数のページがあり、上部のボタンを使用してアクセス可能です:



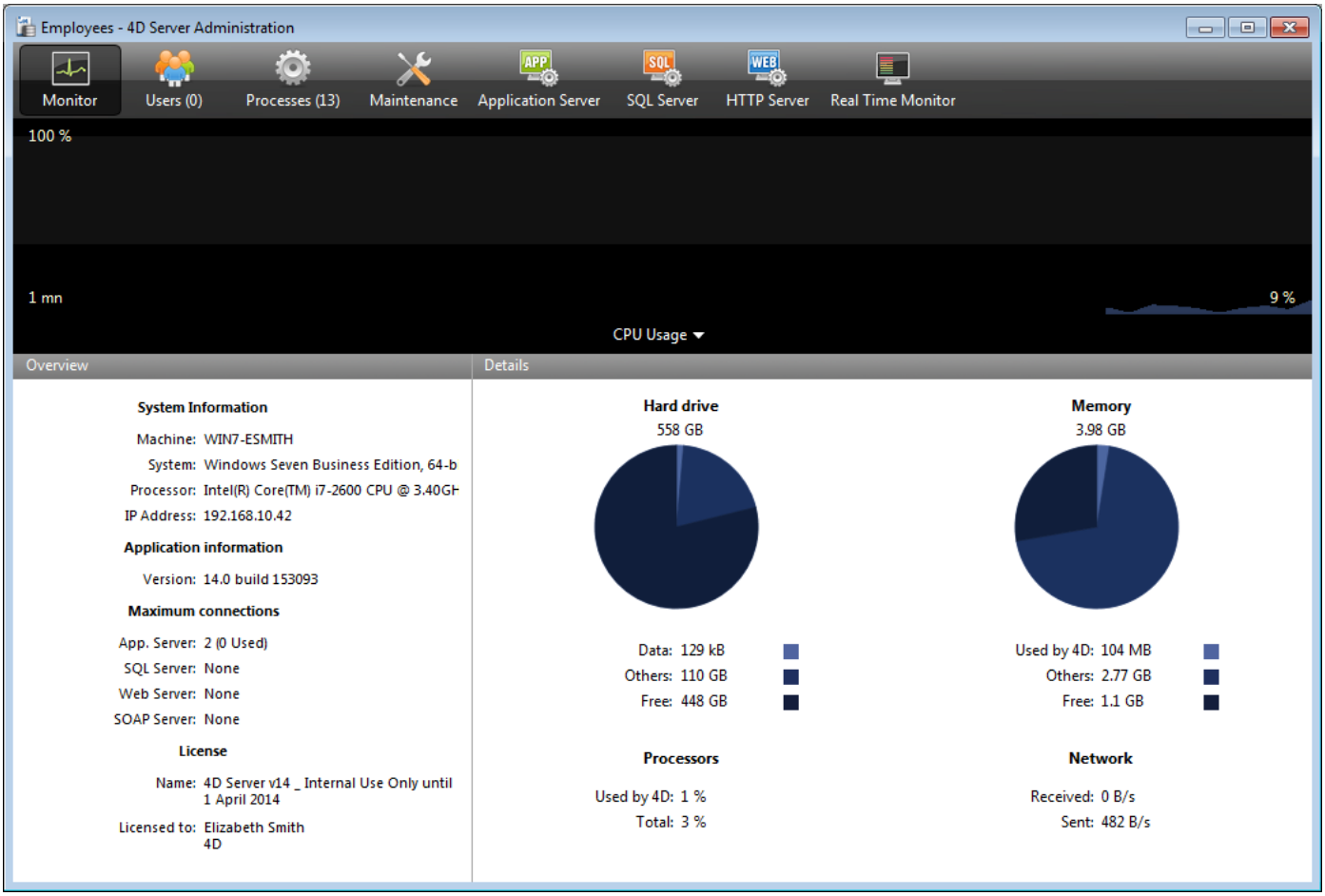
それぞれのページはこの章の各セクションに詳細な説明があります。

注: 管理ウィンドウはリモートの4Dマシンからもアクセス可能です。この点についての詳細は、 **リモートマシンからの管理**のセクションを参照して下さい。

モニタページ
ユーザページ
プロセスページ
メンテナンスページ
アプリケーションサーバページ
SQLサーバページ
HTTPサーバページ
リアルタイムモニター

モニタページ

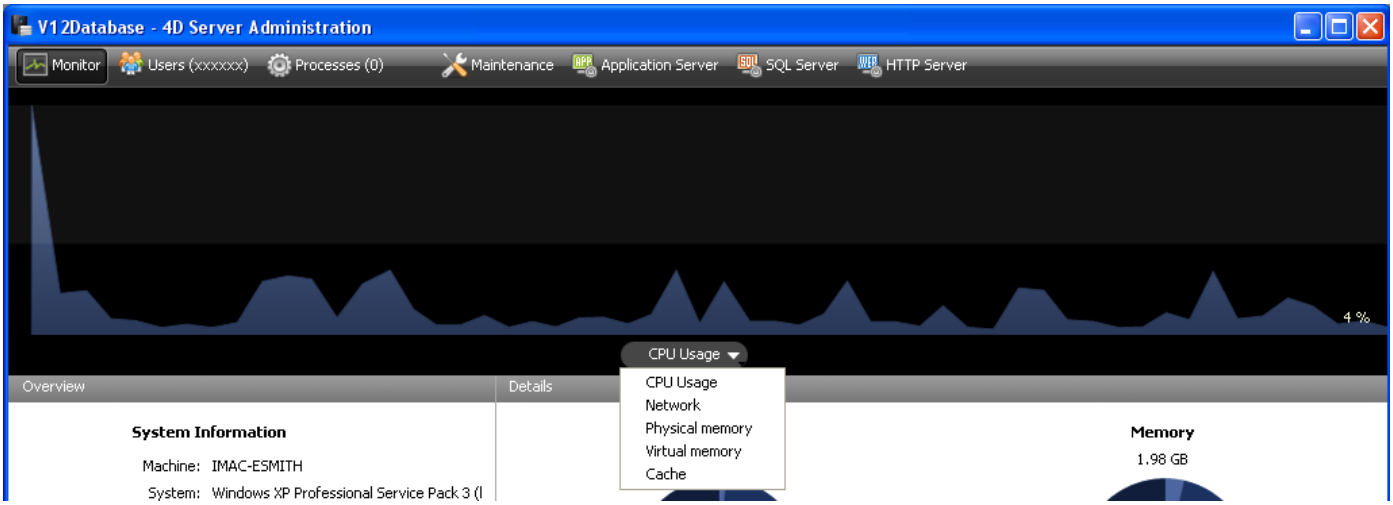
モニタページにはデータベースの利用状況のほか、システムや4D Serverアプリケーションの情報が表示されます:



Note: Windowsでは、表示される情報はセッションを開いたユーザの権限に関連します。詳細は"モニタ情報の表示 (Windows)" の段落を参照してください。

グラフィックエリア

グラフィックエリアではリアルタイムで複数のパラメタ (CPU 利用率、ネットワークトラフィック、およびメモリ) の変化が表示されます。ウィンドウの中央にあるメニューから表示させる内容を選択します:



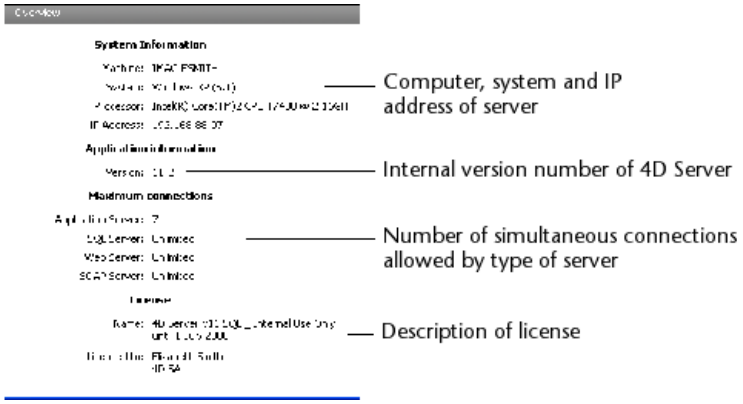
- **CPU利用率**: すべてのアプリケーションによるマシンの全体的なCPU 利用率。

この利用率中4D Server の特定の部分については、"プロセッサ" 情報エリアで提供されます。

- **ネットワーク**: 4D Serverが秒あたりに受信したバイト数。
  4D Serverが送信したバイト数は" ネットワーク" 情報エリアで提供されます。
- **物理メモリ**: 4D Serveが使用する、マシンのRAMの量。
  メモリの利用に関するより詳細な情報は"メモリ" 情報エリアで提供されます。
- **仮想メモリ**: 4D Serverアプリケーションが使用する仮想メモリの量をグラフエリアに表示します。このメモリはアプリケおションのニーズに応じてシステムにより割り当てられます。エリアの右下に表示される値は現在使用されているメモリ量を示します。左上に表示される値は利用可能な仮想メモリの最大値を示します。最大値はアプリケーションの一般メモリ設定に基づき動的に計算されます。
- **キャッシュ**: 4D Serverアプリケーションが使用するキャッシュメモリの量をグラフエリアに表示します。エリアの右下に表示される値は現在使用されているメモリ量を示します。左上に表示される値は、データベース設定に設定されたキャッシュメモリの合計サイズを示します。
  このオプションが選択されている場合、キャッシュの有効な解析を行うために長めの観測時間が必要となるため、グラフエリアのスクロールは遅くなります。

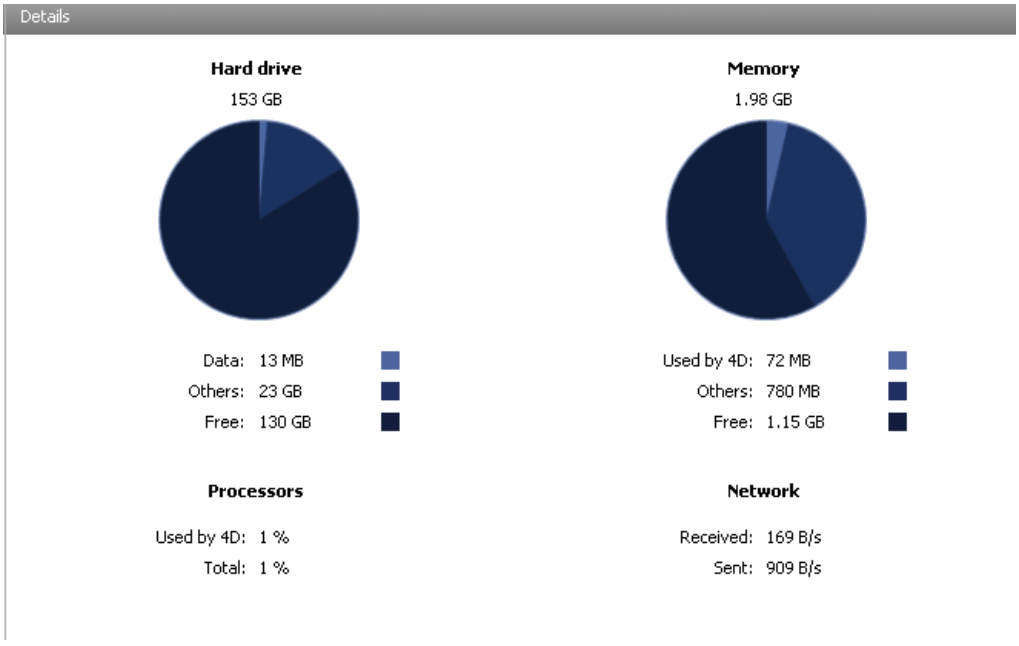
概要エリア

"概要" エリアではシステム、アプリケーション、そして4D Server マシンにインストールされたライセンスに関する様々な情報が提供されます。



詳細エリア

"詳細" エリアはグラフィックエリアで表示されていた情報の一部と、追加の情報を提供します。



- **ハードディスク**: ハードディスク全体およびデータベースデータ (データファイルとインデックスファイル) の使用スペース、他のファイルの使用スペース、空きスペースなどを表示します。
- **メモリ**: マシンにインストールされたRAM メモリ、4D Server の使用量、他のアプリケーションの使用量、そして空き容量。4D Server が使用するメモリはグラフィックエリアにも動的に表示されます。

- **プロセッサ**: 4D Server と他のアプリケーションによる、プロセッサの使用率。この使用率は定期的に再計算されます。4D Server による使用率は、グラフィックエリアにも動的に表示されます。
- **ネットワーク**: ネットワークから4D Server が受信したその瞬間のバイト数、およびアプリケーションが送信したバイト数。この値は定期的に再計算されます。4D Server が受信したバイト数は、グラフィックエリアにも動的に表示されます。

モニタ情報の表示 (Windows)

Windows では、モニタページに表示されるいくつかのシステム情報は、Windows の"Performance Analyzer" ツールを使用して取得しています。これらのツールは、4D Server を起動したセッションを開いたユーザが、必要な認証を得られている場合にのみアクセスできます。このユーザは以下のいずれかでなければなりません:

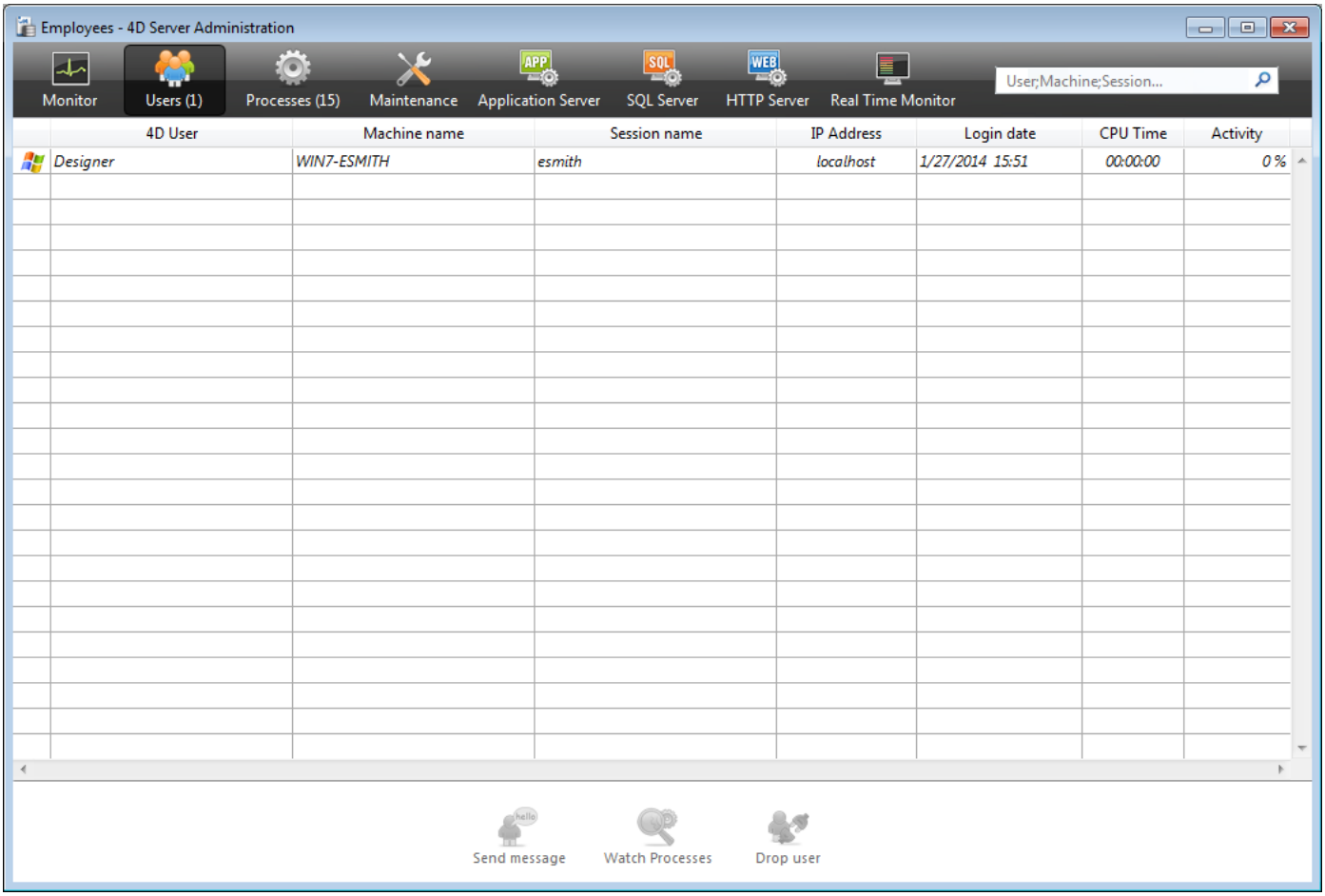
- "Administrators" グループに属する
- Windows Vista: "Power Users" に属する(管理者でないユーザの場合)

Windows Vista で管理者でないユーザを"Power Users"グループに入れるには (この操作を行うには、管理者アカウントを使用しなければなりません):

1. コントロールパネルから"ユーザー アカウント" パネルを開く。
2. "詳細"タブをクリックし、"詳細ユーザ管理"エリアの"詳細"ボタンをクリックする。
   "Local Users and Groups" アプリケーションが実行されます。
3. "Groups" フォルダをダブルクリックする。
4. "Power Users" グループをダブルクリックする。
   "Power Users Properties" という名称のウィンドウが表示されます。
5. **追加...**ボタンをクリックして、ユーザを追加する。
6. "選択するオブジェクト名を入力" テキストエリアに、許可したいユーザ名を入力する。
7. OKをクリックする(2回)。
   "Local Users and Groups" アプリケーションと"ユーザアカウント" を閉じる。

ユーザページ

ユーザページユーザページには、データベースに接続したユーザが表示されます:



"ユーザ"ボタンには、括弧内にデータベースに接続中のユーザ数が表示されます (この番号は、ウィンドウに適用される表示フィルタでは考慮されません)。

このページには検索を行うためのエリアや、コントロールボタンもあります。

ヘッダエリアをドラッグ＆ドロップして、列の順番を入れ替えることができます。またヘッダをクリックすると、リストの値が並び替えられます: 

クリックするごとに昇順/降順が入れ替わります。

ユーザのリスト

データベースに接続したユーザごとに、リストには以下の情報が表示されます:

- システム: クライアントマシンのシステム (Mac OS/Windows)。
- 4Dユーザ: 4Dユーザ名。パスワードシステムが有効になっていない場合、すべてのユーザは"Designer" となります。
- マシン名: クライアントマシンの名前。
- セッション名: クライアントマシン上で開かれたセッション名。
- IP アドレス: クライアントマシンのIPアドレス。
- ログイン日付: クライアントマシンが接続した日付と時刻。
- CPU時間: 接続してからこのユーザが消費したCPU の時間
- Activity: 4D Server がこのユーザのために使用する時間の率 (動的表示)。

検索/フィルタエリア

この機能を使用して、検索エリアに入力されたテキストに対応する行だけをリストに表示させ、行数を減らすことができます。エリアには、どの列に対して検索/フィルタが実行されるかが表示されています。ユーザページでは、4D ユーザ、マシン名、そしてセッション名です。

リストはエリアにテキストが入力されると、リアルタイムで更新されます。

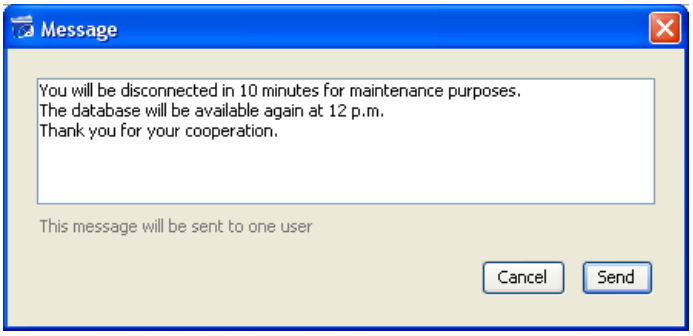
値をセミコロンで区切ることで、一つ以上の値を使用して検索を行うことができます。この場合OR タイプの演算が行われます。

例えば、"John;Mary;Peter" と入力すると、John またはMary またはPeter が上記の列にある行のみが表示されます。

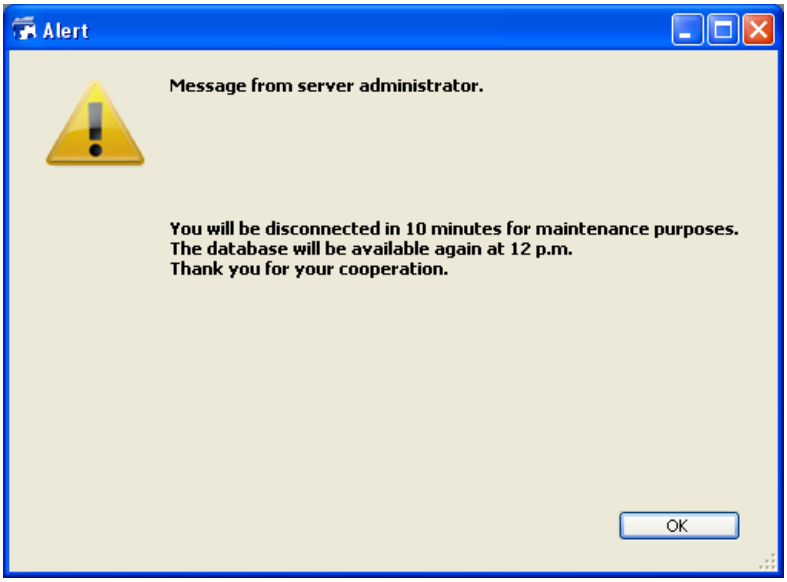
管理ボタン

このページには3 つのコントロールボタンがあります。これらのボタンは最低1 つの行が選択されているときに有効になります。**Shift** キーを押しながらクリックして連続した行を、あるいは**Ctrl** (Windows) / **コマンド** (Mac OS) キーを押しながらクリックして連続しない行を複数選択できます。

- メッセージ送信: このボタンを使用して、ウィンドウで選択した4D ユーザにメッセージを送信できます。ユーザが選択されていないと、ボタンを使用できません。
  ボタンをクリックするとダイアログボックスが表示され、メッセージを入力できます。ダイアログにはメッセージを受信するユーザの数が表示されます:



このメッセージはクライアントマシン上で、警告メッセージとして表示されます:



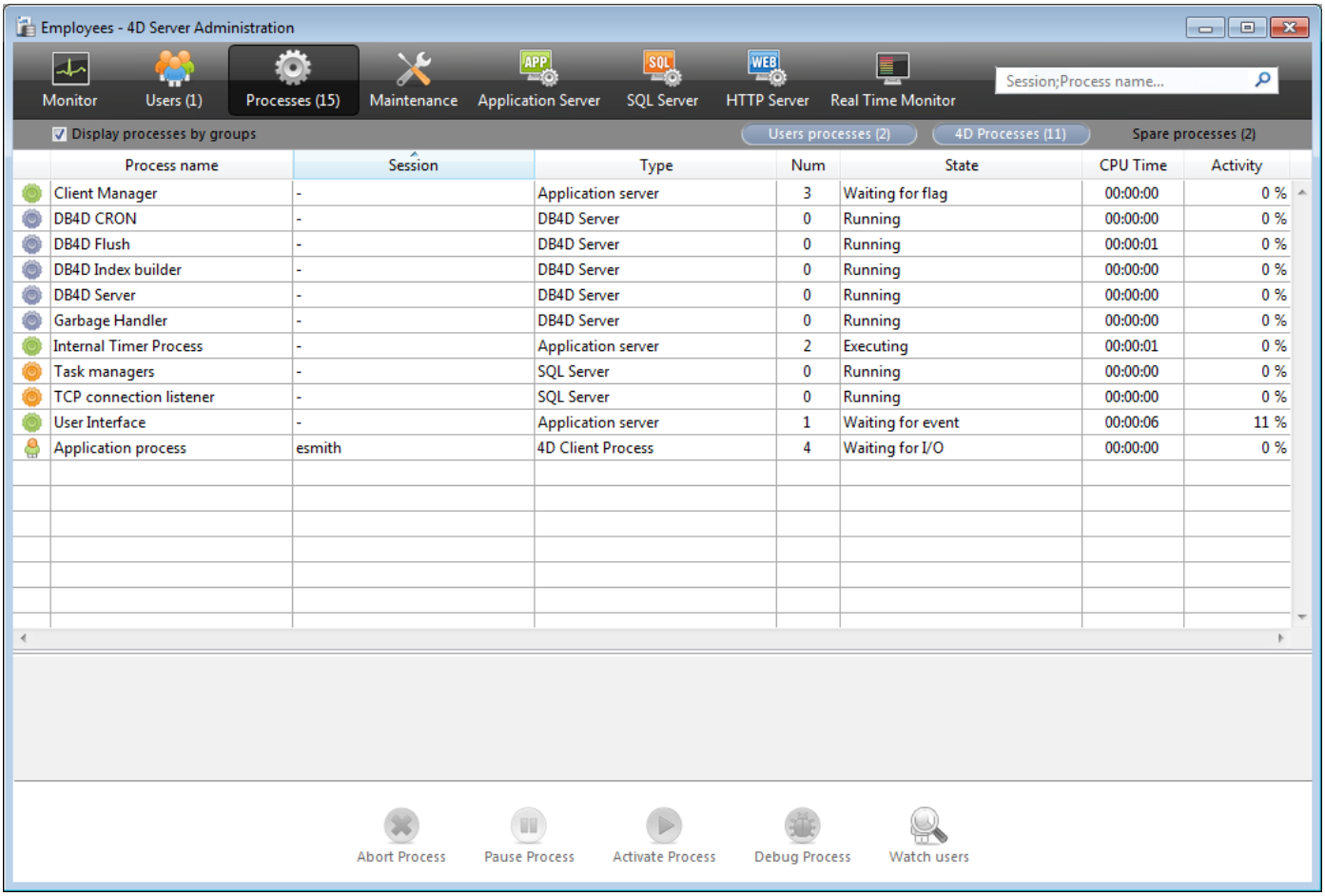
- プロセス表示: このボタンをクリックすると、選択されたユーザのプロセスを直接、管理ウィンドウのプロセス ページに表示させることができます。ボタンをクリックすると、4D Server はプロセスページに移動し、このページの検索/フィルタエリアに選択されたユーザ名を入力します。詳細はこのページの説明を参照してください。

- 接続解除: このボタンは、選択したユーザの接続を強制的に解除するために使用します。
  このボタンをクリックすると警告ダイアログが表示され、接続解除を実行するかキャンセルするか選択できます。

Note: 確認ダイアログを表示させずに選択したユーザの接続を解除できます。これを行うには、**Alt** (Windows) または **Option** (Mac OS) を押しながら、**接続解除**ボタンをクリックします。

プロセスページ

プロセスページには実行中のプロセスが表示されます:



"プロセス"ボタンには、括弧内にデータベースで実行中のプロセス数が表示されます (この番号は、ウィンドウに適用される表示フィルタや**グループ毎にプロセスを表示**オプションのステータスは考慮されません)。

列ヘッダをドラッグ&ドロップして、列の順番を入れ替えることができます。またヘッダをクリックして、値を並び替えることもできます。

ユーザページと同様、このページにも、検索エリアに入力されたテキストに対応する行だけをリストに表示させ、行数を減らすことができる検索/フィルタエリアがあります。検索/ フィルタはセッションおよびプロセス名列に対して実行されます。

ウィンドウに表示されるプロセスをタイプでフィルタするためのボタンが3 つあります:

- ユーザプロセス: ユーザセッションにより、またユーザセッションのために作成されたプロセス。このプロセスには人のアイコンが表示されます。
- 4D プロセス: 4D Server エンジンが生成したプロセス。このプロセスには歯車のアイコンが表示されます。
- 予備プロセス: 使用されていないが一時的に保持され、いつでも再利用が可能なプロセス。このメカニズムは4D Server の反応性を向上させます。このプロセスには薄暗い人のアイコンが表示されます。

グループ毎にプロセスを表示オプションを使用して、4D Server の内部プロセスやクライアントプロセスをグループ化できます。このオプションをチェックすると:

- 4Dクライアントのプロセス (メインの4Dクライアントプロセスや4Dクライアントの基本プロセス、"プロセスタイプ" の段落参照) は1 つにグループ化されます。

- "タスクマネージャ" グループが作成され、タスクを分割するための内部プロセス (共有バランサ、ネットセッションマネージャ、Exclusive pool worker) がグループ化されます。
- "クライアントマネージャ" グループが作成され、クライアントのさまざまな内部プロセスが含まれます。

ウィンドウの下段には選択したプロセスの稼働状況がグラフィカルに表示されます:

Note: **Shift** キーを押しながら連続した行を、**Ctrl** (Windows) / **コマンド** (Mac OS) キーを押しながら非連続の行を選択できます。

プロセスの稼働状況は、4D Server がこのプロセスのために使用した時間のパーセンテージです。
以前のバージョンの4D Server のように、ウィンドウにはプロセスごと以下の情報が表示されます:

- プロセスタイプ (後述)
- セッション (4D プロセスの場合空白、ユーザプロセスの場合4D ユーザ名)
- プロセス名
- プロセス番号 (例えば**New process**関数で返される値)。プロセス番号はサーバ上で割り当てられる番号です。グローバルプロセスの場合、この番号はクライアントマシンで割り当てられた番号と異なる場合があります。
- プロセスの現在の状況
- 作成されてからのプロセスの実行時間 (秒)
- 4D Server がこのプロセスに使用した時間のパーセンテージ

プロセスタイプ

プロセスタイプはアイコンで識別できます。アイコンの色や形により、プロセスタイプは以下のようになります:

- アプリケーションサーバ
- SQL サーバ
- DB4D サーバ (データベースエンジン)
- Web サーバ
- SOAP サーバ
- 保護された4D クライアントプロセス (接続した4D の開発プロセス)
- メイン4Dクライアントプロセス (接続した4D のメインプロセス。クライアントマシン上で作成されたプロセスに対応するサーバプロセス。)
- 4D クライアント基本プロセス (4D クライアントプロセスと並列なプロセス。メイン4D クライアントプロセスをコントロールするプリエンプティブプロセス。)
- 予備プロセス (以前または未来の"4D クライアントデータベースプロセス")
- 4D クライアントプロセス (接続した4D 上で実行しているプロセス)
- ストアドプロシージャ (接続した4D により起動され、サーバ上で実行しているプロセス)
- Web メソッド (4DACTION などにより起動)
- SOAP メソッド (Web サービスにより起動)
- SQL メソッド (SQL クエリにより起動)

Note: **グループ毎にプロセスを表示**オプションがチェックされていると、それぞれの4D クライアントメインプロセスと4D クライアント基本プロセスは一緒にグループ化されて表示されます。

管理ボタン

選択されたプロセスに対して動作する5つのコントロールボタンがあります。ユーザプロセスに対してのみ使用できる点に注意してください。

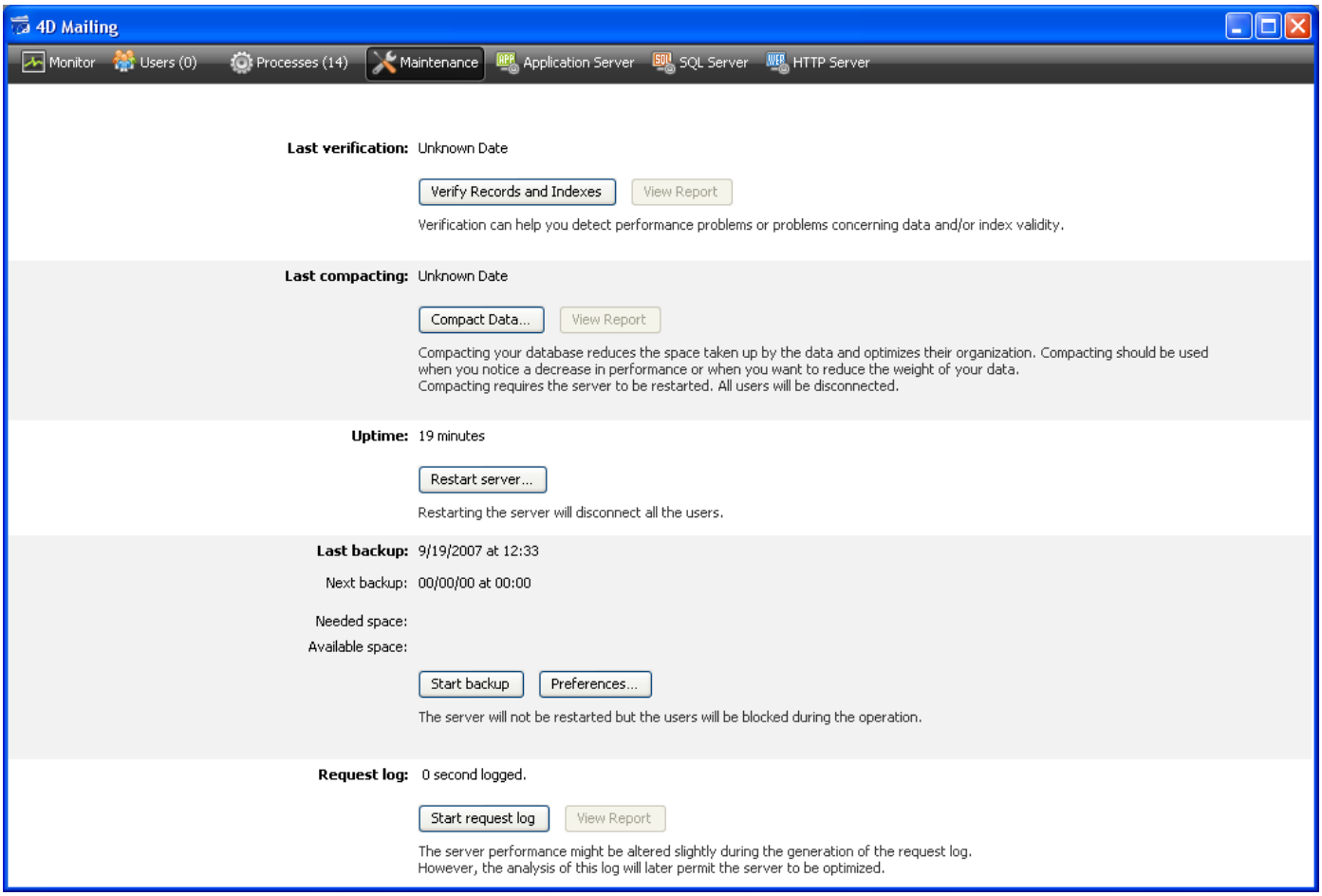
- アボート: 選択したプロセスをアボートします。このボタンをクリックすると、警告ダイアログが表示され、操作を続行またはキャンセルできます。
  Note: 確認ダイアログを表示させずに選択したプロセスをアボートできます。これを行うには、**Alt** (Windows) または

Option (Mac OS) を押しながら、このボタンをクリックします。

- 一時停止: 選択したプロセスを一時停止します。
- 再開: 選択したプロセスの実行を再開します。対象のプロセスは先のボタンを使用するかプログラムにより一時停止状態でなければなりません。そうでなければ、このボタンは効果ありません。
- デバッグ: 選択したプロセスのデバッガをサーバマシン上で開きます。このボタンをクリックすると警告ダイアログが表示され、操作を続行またはキャンセルできます。
  デバッガウィンドウは、4Dコードが実際にサーバマシン上で実行されている場合にのみ、デバッガウィンドウが表示される点に注意してください (例えばトリガのフレームワークや"サーバ上で実行" 属性を持つメソッドの実行時など)。
  Note: 確認ダイアログを表示させずに選択したプロセスのデバッグを開始できます。これを行うには、**Alt** (Windows) または **Option** (Mac OS) を押しながら、このボタンをクリックします。
- ユーザ表示: このボタンをクリックすると、選択されたプロセスのユーザを直接、管理ウィンドウのユーザ ページに表示させることができます。このボタンは最低1 つのユーザプロセスが選択されている場合に有効になります。

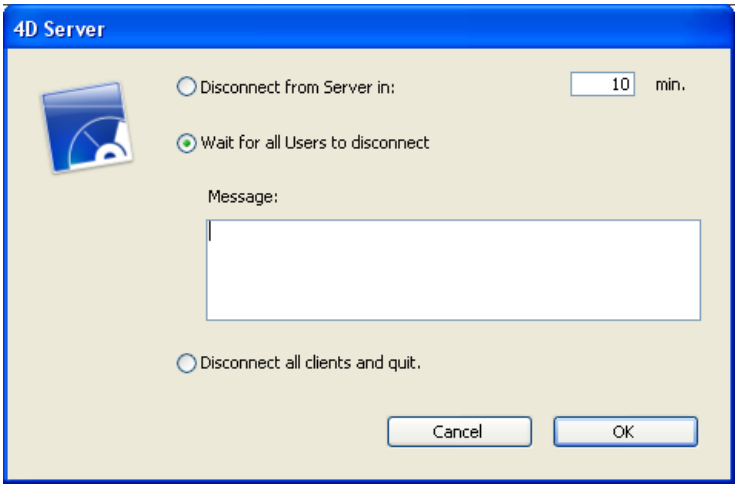
メンテナンスページ

メンテナンスページには、データベースの現在の操作に関する情報が表示されます。また基本的なメンテナンス機能にアクセスすることもできます:



- **最新の検証**: このエリアには、データベース上で実行された最新の検証の日付、時刻、状況が表示されます。データ検証の方法に関する詳細は、デザインリファレンスを参照してください。
  レコードとインデックスを検証ボタンを使用して、サーバを止めることなく、直接検証を起動できます。検証の間、サーバの動作が遅くなるかもしれないことに留意してください。
  データベースのすべてのレコードとすべてのインデックスが検証されます。検証対象を絞り込んだり、追加のオプションを指定したい場合は、Maintenance & Security Center (MSC) を使用します。
  検証後、データベースストラクチャファイルと同階層にあるLogs フォルダに、XMLおよびHTML ファイル形式でログが作成されます。**レポート表示** (クライアントマシンでは **レポートをダウンロード**) ボタンをクリックすると、ブラウザでレポートを参照できます。

- **最新の圧縮**: このエリアには、データベースデータに対して実行された最新の圧縮の日付、時刻、そして状況が表示されます。データ圧縮に関する詳細は、Design Referenceマニュアルを参照してください。
  データ圧縮... ボタンを使用して、圧縮処理を起動できます。この操作を行うためには、サーバを停止させる必要があります。ボタンをクリックすると、4D Server データベースの終了ダイアログ表示され、操作を選択することができます:



このダイアログボックスに関する詳細は、**4D Serverを終了する**を参照してください。

データベースが実際に停止された後、4D Server はデータベースデータに対する標準の圧縮操作を行います。追加のオプションを指定したい場合は、Maintenance & Security Center (MSC) を使用します。

圧縮が終了すると、4D Server は自動でデータベースを再開し、4D ユーザの再接続が可能になります。

Note: 圧縮リクエストが4D クライアントマシンからなされた場合、このマシンは自動で4D Server により再接続されます。

データベースストラクチャと同階層にあるLogs フォルダに、XMLおよびHTMLフォーマットでレポートファイルが作成されます。**レポート表示** (クライアントマシンでは**レポートをダウンロード**) ボタンをクリックすると、ブラウザでレポートを参照できます。

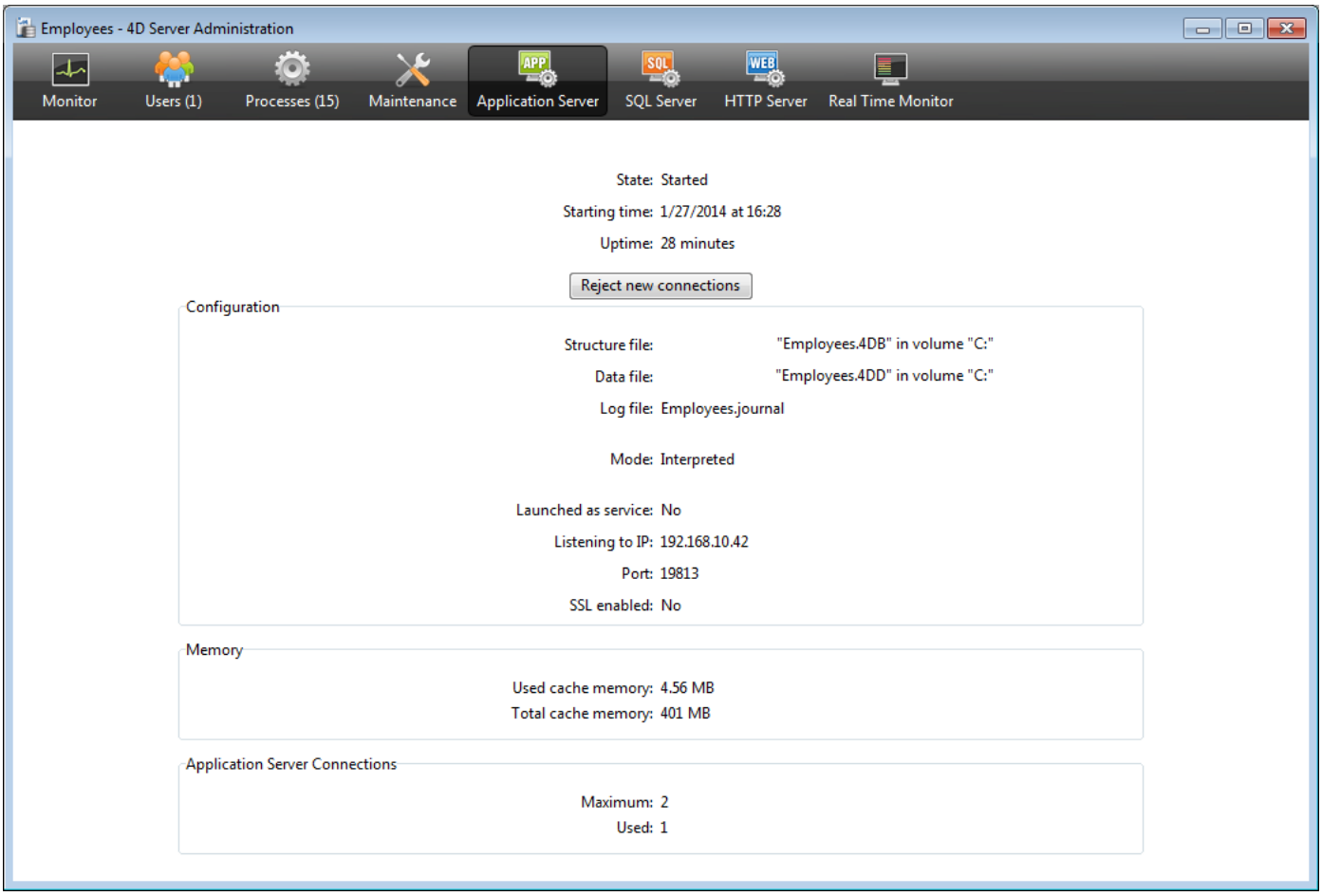
- **稼働時間**: このエリアには、サーバが開始されてからの稼働時間 (日、時、分) が表示されます。
  サーバを再起動... ボタンをクリックすると即座にサーバを再起動できます。このボタンをクリックすると、4D Server データベースの終了ダイアログが表示され、操作をどのように中断するか選択できます (**4D Serverを終了する**を参照)。再起動後、4D Server は自動でデータベースを再度開き、4D ユーザの再接続が可能になります。
  Note: 再起動リクエストが4D クライアントマシンからなされた場合、このマシンは自動で4D Server により再接続されます。
- **最新のバックアップ**: このエリアにはデータベースの最新のバックアップの日付と時刻と、環境設定の"スケジュール"で設定された、次回に予定されるバックアップに関する情報が表示されます。
  - 次回のバックアップ: 次回の自動バックアップの日付と時刻。
  - 必要空き容量: バックアップに必要な計算された空き容量。バックアップファイルの実際のサイズは(圧縮などの)設定や、データファイルの変化により変わります。
  - 空き容量: バックアップボリュームの空き容量。
    バックアップ開始を使用して、現在のバックアップパラメタ (バックアップするファイル、アーカイブの場所、オプションなど) を使用したバックアップを開始できます。**環境設定...** ボタンをクリックして、これらのパラメタを確認できます。サーバ上でのバックアップの間、クライアントマシンはブロックされ (ただし接続解除はされません)、新規のクライアント接続はできなくなります。
- **リクエストログ**: このエリアには、リクエストのログを記録する期間が(それが有効な場合)表示されます。
  - リクエストログファイルには、Webリクエストを除く、サーバが受信したリクエストに関する情報(時間、プロセス番号、ユーザ、リクエストサイズ、処理時間等、サーバの動作を検証するために使用可能な情報)が格納されます。このファイルは4DRequestsLog_X (X はファイルのシーケンス番号) という名前で作成され、データベースのLogsフォルダに格納されます。サイズが10 MB に達するとファイルは閉じられ、新しいファイルが生成されます。このときシーケンス番号はインクリメントされます。
  - デバッグイベントファイルは“4DDebugLog.txt”という名前のファイルにメソッド、4Dコマンドまたはプラグインコマンドのそれぞれの実行を保存します。このファイルはストラクチャーファイルの隣にある**Logs**のフォルダのサブフォルダに自動的に保存されます。それぞれのイベントは実行の前に系統的に記録されるので、アプリケーションが予期せず終了した場合でも確実にファイルに記録を残すことが出来ます。このファイルはアプリケーションを起動するたびに消去・上書きされることに注意して下さい。このファイルについては **SET DATABASE PARAMETER** コマンドによって設定を変更することが出来ます。

  リクエストログ開始ボタンを使用すると、新しいファイルが作成され、リクエストの記録が有効になります。記録が開始されるとフォーマンスが著しく低下するため、これはアプリケーションの開発フェーズでのみ使用します。
  リクエストのログが有効になると、ボタンのタイトルが**リクエストログ停止**に変わり、いつでもリクエストの記録を停止できます。リクエストのログ記録を再開すると、以前のファイルは消去されることに留意してください。
  Note: **SET DATABASE PARAMETER** コマンドを使用してプログラムでログの開始/停止ができます。
  レポート表示 (クライアントマシンでは**レポートをダウンロード**) ボタンをクリックすると、システムウィンドウが開い

て、リクエストログファイルが表示されます。

アプリケーションサーバページ

アプリケーションサーバページには4D Server が公開しているデータベースについての情報がまとめられていて、公開を管理できます:



ステータス情報

ページの上部には、4D Server アプリケーションサーバの現在の状況が表示されます。

- 状態: 開始または停止状態
- 開始時刻: サーバデータベースの起動日と時刻。これは、データベースが4D Server によって開かれた日付です。
- 稼働時間: データベースが最後に開かれた時からの経過時間。

新規接続許可/拒否ボタン

このボタンは入れ替わり表示され、アプリケーションサーバへの新しいクライアントの接続を管理するために使用できます。

- データベースが公開された時、デフォルトで:
  - ボタンのラベルは"新規接続を拒否"
  - ライセンスが許可する限り、新規クライアントは自由に接続が可能
  - データベース名は接続ダイアログに公開される ("開始時に接続ダイアログにデータベース名を公開"オプションが環境設定で有効になっている場合)
- **新規接続を拒否**ボタンをクリックすると:
  - ボタンラベルが "新規接続を許可" に変化
  - 新規クライアントは接続不可

- データベース名が接続ダイアログに表示されなくなる
- すでに接続済みのクライアントは接続解除されず、通常通りに操作が可能
- **新規接続を許可**ボタンをクリックすると、データベースはデフォルトに戻ります。

この機能は例えば、サーバ開始直後に管理者が様々なメンテナンス操作 (検証や圧縮など) を行うことを可能にします。管理者がクライアント接続を使用する場合、この機能により、一つのクライアントだけがデータを更新できることを確実にできます。またクライアントマシンがまったく接続されていない状態で行わなければならないメンテナンス操作の準備のために、この機能を使用することができます。

設定

このエリアには、サーバが公開する4D データベースについての情報 (名前、データやストラクチャファイルの場所、データベースログファイルの場所) が表示されます。ストラクチャやデータファイル名をクリックすると、完全なパス名を表示させることができます:

"モード" フィールドには、データベースの現在の実行モード 、コンパイル済みかインタプリタかが表示されます。

エリアの下部には、サーバ設定パラメタ (サービスとして起動、ポート、IP アドレス) やクライアント/ サーバ接続用のSSL (SQLやWeb 接続は除く) の状態が表示されます。

メモリ

このエリアには**総キャッシュメモリ** (データベース環境設定で設定されたパラメタ) と**使用キャッシュメモリ** (必要に応じて 4D Server が動的に割り当て) が表示されます。

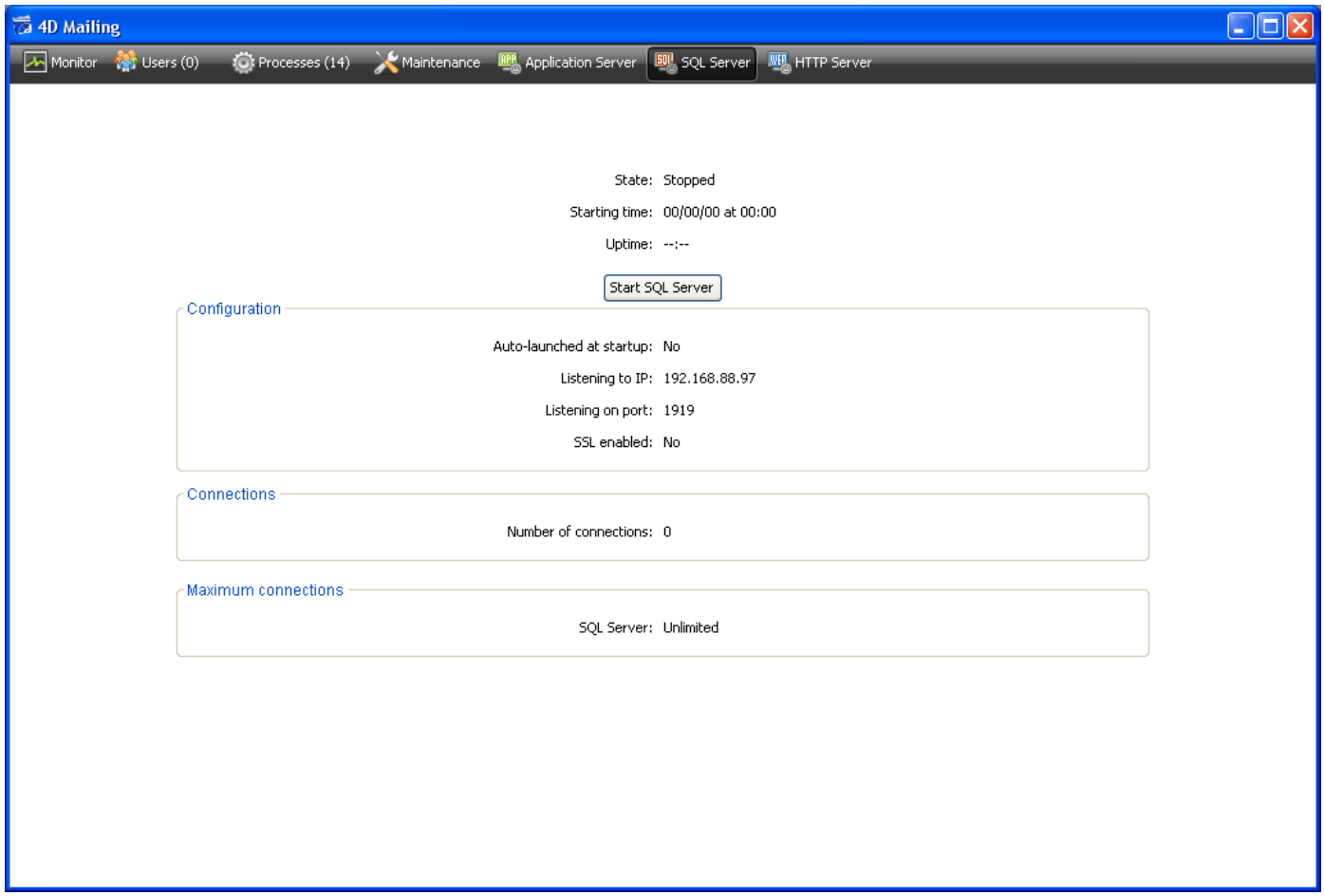
アプリケーションサーバー接続

"最高:"はアプリケーションサーバーに許可された同時クライアント最大接続数を表します。この値はサーバーマシンにインストールされているライセンスによります。

"使用中:"は現在使用中の実際の接続数を表します。

SQLサーバページ

SQLサーバページには、4D Server に統合されたSQLサーバについての情報が集められています。またSQLサーバを有効にするためのボタンも含まれています:



状況

ページの上部には、4D Server のSQLサーバの現在の状況が表示されます。

- 状態: 稼働中または停止中
- 開始時刻: SQL サーバが起動された日付と時刻。この値は、SQLサーバが4D Server の起動時に開始されていない場合、アプリケーションサーバの起動時刻とは異なることがあります。
- 稼働時間: SQL サーバが開始されてからの経過時間。

SQLサーバ開始/停止

このボタンは交互に表示され、4D Server SQLサーバをコントロールするために使用します。

- SQL サーバの状態が" 稼働中" の場合、ボタンのタイトルは**SQLサーバ停止**になります。このボタンをクリックすると、4D Server SQLサーバは即座に停止します。指定したTCP ポートで受信した外部からのSQL クエリには応答しなくなります。
- SQL サーバの状態が"停止中" の場合、ボタンのタイトルは**SQLサーバ開始**になります。このボタンをクリックすると、4D Server SQLサーバは即座に開始します。指定したTCP ポートで受信した外部からのSQL クエリに応答します。4D SQL サーバを使用するには、適切なライセンスが必要な点に注意してください。

Note: 環境設定で設定してアプリケーションの起動時に、またはプログラムを使用して、SQLサーバを自動で開始できます。

設定

このエリアには、SQLサーバ設定のパラメタ (起動時の自動開始、待ち受けIP アドレス、TCP ポート (デフォルトで 19812)、そしてSQL 接続用のSSL (4D やWeb 接続を除く)) が表示されます。

これらの値は4D の環境設定で変更できます。

接続

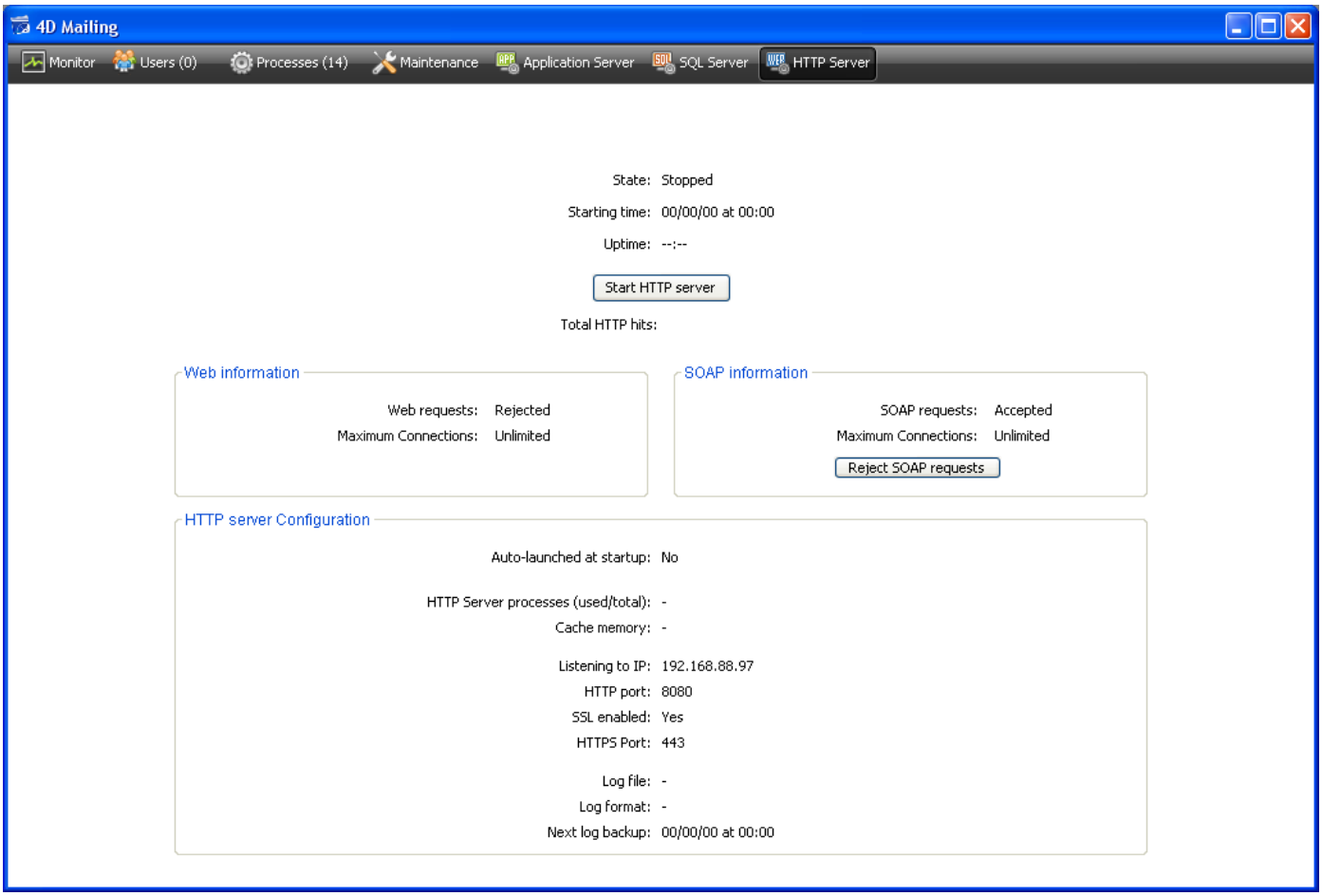
4D Server上で現在開かれているSQL 接続の数。

最大接続数

許可される同時SQL 接続最大数。この値はサーバマシンにインストールされたライセンスに基づきます。

HTTPサーバページ

HTTPサーバページには4D Server のWebサーバやSOAP サーバに関する情報が集められています。Webサーバは、HTMLページやピクチャなどのWebコンテンツの公開を可能にします。SOAPサーバはWebサービスの公開を管理します。 これら2つのサーバは、4D Serverの内部的なHTTPサーバに依存しています。このページにはまたこれらのサーバをコントロールするためのボタンが含まれます:



状況

ページの上部には4D Server のHTTP サーバの現在の状態についての情報が表示されます。

- 状態: 稼働中または停止中
- 開始時刻: HTTP サーバが起動された日付と時刻。この値は、HTTPサーバが4D Server の起動時に開始されていない場合、アプリケーションサーバの起動時刻とは異なることがあります。
- 稼働時間: HTTP サーバが開始されてからの経過時間。
- 総HTTP ヒット数: HTTP サーバが開始されてから、サーバが受信したローレベルのHTTP ヒット数。

HTTP サーバ開始/停止

このボタンは交互に表示され、4D Server HTTP サーバをコントロールするために使用します。

- HTTP サーバの状態が" 稼働中" の場合、ボタンのタイトルは**HTTP サーバ停止**になります。このボタンをクリックすると、4D Server HTTP サーバは即座に停止します。Web サーバとSOAP サーバはリクエストを受け付けなくなります。
- HTTP サーバの状態が" 停止中" の場合、ボタンのタイトルは**HTTP サーバ開始**になります。このボタンをクリックすると、4D Server HTTP サーバは即座に開始します。Web とSOAP のリクエストが受け入れられます。SOAP サーバは別途停止できることに留意してください。

Notes:

- HTTP サーバを開始するには適切なライセンスが必要です。
- 環境設定で設定してアプリケーションの起動時に、またはプログラムを使用して、HTTP サーバを自動で開始できます。

Web情報

このエリアには4D Server のWeb サーバに関する情報が表示されます。

- Web リクエスト: 許可または拒否。 この情報はWebサーバが有効かどうかを示します。Web サーバは直接HTTP サーバにリンクしているため、Web リクエストはHTTP サーバが開始されていれば受信され、停止されていれば拒否されます。
- 最大接続数: 許可されるWeb 接続最大数。この値はサーバマシンにインストールされたライセンスに基づきます。

SOAP 情報

このエリアには4D Server のSOAP サーバに関する情報が表示され、またコントロールボタンが含まれます。

- SOAP リクエスト: 許可または拒否。 この情報はSOAP サーが有効かどうかを示します。SOAP リクエストを受け入れるためには、HTTP サーバが開始され、かつSOAP サーバが明示的にリクエストを受け入れなければなりません (許可/拒否ボタンの説明参照)。
- 最大接続数: 許可されるSOAP 接続最大数。この値はサーバマシンにインストールされたライセンスに基づきます。
- **SOAPリクエスト許可/拒否**ボ タン: このボタンは交互に表示され、4DServer のSOAP サーバをコントロールするために使用します。このボタンをクリックすると、環境設定の"Web サービス/SOAP" ページのSOAPリクエストを許可設定が変更されます。また環境設定が変更されれば、このボタンのラベルも変わります。
  HTTP サーバ停止中に**SOAP リクエスト許可**ボタンをクリックすると、4D は自動でHTTP サーバを開始します。

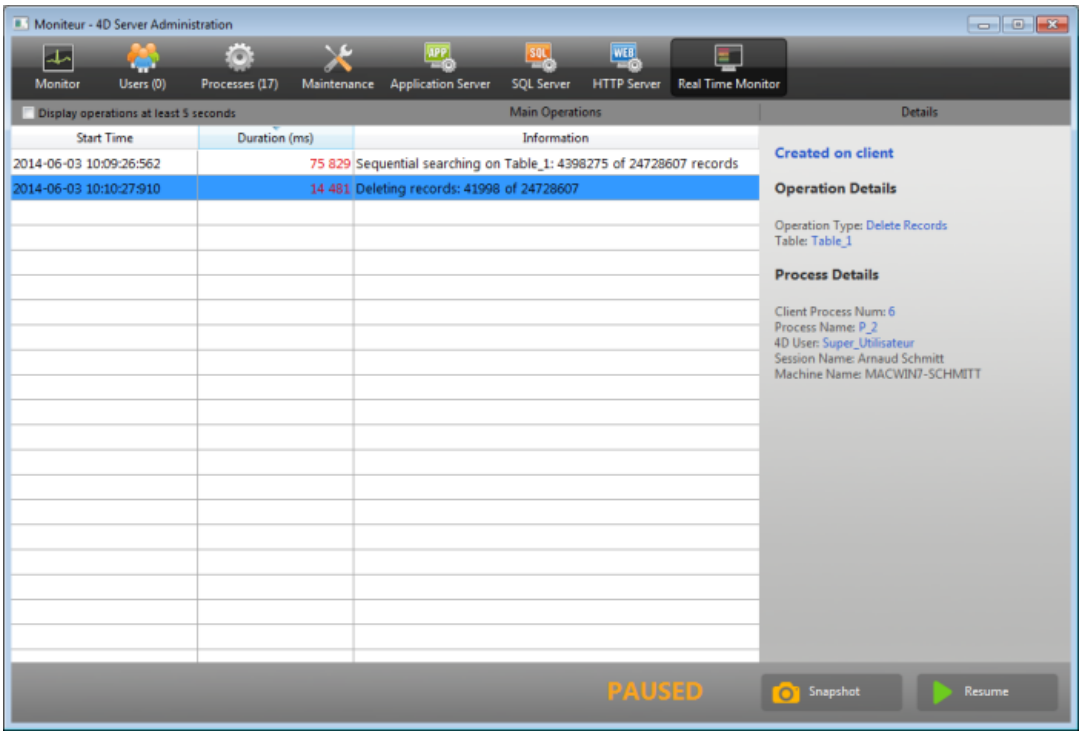
HTTP サーバ設定

このエリアにはHTTP サーバの設定パラメタや動作に関する情報が表示されます。

- 起動時にHTTP サーバを自動で開始: 4D Server の環境設定で設定されたパラメタ。
- HTTP サーバプロセス (使用済み/ 総計): サーバ上で作成されたHTTP プロセス数 (現在のプロセス数 / 作成されたすべてのプロセスの総数)。
- キャッシュメモリ (使用済み/ 総計): HTTP サーバキャッシュメモリサイズ (キャッシュが実際使用しているサイズ / 環境設定で理論的にキャッシュに割り当てられた最大サイズ)。
- 待ち受けIP、TCP ポート (デフォルトは80)、(4D とSQL 接続を除く) HTTP 接続用のSSL 有効、そしてHTTPS ポート。環境設定のWeb/ 設定ページで設定された、HTTP サーバの現在の設定 (4D Language Referenceの**ABORTQR DELETE COLUMN**参照)。
- ログファイル情報: 場所、フォーマット、そしてHTTP サーバの次回の自動ログバックアップの日付 (logweb.txt ファイル)。

リアルタイムモニター

リアルタイムモニターは、アプリケーションによって実行された、「長い」オペレーションの操作をリアルタイムでモニターします。これらのオペレーションとは、例えばシーケンシャルクエリや4D式の実行などです:



このページは、サーバーマシンの管理ウィンドウの中にありますが、リモートの4Dマシンからも見られます。リモートマシンの場合は、サーバーマシン上で実行されてる操作からのデータを表示します。

データ上で実行されている長い処理は、それぞれ行が割り当てられます。この行は、操作が完了すると消えます(**オペレーションを最低5秒間表示する**option オプションをチェックすることで早いオペレーションを5秒間表示したままにする事ができます。以下を参照して下さい)。

行の中には、以下の情報が行ごとに表示されています:

- **開始時刻**: 操作の開始時刻が、 "dd/mm/yyyy - hh:mm:ss" というフォーマットで表示されます。
- **経過時間 (ms)**: 進行中の操作の経過時間がミリ秒で表示されます。
- **情報**: 操作の説明
- **サブオペレーション**: 選択したオペレーションに依存しているオペレーションが表示されます(サブオペレーションを含んでいる行は太字で表示されます)。
- **詳細**: このエリアには選択したオペレーションのタイプに応じて、その詳細な情報が表示されます。具体的には以下の情報が表示されます:
  - **作成された場所**: そのオペレーションがクライアントアクションの結果(クライアントで作成された)のか、サーバー上でストアドプロシージャ―、または"Execute on server" オプションを使用したのか(サーバーで作成された)を表示します。
  - **オペレーション詳細**: オペレーションタイプと、(クエリオペレーションに対しては)クエリプランを表示します。
  - **サブオペレーション**(あれば): 選択したオペレーションに従属するオペレーションを表示します(例:親レコードの前にリレートするレコードを削除する)
  - **プロセス詳細:** テーブル、フィールド、プロセスまたはクライアントに関する情報が表示されます。オペレーションのタイプによって異なります。

このページは表示されてからすぐにアクティブになり、恒久的に更新され続けます。ただし、この操作はアプリケーションの実行を極端に遅くさせる可能性があることに注意して下さい。以下の方法を用いて更新を一時的に停止させることが可能です:

- **停止**ボタンをクリックする

- リストの中をクリックする
- スペースバーを押す

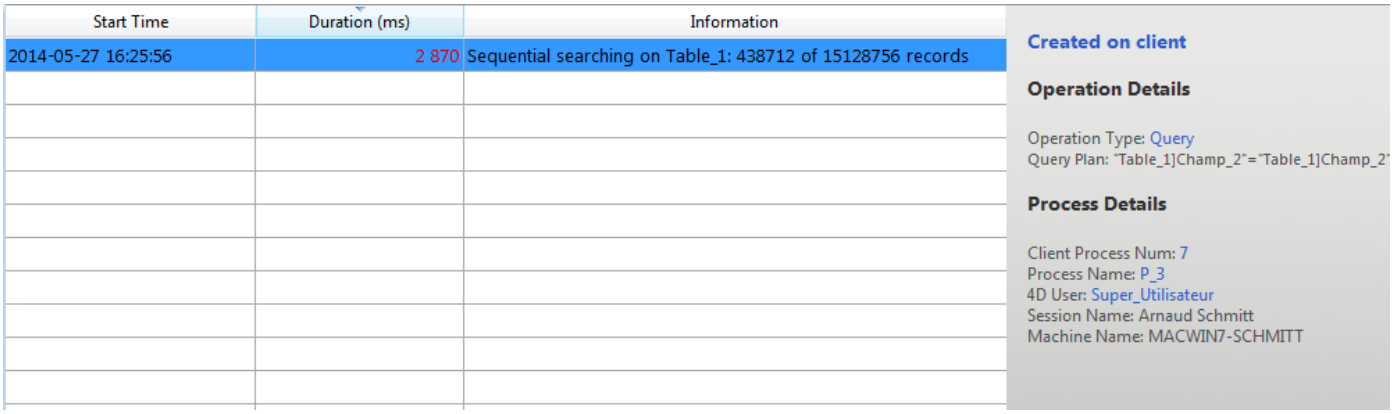
ページを停止させると、一時停止のメッセージが表示され、ボタンの表示が「**再開**」へと変わります。
モニタリングを停止する操作と同じ操作をすることでモニタリングを再開させることができます。
注: リアルタイムモニタリングは、**GET ACTIVITY SNAPSHOT** コマンドを使用して行うこともできます。

アドバンスドモード

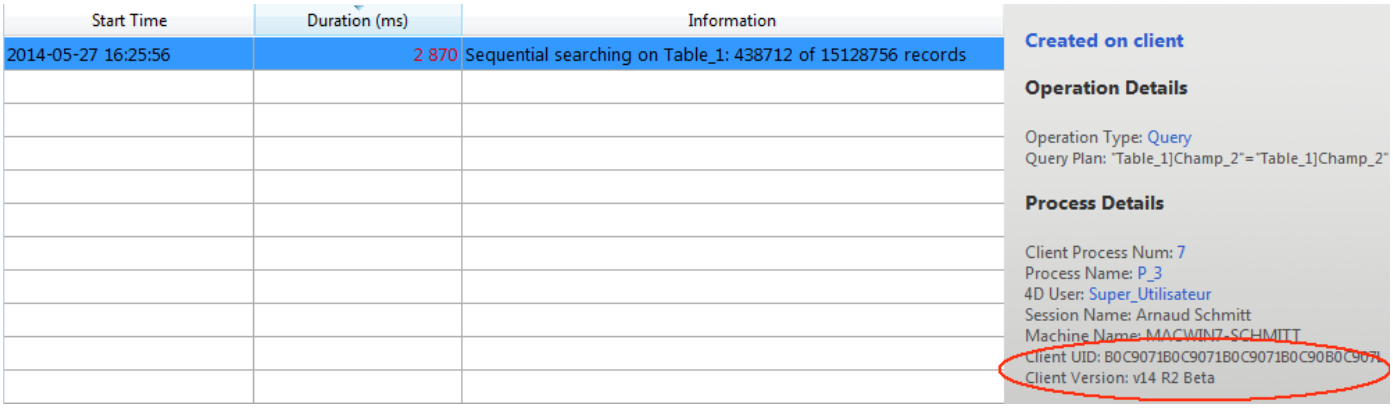
必要であれば、RTMページは追加の情報をオペレーションごとに表示することができます。

オペレーションのアドバンスドモードへとアクセスるには、**Shift**を押しながら情報を取得したいオペレーションを選択して下さい。閲覧可能な情報が全て、フィルタリングなしで"プロセス詳細"エリアに表示されます(**GET ACTIVITY SNAPSHOT** コマンドで返されるものと同様です)。表示される情報は選択したオペレーションによって異なります。

標準モードで表示される情報の例です:

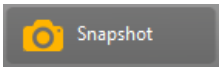


アドバンスドモード(オペレーションを**Shift+クリック**)では、さらなる情報が表示されます:



スナップショットボタン

スナップショットボタンを使用すると、RTMパネルに表示されている全てのオペレーションと、それに関連する詳細(プロセスとサブオペレーション情報)がクリップボードへとコピーされます:



オペレーションを最低5秒間表示する

オペレーションを最低5秒間表示するオプションをチェックすると、表示されたオペレーションはどれも(実行が終了した後も)最低5秒間は表示されたままになります:

このオプションが適用されたオペレーションは、オペレーションリストの中で灰色に表示されます。

この機能は、とても早く終わってしまうオペレーションの情報を取得したい場合に有効です。

4D Serverデータベースメソッド

On Server Startupデータベースメソッド
On Server Shutdownデータベースメソッド
On Server Open Connectionデータベースメソッド
On Server Close Connectionデータベースメソッド

On Server Startupデータベースメソッド

On Server Startupデータベースメソッド
このコマンドは引数を必要としません

On Server Startupデータベースメソッドは、4D Server でデータベースを開くと、サーバマシン上で一度呼び出されます。4D Server 以外の4D環境で**On Server Startupデータベースメソッド**が起動されることはありません。

On Server Startupデータベースメソッドは次の事柄を行うのに最適です:

- 4D Server セッション全体を通して使用するインタープロセス変数を初期化する
- データベースが開かれる時に自動で**ストアドプロシージャ**を開始する
- 前の4D Serverセッション中に保存された初期設定や各種設定をロードする
- 明示的に**QUIT 4D**を呼び出すことによって、(システムリソースが見つからない等) 条件が満たされていない場合にデータベースを開けないようにする
- データベースが開かれるたびに自動的に実行させたいその他の動作を実行する

リモート4Dがサーバに接続する時に、クライアントマシン上で自動的にコードを実行するには**On Startupデータベースメソッド**を使用します。

Note: On Server Startup データベースメソッドはアトミックに実行されます。つまりこのメソッドの実行が終了するまで、リモート4Dは接続を行うことができません。

On Server Shutdownデータベースメソッド

On Server Shutdownデータベースメソッド
このコマンドは引数を必要としません

On Server Shutdownデータベースメソッドは、カレントのデータベースが4D Server上で閉じられるときに、サーバマシン上で一度呼び出されます。4D Server以外の4D 環境では**On Server Shutdownデータベースメソッド**が起動されることはありません。

サーバ上のカレントデータベースを閉じるには、サーバ上で**データベースを閉じる...** メニューコマンドを使用します。また**4D Serverを終了** メニューコマンドを選択したり、サーバ上で実行されるストアドプロシージャ内で**QUIT 4D** コマンドを呼び出すこともできます。

データベースの終了が開始されると、4D は次の動作を実行します:

- **On Server Shutdownデータベースメソッド**がない場合、4D Server は実行中の各プロセスを区別なく1 つずつアボートします。
- **On Server Shutdownデータベースメソッド**がある場合、4D Server は新しく作成されたローカルプロセス内でこのメソッドの実行を開始します。したがって、このデータベースメソッドを使用し、プロセス間通信を介して、他のプロセスに対し、実行を停止するよう通知することができます。結局は、4D Server が終了するという点に注意してください**On Server Shutdownデータベースメソッド**では、片付けたり、クローズする操作をすべて実行することができますが、終了を拒否することはできないため、いずれかの時点で終了することになります。

On Server Shutdownデータベースメソッドは次の事柄を行うのに最適です:

- データベースが開かれた時に自動的に起動されたストアードプロシージャを停止する
- 次のセッションの始めに**On Server Shutdownデータベースメソッド**で再使用するために、初期設定や各種設定を（ディスク上にローカルに）保存する
- データベースが終了するたびに自動的に実行させたいその他の動作を実行する

警告: On Server Shutdownデータベースメソッドを使用してストアドプロシージャをクローズする場合、サーバは (ストアドプロシージャではなく)**On Server Shutdownデータベースメソッド**が実行されると終了することに留意してください。この時点でストアドプロシージャが起動されていると、それらはキルされます。
このため、サーバによりキルされる前に、ストアドプロシージャが完全に実行されたことを確認したい場合、**On Server Shutdownデータベースメソッド**はストアドプロシージャに対し実行を終了しなければならないことを通知して (例えばインタープロセス変数を使用)、そして終了を待つようにするべきです (x秒のループや他のインタープロセス変数を使用)。

リモートの4Dがサーバへの接続を停止する時に、クライアントマシン上で自動的にコードを実行させたい場合には、**On Exitデータベースメソッド**を使用してください。

On Server Open Connectionデータベースメソッド

$1, $2, $3 -> On Server Open Connectionデータベースメソッド -> $0

引数	型		説明
$1	倍長整数	⇦	ユーザーを識別するために4D Serverが内部的に使用するユーザーID
$2	倍長整数	⇦	接続を識別するために4D Serverが内部的に使用する接続ID
$3	倍長整数	⇦	廃止: 常に0が返されますが、宣言はしなくてはなりません。
$0	倍長整数	⇨	0または省略時 = 接続を受け入れる、0以外 = 接続を拒否する

On Server Open Connection データベースメソッドはいつ呼び出されるか

On Server Open Connectionデータベースメソッドは、4Dリモートワークステーションが接続プロセスを開始するたびに、サーバーマシン上で一度、呼び出されます。4D Server 以外の4D 環境では**On Server Open Connectionデータベースメソッド**が起動されることはありません。

On Server Open Connectionデータベースメソッドは以下のときに呼び出されます:

- リモート4Dが接続した (アプリケーションプロセスが開始するため)
- リモート4Dがデザインモードを開く (デザインプロセスが開始するため)
- リモート4Dが、サーバー上でコオペラティブプロセスの作成を必要とする* (プロセス名が"$"で始まらない) グローバルプロセスを開始した。このプロセスは**New process**コマンド、メニューコマンド、またはメソッド実行ダイアログボックスを使用して作成されます

リモート4Dでは、いずれの場合にも3つのプロセスが開始されます (クライアントマシン上に1つ、また必要に応じてサーバマシン上に2つ) 。クライアントマシンでは、プロセスでコードが実行され、4D Serverに要求が送られます。サーバマシンでは、**4Dクライアントプロセス**はクライアントプロセスのためのデータベース環境 (ユーザプロセスのためのカレントセレクションやレコードのロック等) を管理し、クライアントマシン上で実行中のプロセスから送られた要求に対して応答を返します。**4Dクライアントデータベースプロセス**は対応する4Dクライアントプロセスのモニタを担当します。

(*) 4D v13より最適化のため、サーバープロセス (データベースエンジンにアクセスするためのプリエンプティブプロセスとランゲージアクセスのためのコオペラティブプロセス) はクライアント側のコードを実行する際必要な時にだけ作成されるようになりました。以下は新規クライアントプロセスを実行する4Dコードの詳細です:

```
  // グローバルプロセスが開始されるがこの時点では
  //サーバー上にはプロセスは作成されない
CREATE RECORD([Table_1])
[Table_1])field1_1:="Hello world"
SAVE RECORD([Table_1]) // この時点でサーバーにプリエンプティブプロセスが作成される
$serverTime:=Current time(*) // ここでサーバー上にコオペラティブプロセスが作成され
  // On Server Open Connectionが呼び出される
```

重要: Web接続およびSQL接続は**On Server Open Connectionデータベースメソッド**を起動しません。Webブラウザーが4D Server に接続する場合は **On Web Authenticationデータベースメソッド** (あれば) と**On Web Connectionデータベースメソッド**が起動されます。4D ServerがSQLクエリを受信すると、**On SQL Authenticationデータベースメソッド**が (あれば) 呼び出されます。詳細については、4D Language Referenceマニュアルのデータベースメソッドに関する説明を参照してください。

重要: ストアドプロシージャの開始時には、**On Server Open Connectionデータベースメソッド**は起動されません。 **ストアドプロシージャ** はサーバプロセスであり、4Dクライアントプロセスではありません。ストアドプロシージャはサーバマシン上でコードを実行しますが、 4Dクライアント (または他のクライアント) と4D Server によってやり取りされる要求に対して応答を返すことはありません。

On Server Open Connection データベースメソッドはどのように呼び出されるか

On Server Open Connectionデータベースメソッド は4D Serverマシン上で、このメソッドを呼び出しを引き起こした4Dクライアントプロセス内で実行されます。

例えばリモート4Dが4D Server インタープリタデータベースに接続すると、そのクライアント用のユーザプロセスとデザインプロセス、クライアント登録プロセスが (デフォルトで) 開始されます。したがって **On Server Open Connectionデータベースメソッド** は3回実行されます。つまり1回目はアプリケーションプロセス内で、2回目はクライアント登録プロセス内で、3回目はデザインプロセス内で実行されます。3つのプロセスがそれぞれサーバマシン上で開始される6番目と7番目と8番目のプロセスである場合に、**On Server Open Connectionデータベースメソッド**内から **Current process** を呼び出すと、 **Current process** は1回目には6を、2回目には7を、3回目に8を返します。

On Server Open Connectionデータベースメソッド はサーバマシン上で実行されることに注意してください。このデータベースメソッドは、クライアント 側で実行中のプロセスとは無関係に、サーバマシン上で実行中の4Dクライアントプロセス内で実行されます。また、このメソッドが起動された時点では、4D クライアントプロセスにはまだ名前が付いていません (この時点では、**PROCESS PROPERTIES**は4Dクライアントプロセスの名前を返しません)。

On Server Open Connectionデータベースメソッド は、クライアント側で実行中のプロセスのプロセス変数テーブルにアクセスしません。このテーブルはサーバマシンではなく、クライアントマシンに存在します。

On Server Open Connectionデータベースメソッド がプロセス変数にアクセスすると、4Dクライアントプロセス用に動的にプロセス変数テーブルが作成され、プライベートに使用されます。

4D Serverは **On Server Open Connectionデータベースメソッド** に3つの倍長整数タイプの引数を渡し、倍長整数タイプの結果を求めます。したがってこのメソッドでは3つの引数と戻り値を倍長整数として明示的に宣言しなくてはなりません:

```
C_LONGINT($0;$1;$2;$3)
```

*$0*に値を返さず、その結果変数を未定義のままにするかまたはゼロに初期化した場合、4D Server はデータベースメソッドが接続を受け付けたものとみなします。接続を受け付けない場合、*$0*にヌルではない値を返します。

次の表はこのデータベースメソッドに渡される3つの引数が示す情報を表わしています:

引数	説明
$1	4D Serverがユーザを識別するために内部的に使用するユーザID番号
$2	4D Serverが接続を識別するために内部的に使用する接続ID番号
$3	廃止: 常に0が渡されますが、宣言は必要

これらのID番号は、例えば4Dコマンドに渡す引数のように、情報ソースとして直接使用することはできません。しかしこれらのID番号は**On Server Open Connectionデータベースメソッド**と**On Server Close Connectionデータベースメソッド**との間で、4Dクライアントプロセスを一意に識別するために利用できます。4D Server セッションのどの時点でも、これらの値の組み合わせはユニークです。インタープロセス配列やテーブルにこの情報を格納することによって、2つのデータベー スメソッド間で情報をやり取りできます。この節の最後に示された例では、2つのデータベースメソッドがこの情報を使用して、テーブルの同一レコードに接続 の開始と終了の日付と時間を格納しています。

例題 1

次の例は**On Server Open Connectionデータベースメソッド**と**On Server Close Connection データベースメソッド**を使用して、データベースへの接続ログを管理する方法を示しています。**[Server Log]**テーブル (下図) は接続処理の記録を取るために使用されています:

このテーブルに格納される情報は、次の**On Server Open Connectionデータベースメソッド**と**On Server Close Connection データベースメソッド**によって管理されます:

```
` On Server Open Connection データベースメソッド
```

```
C_LONGINT($0;$1;$2;$3)
  ` [Server Log] レコード作成
CREATE RECORD([Server Log])
[Server Log]Log ID:=Sequence number([Server Log])
  ` 接続日付と時間を保存
[Server Log]Log Date:=Current date
[Server Log]Log Time:=Current time
  ` 接続情報を保存
[Server Log]User ID:=$1
[Server Log]Connection ID:=$2
SAVE RECORD([Server Log])
  ` エラーなしを返すと接続が続行される
```

```
$0:=0  ` On Server Close Connection データベースメソッド
C_LONGINT($1;$2;$3)
  ` [Server Log] レコードを取得
QUERY([Server Log];[Server Log]User ID=$1;*)
QUERY([Server Log];&;[Server Log]Connection ID=$2)
  ` 終了日付と時間を保存
[Server Log]Exit Date:=Current date
[Server Log]Exit Time:=Current time
  ` プロセス情報を保存
[Server Log]Process ID:=Current process
PROCESS PROPERTIES([Server Log]Process ID;$vsProcName;$vlProcState;$vlProcTime)
[Server Log]Process Name:=$vsProcName
SAVE RECORD([Server Log])
```

下図は[Server Log]に登録されたレコードで、いくつかのリモート接続を示しています:

MyMusic.4DB - Server Log : 24 enregistrement(s) sur 24

Log ID :	Log Date :	Log Time	Exit Date :	Exit Time	User ID :	Connection ID	Process ID	Process Name :
13	16/06/2008	17:46:20	16/06/2008	17:50:10	12274272	122978312	6	Process principal
14	16/06/2008	17:46:23	16/06/2008	17:50:09	12274272	122444176	7	Process développement
15	16/06/2008	17:46:41	16/06/2008	17:46:49	12274272	124620824	8	P_1
16	16/06/2008	17:47:21	16/06/2008	17:50:03	12274272	122683400	8	P_2
17	16/06/2008	17:49:53	16/06/2008	17:50:05	12274272	122797960	9	P_3
18	16/06/2008	17:50:17	16/06/2008	18:25:22	16112358	122978312	6	Process principal
19	16/06/2008	17:50:20	16/06/2008	18:25:11	16112358	252709968	7	Process développement
20	16/06/2008	17:51:08	16/06/2008	17:51:08	16112358	122826440	8	P_1
21	16/06/2008	17:51:13	16/06/2008	18:25:21	16112358	122939152	8	P_2
22	16/06/2008	17:51:16	16/06/2008	18:24:43	16112358	122960760	9	P_3
23	16/06/2008	17:51:19	16/06/2008	18:24:45	16112358	123112040	10	P_4
24	16/06/2008	17:51:36	16/06/2008	18:25:21	12274272	123346952	11	P_5
25	16/06/2008	17:51:39	16/06/2008	17:51:39	12274272	123575008	12	P_6
26	16/06/2008	17:51:41	16/06/2008	17:51:41	12274272	123575968	12	P_7
27	16/06/2008	17:51:53	16/06/2008	18:07:56	12274272	123621968	12	P_8
28	16/06/2008	18:25:25	16/06/2008	18:30:22	12274272	122978312	6	Process principal
29	16/06/2008	18:25:34	16/06/2008	18:30:21	12274272	122879504	7	Process développement
30	16/06/2008	18:26:58	16/06/2008	18:26:58	12274272	124727792	8	P_1
31	16/06/2008	18:26:58	16/06/2008	18:27:46	16112358	124772984	9	Client en attente
32	16/06/2008	18:27:16	16/06/2008	18:28:06	12274272	124828872	8	P_2

例題 2

以下の例題は午前2時から4時の間の接続を拒否します。

```
  ` On Server Open Connection データベースメソッド
C_LONGINT($0;$1;$2;$3)
```

```
If((?02:00:00?<=Current time)&(Current time<?04:00:00?))
   $0:=22000
Else
   $0:=0
End if
```

On Server Close Connectionデータベースメソッド

$1, $2, $3 -> On Server Close Connectionデータベースメソッド

引数	型		説明
$1	倍長整数	⇦	ユーザーを識別するために4D Serverが内部的に使用するユーザーID
$2	倍長整数	⇦	接続を識別するために4D Serverが内部的に使用する接続ID
$3	倍長整数	⇦	廃止: 常に0が返されますが、宣言はしなくてはなりません。

説明

On Server Close Connectionデータベースメソッドは、4Dクライアントプロセスが終了するたびに、サーバマシン上で一度呼び出されます。

On Server Open Connectionデータベースメソッドの場合と同様に、4D Server は**On Server Close Connectionデータベースメソッド**に3つの倍長整数タイプの引数を渡しますが、結果は求めません。

したがって、このメソッドでは3つの引数を倍長整数として明示的に宣言しなくてはなりません:

```
C_LONGINT($1;$2;$3)
```

次の表は、このデータベースメソッドに渡される3つの引数が示す情報を表わしています:

引数	説明
$1	4D Serverがユーザを識別するために内部的に使用するユーザID番号
$2	4D Serverが接続を識別するために内部的に使用する接続ID番号
$3	廃止: 常に0が渡されますが、宣言は必要

On Server Close Connectionデータベースメソッドは、**On Server Open Connectionデータベースメソッド**と対をなすメソッドです。4Dクライアントプロセスについての詳細は、このデータベースメソッドの説明を参照してください。

例題

On Server Open Connectionデータベースメソッドの例題参照

リモートの4Dを使用する

4D Serverデータベースへの接続

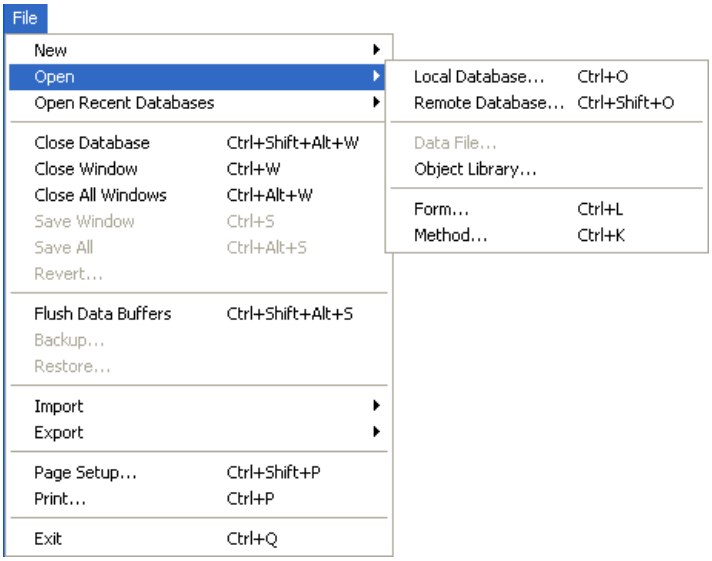
リモート4Dから4D Serverデータベースに接続する方法は3つあります:

- 接続ダイアログボックスを使用する
- **最近使用したデータベースを開く**メニューを使用する
- データベースへのアクセスパラメタを含む4DLinkショートカットファイルを使用する

接続ダイアログボックスを使用する

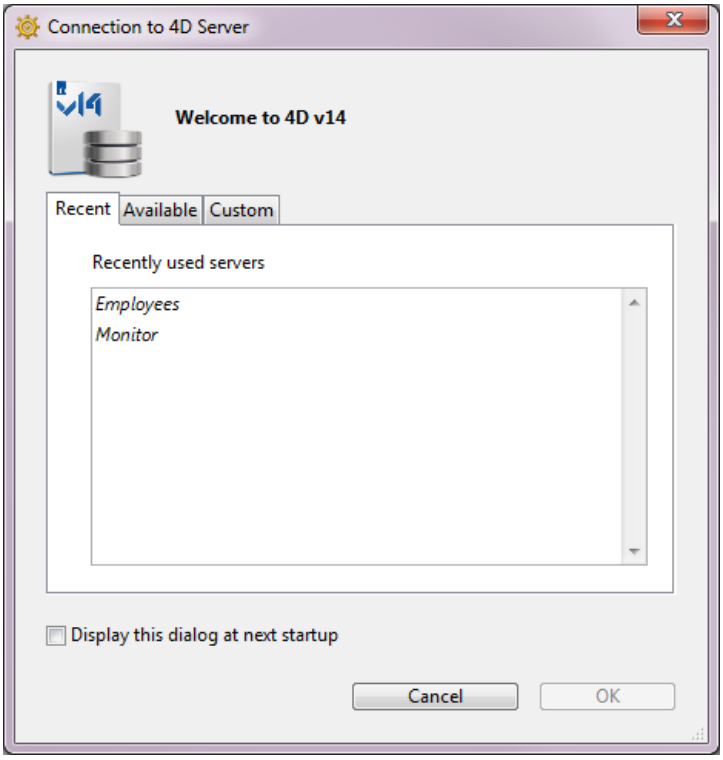
4D Server接続ダイアログボックスを表示するには、まず4Dアプリケーションを起動します。

ファイルメニューの**開く**コマンド (または4Dツールバーの対応するボタン) を使用して、4Dデータベースを開くモードを選択できます:



開く>リモートデータベース... コマンドを選択

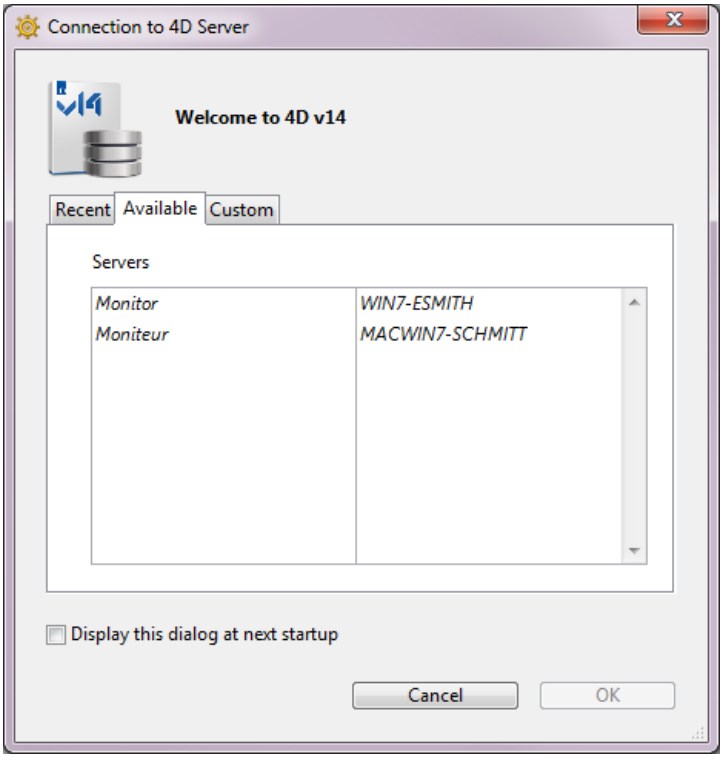
4D Server接続ダイアログボックスが表示されます。このダイアログボックスには最近使用したサーバ、利用可、そしてカスタムの3つのページがあります:



このダイアログを次回起動時に表示オプションが選択されていると、4Dアプリケーション起動時に自動でこのダイアログボックスが表示されます。

Note: 4Dのウェルカムダイアログボックスで**4D Serverに接続**をクリックしてもこのダイアログを表示できます。

“利用可” ページ



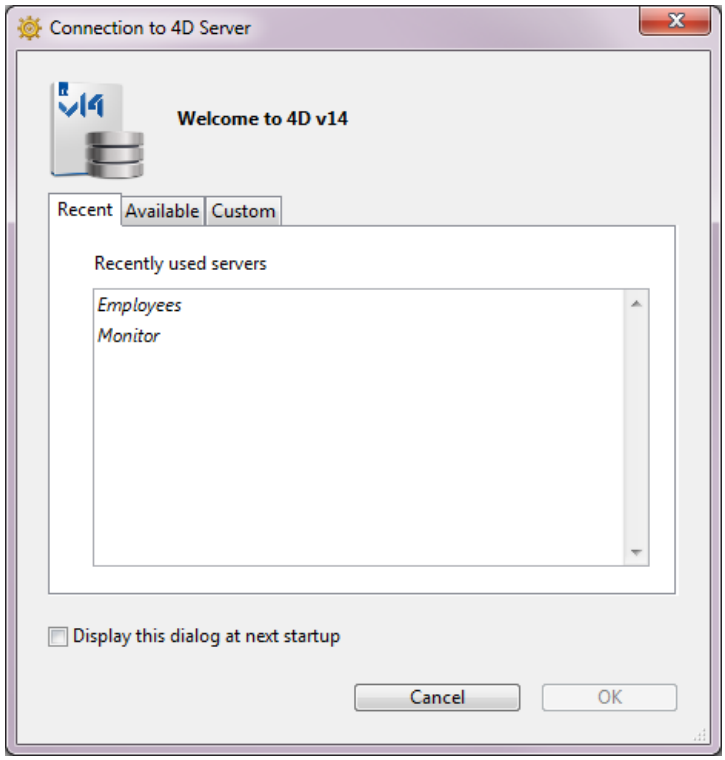
4D Serverには組み込みのTCP/IPブロードキャストシステムがあり、デフォルトで、ネットワーク上に利用可能な4D Serverデータベースの名前を公開します。この名前は接続ダイアログボックスの**利用可**ページに表示されます。

このリストは、名前が見つかった順に表示され、動的に更新されます。このリストからサーバに接続するには、名前上でダブルクリックするか、名前を選択して**OK** ボタンをクリックします。

Notes:

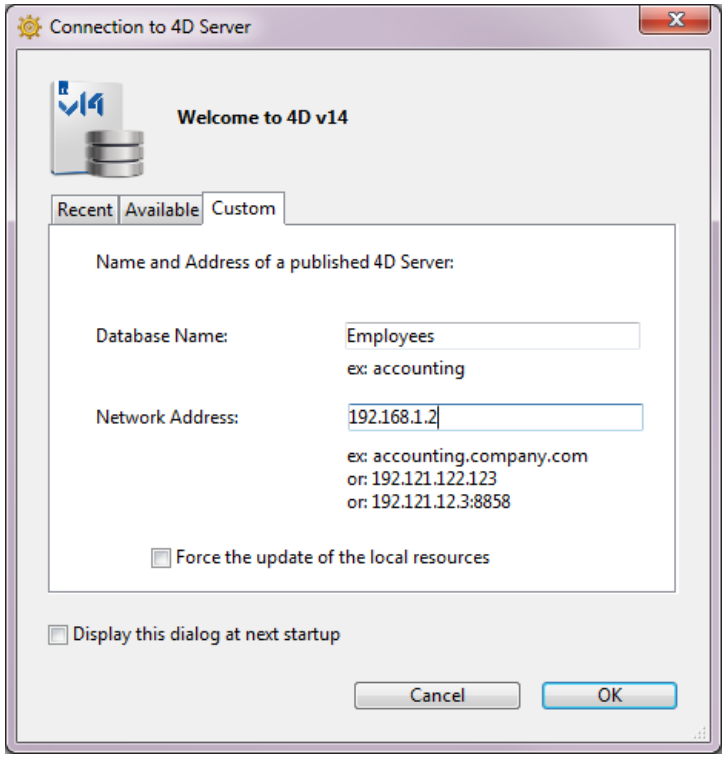
- 暗号化オプションが有効で公開されているデータベース名の前にはキャレット (^) が置かれます。詳細は**クライアント/サーバ接続の暗号化**を参照してください。
- ネットワーク上へのデータベース名の公開を禁止できます (**公開 (環境設定)**参照)。この場合、このページからの接続はできません。カスタムページを使用します。

“最近使用したサーバ” ページ



最近使用したサーバページは、最近使用した4D Serverデータベースを記憶しています。リストは名前順でソートされます。このリストからサーバに接続するには、名前上でダブルクリックするか、名前を選択して**OK** ボタンをクリックします。

“カスタム” ページ



カスタムページではIPアドレスでネットワーク上のサーバを指定し、それに任意の名前をつけられます。

4D ServerのTCP/IPブロードキャストシステムをカスタマイズして、ネットワーク上に自動ではサーバデータベースの名前を公開しないようにできます (**公開 (環境設定)**参照)。この場合、サーバデータベース名は"利用可"ページに表示されません。しかし、ブロードキャストされていないサーバデータベースのIPアドレスを知っていれば、そのIPアドレスを入力できます。

- **データベース名**: 4D Serverデータベース名を指定できます。この名前は**最近使用したサーバ**ページでデータベースを参照する際に使用されます。
- **ネットワークアドレス**: 4D Serverが起動されたマシンのIPアドレスを指定します。2つのサーバが同時に同じマシン上で起動されているときは、IPアドレスの後にコロンとポート番号を続けます。例えば: 192.168.92.104:19820
  デフォルトで、4D Serverの公開ポートは19813です。この番号はアプリケーションの環境設定で変更できます (**設定**

(環境設定))。

Note: カスタムタブをクリックするときに**最近使用したサーバ**や**利用可**ページで項目が選択されていた場合、これら2つのフィールドには対応する情報が表示されます。

このページでサーバを指定したら、**OK**ボタンをクリックしてサーバに接続できます。接続したサーバは**最近使用したサーバ**のリストに加えられます。

Note: データベースが暗号化されて公開されている場合、名前の前にキャレット (^) を置かなければなりません。そうでなければ接続は拒否されます。詳細は**クライアント/サーバ接続の暗号化クライアント/サーバ接続の暗号化**を参照してください。

ローカルリソースを強制的に更新

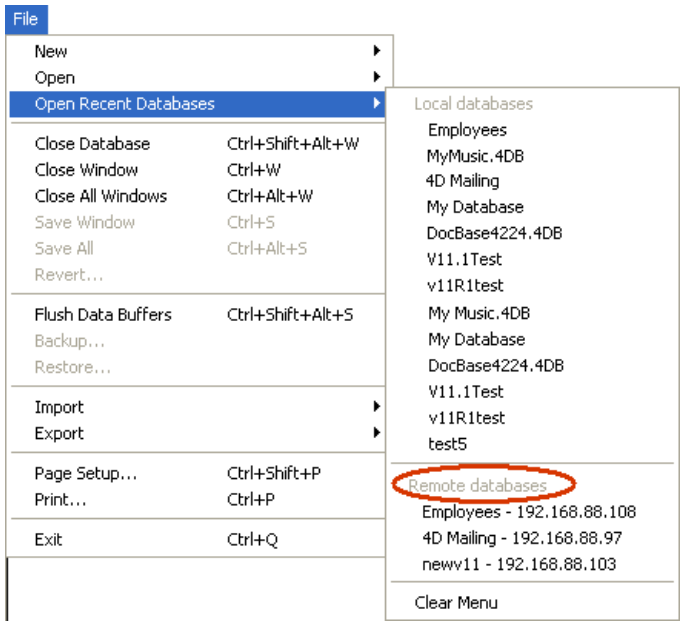
このオプションを選択すると、接続時にクライアントマシン上のローカルリソースを機械的に更新します。ローカルリソースはそれぞれのクライアントマシンに格納される、データベースに関連したストラクチャ情報です。

ルール上ローカルリソースの更新は、リモートマシンが接続したときに、データベースのストラクチャが前回から更新されていれば、自動で行われます。ほとんどの場合、このオプションは必要ありません。特別なケースで更新を強制したい時にこのオプションを使用します。

最近使用したデータベースを開くメニューを使用する

最近使用したデータベースを開くメニューコマンドは、以前に接続したことのある4D Serverデータベースに直接接続するために使用します。

このコマンドは4Dの**ファイル**メニューにあります。4Dを使用してローカルデータベースおよびリモートデータベースを開いている場合、このメニューには両タイプのデータベースがリストされます。リモートデータベースはメニューの下部にリストされます:



データベース名の隣にはIPアドレスが表示されます。

メニュークリア コマンドでメニューをリセットできます。

4DLinkファイルを使用する

4Dデータベースを自動で簡単に開くために使用するアクセスファイルを生成できます。通常アクセスファイルは4D Serverリモートデータベースの IPアドレスと接続識別子を保存し、ユーザが行うべき作業を取り除きます。

アクセスファイルを使用してローカルデータベースを開くこともできます。

ファイルの作成

4Dデータベースの接続ファイルはXML ファイルで、".4DLink" 拡張子が付けられます。4Dは"最近使用したデータベース" のサブメニューを構築するために、このタイプのファイルを生成して利用します。ローカルデータベースを初めて開くとき、またはサーバに接続するとき、4D は.4DLink ファイルを自動で生成します。

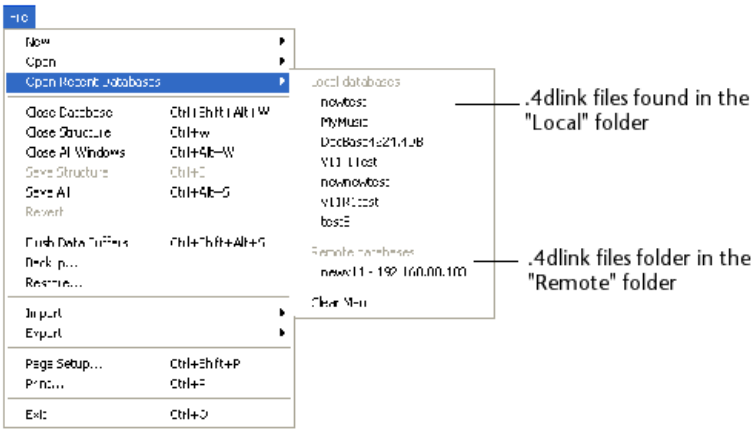
4Dが自動で作成した.4DLink ファイルは、ユーザのローカルな環境設定フォルダに置かれます。このフォルダには、2 つのディレクトリLocal とRemote が作成されます。Local フォルダにはローカルデータベースへの接続に使用できる".4DLink" ファイルが、Remote フォルダにはリモートデータベースへの接続に使用できる"4DLink" ファイルが置かれます。

ローカル環境設定フォルダは以下の場所にあります:

- Windows 7 以降: C:¥Users¥*UserName*¥AppData¥Roaming¥4D¥Favorites v*XX*¥
- OS X: Users/*UserName*/Library/Application Support/4D/Favorites v*XX*/

XX はアプリケーションのバージョン番号を意味します(例えば、4D v14なら"Favorites v14"となります)。

このディレクトリに置かれたファイルは、4D の**ファイル**メニューの**最近使用したデータベースを開く**> サブメニューに表示されます:



".4DLink" ファイルはXML エディタを使用して作成し、接続ID (ユーザ名やパスワード) やデータベースを開くモードなどをカスタマイズした情報を含めることもできます。

".4DLink" ファイルを構築するために使用できるXML キーを定義したDTD が4D より提供されます。このDTD は**database_link.dtd**という名前で、4D の**/Resources/DTD**サブフォルダに見つけることができます。

ファイルを使用する

.4DLinkアクセスファイルを使用して4Dアプリケーションを起動し、目的の4Dデータベースを開くことができます。ファイルを使用する方法は2つあります:

- ダブルクリック、または4Dアプリケーションにドラッグ&ドロップ
- **最近使用したデータベースを開く**サブメニュー (ローカル環境設定フォルダ内のファイル)
  "リモート"データベースタイプの.4DLinkファイルを他のマシンにコピーして使用できます。

Note: 4Dと4D Serverの接続ダイアログで4DLinkファイルを選択することも可能です (ローカルデータベースの開始のみ)。

リモートマシンからの管理

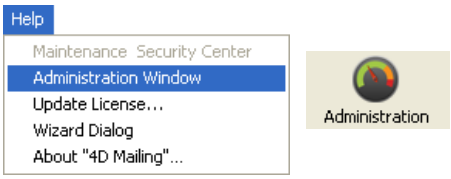
4D Serverマシンを4D (クライアントマシン) から管理できます。これを行うには4D Server管理ウィンドウ (**C_POINTERモニタページ**参照) をクライアントマシンで開きます。

リモート4Dマシンで管理ウィンドウを開く

クライアントマシンでサーバ管理ウィンドウを開くには、Designer またはAdministrator としてリモートデータベースに接続しなければなりません。そうでないばあい、管理ウィンドウを開こうとするとアクセス権エラー (-9991) が生成されます。

以下の方法を使用できます:

- **ヘルプ**メニューから**管理ウィンドウ**コマンドを選択するか、4D ツールバーの対応するボタンをクリックする:



- **OPEN ADMINISTRATION WINDOW** コマンドを実行する

サーバ管理ウィンドウがクライアントマシン上で表示されます:

クライアントマシンでの管理における注意点

サーバ管理ウィンドウを表示しているクライアントマシンは、利用可能なすべての情報にアクセスでき、プロセスを操作したり、サーバを開始/停止したりできます。
しかしながら、クライアントマシン上でサーバ管理ウィンドウが表示されているとき、特定の制限や操作に関する機能があります:

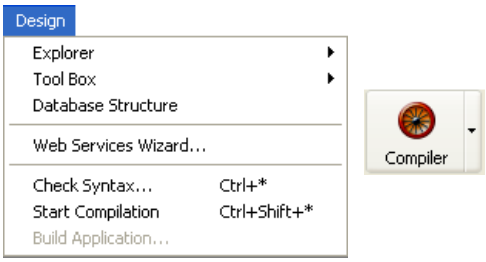
- **プロセスページ**,で、ユーザプロセスをデバッグすることはできません (デバッグウィンドウはサーバ上に表示されます)。
- **メンテナンスページ**,ですべてのユーザの接続を解除し、サーバの再起動を引き起こす作業を実行できます (圧縮と再起動動作)。この場合、その操作をリクエストしたクライアントマシンは、再起動時に自動で再接続されます。
- **メンテナンスページ**,で、メンテナンス操作後、**レポート表示**ボタンは**レポートダウンロード**ボタンとして表示されます。これらのファイルは表示される前に、クライアントマシンのデータベースフォルダにダウンロードされます。

リモートマシンからのコンパイル

リモート接続を使用して、4Dアプリケーションをコンパイルできるようになりました。言い換えれば、4Dクライアントマシンからコンパイルを行えます。以前のバージョンでは、コンパイルはシングルユーザアプリケーションでのみ可能でした。

Note: 他方、カスタム4Dアプリケーション (シングルユーザあるいはクライアント/サーバ) をリモート接続からビルドすることはできません。アプリケーションビルダにこの環境からアクセスすることはできません (メニューコマンドが選択不可になります)。

クライアント側で、クライアントインターフェースと操作方法は、シングルユーザバージョンのそれと同じです。コンパイルは**デザイン**メニュー、ツールバー、あるいはコンパイルウィンドウから実行できます:



Note: クライアントマシンがコンパイル機能にアクセスするためには4D Server側に、"4D Team Server" ライセンスが必要です。

一度に1 つのクライアントマシンだけがコンパイルを行えます。クライアントがコンパイルを行っている間、この機能は他のマシンに対してロックされます。同時に他のマシンがデータベースのコンパイルを行おうとすると、警告ダイアログが表示されます。

1 つのクライアントマシンがコンパイルを行っている間、他のクライアントマシンはメソッドやその他のストラクチャ要素を更新できます。このような場合、コンパイルされたコードとインタプリタコードは異なることになり、再度コンパイルが必要になります。

コンパイルされたコードは、コンパイルが実行されるごとにサーバ上の.4DB ファイルに送信されます。
クライアント側で、コンパイルが終了したあと、**実行**メニューの対応するコマンドを使用して、インタープリタモードまたはコンパイル済みモードでデータベースを再起動できます。コンパイル済み/インタープリタでサーバの再起動がクライアントからリクエストされると、標準のサーバ終了ダイアログが表示され、待ち時間や他のクライアントに送信するメッセージを設定できます (**4D Serverを終了する**参照)。サーバが再起動されると、再起動のリクエスト元のクライアントは自動で再接続されます。
サーバ側では、コンパイル済み/インタープリタでの再起動は標準の開くダイアログボックスを使用する必要があります (**開く**ボタンに関連付けられたポップアップメニュー)。

Note: パフォーマンスの理由から、WAN ネットワーク上でのコンパイルは推奨されません (特にデータベースに数多くのメソッドが含まれている場合)。これを行うと、ネットワーク上での数多くのデータ交換が発生します。

4D Serverと4Dランゲージ

4D Serverと4Dランゲージ

4D Serverで、サーバマシン上で4Dコードを実行する状況が4つあります:

- トリガ
- ストアドプロシージャ
- "サーバ上で実行"属性が設定されたプロジェクトメソッド
- データベースメソッド

トリガ

トリガはテーブルに付属するメソッドです。トリガを使用すれば、データベースのレコードに対して"不正な"操作が行われるのを防ぐことができます。 トリガは、偶発的にデータが失われたり、変更されたりするのを防ぐだけでなく、テーブルに対する操作を制限するための非常に強力なツールです。例えば、請 求システムにおいて、誰かが請求先である顧客を指定せずに請求書を追加するのを防止することができます。

トリガは、データベースエンジンが実際に存在するマシン上で実行されます。
4D Serverでは、トリガはクライアントマシンではなく、サーバマシン上で実行されているプロセスのコンテキストで実行されます。すなわちトリガは、デー タベース処理を呼び出したユーザプロセスに対応するサーバプロセスのコンテキストで実行されます (特にトランザクションの状態とレコードロック) が、ランゲージコンテキスト (変数、プロセス、セット、カレントセレクション) は共有しません。ただしトリガテーブルのカレントレコードはすべてのコンテキストで同じです。

トリガについては4D Language Reference の**ARRAY REAL**を参照してください。

ストアドプロシージャ

4Dのストアードプロシージャは、それを起動したクライアントマシンの代わりに、サーバマシン上で実行されるプロセス内のプロセスメソッドを実行するプロジェクトメソッドです。**ストアドプロシージャ**を参照してください。

"サーバ上で実行" 属性付きのメソッド

"サーバ上で実行"の属性が指定されたプロジェクトメソッドもサーバ上で実行されます。しかしストアドプロシージャと異なり、このメソッドはクライアントプロセスに対応するサーバ上のプロセスで実行され、トリガのように、データベースコンテキストを利用できます。詳細は**サーバー上で実行属性**を参照してください。

データベースメソッド

4つのデータベースメソッドがサーバマシン上でのみ実行されます:

- **On Server Startupデータベースメソッド**
- **4D Server管理ウィンドウOn Server Shutdownデータベースメソッド**
- **On Server Open Connectionデータベースメソッド**
- **On Server Close Connectionデータベースメソッド**

他の5つのデータベースメソッドはコンテキストに応じてサーバマシンおよびクライアントマシン両方で実行されます:

- **On Web Authenticationデータベースメソッド**
- **On Web Connectionデータベースメソッド**
- **On SQL Authenticationデータベースメソッド**
- **On Backup Startupデータベースメソッド**
- **On Backup Shutdownデータベースメソッド**

3つのデータベースメソッドはクライアントマシン上でのみ実行されます:

- **On Startupデータベースメソッド**
- **On Exitデータベースメソッド**
- **On Dropデータベースメソッド**

詳細はこのマニュアル内および4D Language Reference マニュアルのそれぞれ対応する節を参照してください。

4D Server と変数

- 4D Server はインタープロセス変数テーブルを1つ維持します。これらの変数の有効範囲はサーバマシンです。コンパイルしたデータベースを実行している場合、インター プロセス変数テーブルの定義は、サーバマシンとすべてのクライアントマシンとで共通です。各マシンはそれぞれ独自のインスタンスを保持します。
- すべてのプロセスと同様、それぞれのストアドプロシージャ、データベースメソッド、トリガは独自のプロセス変数テーブルを持ちます。これらのプロセス変数は実行の各フェーズの間、動的に作成、使用されます。

4D Serverのセットと命名セレクション

4D Serverにおいて、セットと命名セレクションの可視範囲はそれが作成された場所 (サーバプロセスまたはクライアントプロセス) およびそのオブジェクトタイプ (ローカル、プロセス、またはインタープロセスオブジェクト) に基づきます。詳細は**4D Server: セットと命名セレクション**を参照してください。

4D Server: セットと命名セレクション

4D Language Referenceの**セット**および**命名セレクション**で説明しているように、ユーザはインタープロセス、プロセス、およびローカルのセットと命名セレクションを使用できます:

- **プロセスセット/命名セレクション**: プロセスオブジェクトには、それを作成したプロセス内、およびクライアントプロセスで作成された場合には対応するサーバ上のクライアントプロセス上でのみアクセスできます。プロセスオブジェクトはプロセスメソッドが終了すると消去されます。プロセスオブジェクトはその名前に特別な接頭辞を必要としません。
- **インタープロセスセット/命名セレクション**: インタープロセスオブジェクトはそれが作成されたマシン上 (クライアントまたはサーバ) のすべてのプロセスで可視です。名前の前に小なり記号と大なり記号の組合せ (**<>**) が付いていれば、そのセットや命名セレクションはインタープロセスセットです。このシンタックスはWindowsとMac OS両方で使用できます。
- **ローカル/クライアントセット/命名セレクション**: ローカル/クライアントオブジェクトはそれが作成されたプロセス内でのみ可視です。ローカル/クライアントオブジェクトの名前の前には、ドル記号 (**$**) を付けます。
  注: *UserSet*システムセットの名前は**$**で始まっていませんが、ローカル/クライアントセットです。

以下の表は命名セレクションとセットがどこで作成されたかに基づき、これらの可視性の原則を示しています (表は両タイプのオブジェクトで共通です):

	Client Process	Other client processes	Other clients	Server process	Other server processes
Creation in a client process					
$test	x				
test	x			x (Trigger)	
<>test	x	x			
Creation in a server process					
$test				x	
test				x	
<>test				x	x

x = 可視

行いたい処理に基づき、この可視表を常に心にとめておいてください。例えば**DIFFERENCE**, **INTERSECTION** あるいは **UNION**タイプの処理を行うとき、すべてのセットや命名セレクションが、処理を行うマシン上で可視であることを確認してください。

最適化のため、可視性の必要性に基づきオブジェクトを作成する場所やスコープを決定することをお勧めします。

ストアドプロシージャ

SQLベースのストアドプロシージャとは?

ストアドプロシージャーという表現は、SQLベースのサーバーの世界に由来しています。クライアントワークステーションがSQLベースのサーバーに要求を送信する時、実際にはSQLサーバーに対してSQL言語で記述されたテキストを送信します。この要求は、実行される前にSQLサーバー上で解釈されます。要求のソースコードのサイズが大きく、1回のセッション中に要求が何度も送信される場合には、送られる要求の回数が多いほど、ネットワーク経由でソースコードを送信、解析し、解釈する時間が長くなることは明らかです。

そこでネットワーク経由で要求を送信し、解析および解釈を一度だけ行い、クライアントワークステーションから受信するたびにこれを実行する方法を探しました。この解決方法は、要求のソースコード (つまりプロシージャー) をサーバー側に保存し、クライアントワークステーションには実行するプロシージャーの名前だけで構成される要求を送らせることでした。結果的に、このプロシージャーはサーバー上にストアされるため、"ストアドプロシージャー"という用語になっています。

SQLベースのストアドプロシージャーは、クライアントワークステーションから引数を受信し、実現するタスクを (同期的または非同期的に) 実行し、最終的に結果をクライアントワークステーションに戻すことができるプロシージャーであるということに注意してください。クライアントワークステーションがストアドプロシージャーの実行を開始すると、ある程度サーバーマシンにコードの実行を任せます。

4D Server のストアドプロシージャとは?

4D Server では、業界で通用しているストアドプロシージャーという名称を使用していますが、4D Server のストアドプロシージャーの機能は、通常のストアドプロシージャーの概念をはるかに超えています。

ローカルモードの4D で**New process**のようなコマンドを使用すると、メソッドを実行できるユーザープロセスを開始することができます。このメソッドは**プロセスメソッド**と呼ばれています (4D Language Referenceマニュアルの**プロジェクトメソッド**参照)。

4D Server上でもクライアントマシンと同様の操作が可能です。さらに**Execute on server**コマンドを使用すると、4D Serverマシン上でメソッドを実行できるユーザープロセスを開始できます。**EXECUTE ON CLIENT**を使用すれば異なるクライアント上の他のプロセスでメソッドを実行できます。
両ケースでこのメソッドは**ストアドプロシージャー**と呼ばれ、(用語の濫用になりますが) サーバーマシンやクライアント上で開始されたプロセスもストアドプロシージャーと呼ばれます。

重要: SQL ベースのストアドプロシージャーと4D Server のストアドプロシージャーの本質的な違いは、SQL ベースのストアドプロシージャーではSQL プロシージャーを実行し、4D Server のストアドプロシージャーではスタンドアロン4Dプロセスを実行するという点にあります。

4Dストアドプロシージャのアーキテクチャ

通常のプロセスと同様に、ストアドプロシージャーには次のような独自の環境があります

- **テーブルごとのカレントセレクション**: 各ストアドプロシージャーには、個別のカレントセレクションがあります。1 つのテーブルは、別々のストアドプロシージャーで異なるカレントセレクションを持つことができます。
- **テーブルごとのカレントレコード**: 各テーブルは、ストアドプロシージャーごとに異なるカレントレコードを持つことができます。
- **変数**: 各ストアドプロシージャには独自のプロセス変数があります。プロセス変数は、その変数が設定されたストアドプロシージャの範囲内でのみ認識されます。
- **デフォルトテーブル**: 各ストアドプロシージャーには、独自のデフォルトテーブルがあります。
- **プロセスセット**: 各ストアドプロシージャーには、独自のプロセスセットがあります。
- **エラー処理**: 各ストアドプロシージャーには、独自のエラー処理メソッドを持てます。
- **デバッガウィンドウ**: 各ストアドプロシージャーは、独自のデバッガーウィンドウを持てます。

ユーザーインターフェースの点では、ストアドプロシージャーは、ウィンドウを開き、データを表示する (例えば**DISPLAY RECORD**を使用) ことができます。
4Dクライアントマシン上で実行されるストアドプロシージャーではデータ入力ができます。

一方、サーバー上で実行されるストアドプロシージャーではデータ入力を開始する (例えば**ADD RECORD**を使用) ことはできません。これは、サーバーマシン上にデータ入力カーネルがないためです。

ストアドプロシージャーは、システム (ハードウェアおよびメモリ) が許す限りいくつでも開始することができます。事実、4D Server マシンは、4Dクライアントおよび Web ブラウザーに応答するマシンであるだけではなく、サーバーマシンおよびリモート4Dマシン上で実行中の他のプロセスと対話するプロセスを実行するマシンである、という見方をする必要があります。

4Dがマシン上で実行されるユーザープロセスのマルチタスク環境を提供するのと同じ方法で、4D Serverはストアドプロシージャーに対してマルチタスク環境を提供します。たとえば、4D Server はプロセス間通信用にストアドプロシージャーで使用できるインタープロセス変数テーブルを管理しています。

注: "サーバー上で実行"メソッド属性を使用して、サーバー上のプロセスでメソッドを実行することもできます。ただしこの場合メソッドは、クライアントプロセスに対応するサーバー上のクライアントプロセスで実行されます。つまりクライアントプロセスの環境を使用できます。この場合、これは4Dのストアドプロシージャーではありません。詳細は**サーバー上で実行属性**を参照してください。

ストアドプロシージャの機能

データ入力を除き、4D Language Referenceマニュアルで説明されている、ほとんどすべてのプロセスおよびコマンドの機能は、ストアドプロシージャーにも適用されます。

ストアドプロシージャーではレコードの追加、検索、並べ替え、更新、削除が可能です。ストアドプロシージャーではセットや命名セレクションの使用、ディスク上のドキュメントファイルへのアクセス、BLOB を使用した作業、レコードの印刷等が行えます。ローカルの4Dマシン上で作業を行う代わりに、サーバーマシン上や他の4Dクライアントマシン上で実行していると考えてください。

ストアドプロシージャーの明らかな利点は、データベースエンジンがあるサーバーマシン上でローカルに実行されるということです。例えば、ネットワーク経由で**APPLY TO SELECTION**を行うと効率的ではありませんが、ストアドプロシージャー内では効率良く実行されます。**SPベースの読み込み (例題)**に示された例では、“スマート”なストアドプロシージャーを使用して、大幅なパフォーマンスの最適化を実現しています。

クライアントマシン上で実行されるストアドプロシージャーを使用すれば、タスクの分割やクライアントマシン間の通信を最適化できます。複数のマシンでストアドプロシージャーを実行する例題は、4D Language Reference内の**REGISTER CLIENT**を参照してください。

しかし、ストアドプロシージャアーキテクチャーの最も重要な利点は、4D Server に追加の世界をもたらすところです。ストアドプロシージャーを利用すると、独自の4D Serverサービスを実現することができます。これを制限するのは想像力だけです。**SPベースのサービス (例題)**の例では、4D Server またはサーバーマシンについての情報をクライアントに提供するストアドプロシージャーを示しています。例えば、サーバーマシンのボリュームを一覧表示することが可能です。この例は、ディレクトリ情報やドキュメント情報をクライアントに返すように簡単に拡張することができます。

(サーバー上で実行された)ストアドプロシージャーで行わないこと

一般に言って、サーバー上で実行されるストアドプロシージャーはインターフェース (メニューやウィンドウ、フォームなど) を扱うべきではありません。実際インターフェースはサーバー上では管理されません。
サーバーマシン上でダイアログボックスを表示するコマンドやデータ入力のためにダイアログボックスを表示するようなコマンドは避けるべきです。

以下はサーバー上で実行されるストアドプロシージャー内で使用すべきでないコマンドのリストです。コマンドは3つにグループ化されます:

- **サーバー上で禁止されるコマンド**

以下のコマンドの１つををストアドプロシージャー内で使用したら、4D Server 上で使ってはいけないというアラートが表示されます。エラーは#67 が返ります。**ON ERR CALL**コマンドでインストールされたメソッドを通し、受け取ることができます。

ACCUMULATE
ADD RECORD
_o_ADD SUBRECORD
APPEND MENU ITEM
BREAK LEVEL
CALL PROCESS

CHANGE LICENSES
Count menu items
Count menus
CREATE USER FORM
DELETE MENU ITEM
DELETE USER FORM
DISABLE MENU ITEM
DISPLAY SELECTION
EDIT ACCESS
EDIT FORM
ENABLE MENU ITEM
FILTER EVENT
Get menu item
Get menu item key
Get menu item mark
Get menu item style
Get menu title
SET PICTURE TO LIBRARY
_o_GRAPH TABLE
INSERT MENU ITEM
Level
LIST USER FORMS
Menu selected
MODIFY RECORD
MODIFY SELECTION
_o_MODIFY SUBRECORD
ON EVENT CALL
Open external window
PAGE BREAK
PAGE SETUP
PRINT SETTINGS
QUERY BY EXAMPLE
QR REPORT
Printing page
REMOVE PICTURE FROM LIBRARY
SET MENU ITEM
SET MENU ITEM SHORTCUT
SET MENU ITEM MARK
SET MENU ITEM STYLE
SET PICTURE TO LIBRARY
SHOW MENU BAR
Subtotal

- **サーバー上で使用すべきでないコマンド**

ストアドプロシージャーの中で下記のコマンドを使用しないことを強くお勧めします。これらはサーバーをブロックすることができ、エラーが生じるかもしれません。そしてどのような場合でも正しく実行できません。特別なエラーコードはありません。

ACCEPT
Activated
_o_ADD DATA SEGMENT
After
APPEND DATA TO PASTEBOARD
APPEND TO LIST
Before
BLOB TO DOCUMENT
BLOB to list

BRING TO FRONT
C_GRAPH
CANCEL
CHANGE CURRENT USER
CHANGE PASSWORD
CLEAR LIST
CLEAR PASTEBOARD
Copy list
Count list items
Count screens
Create document(1)
_o_Create resource file(1)
Current form table
Current user
Deactivated
DELETE FROM LIST
DELETE USER
DIALOG
_o_DISABLE BUTTON
DRAG AND DROP PROPERTIES
DRAG WINDOW
Drop position
_o_During
_o_ENABLE BUTTON
ERASE WINDOW
EXPORT DATA(1)
FILTER KEYSTROKE
Find window
Focus object
FONT LIST
_o_Font name
_o_Font number
Form event
FORM FIRST PAGE
FORM Get current page
FORM GET PROPERTIES
FORM GOTO PAGE
FORM LAST PAGE
FORM NEXT PAGE
FORM PREVIOUS PAGE
FORM SET INPUT
FORM SET OUTPUT
Frontmost process
Frontmost window
Get edited text
GET GROUP LIST
GET GROUP PROPERTIES
GET HIGHLIGHT
GET LIST ITEM
GET LIST ITEM PROPERTIES
GET LIST PROPERTIES
GET MOUSE
GET PASTEBOARD DATA
GET PICTURE FROM PASTEBOARD
Get text from pasteboard
GET USER LIST

GET USER PROPERTIES
GET WINDOW RECT
Get window title
GOTO OBJECT
GRAPH SETTINGS
HIDE PROCESS
HIDE TOOL BAR
HIDE WINDOW
HIGHLIGHT RECORDS
HIGHLIGHT TEXT
IMPORT DATA(1)
In break
In footer
In header
INSERT IN LIST
_o_INVERT BACKGROUND
Is a list
Is user deleted
Keystroke
List item parent
List item position
LIST TO BLOB
Load list
MAXIMIZE WINDOW
Menu bar height
Menu bar screen
MINIMIZE WINDOW
Modified
New list
Next window
OBJECT GET COORDINATES
OBJECT MOVE
OBJECT SET LIST BY NAME
OBJECT SET COLOR
OBJECT SET ENTERABLE
OBJECT SET FILTER
OBJECT SET FORMAT
OBJECT SET RGB COLORS
OBJECT SET TITLE
OBJECT SET VISIBLE
Old
Open document(1)
Open resource file(1)
ORDER BY(2)
Outside call
Pasteboard data size
Pop up menu
POST CLICK
POST EVENT
POST KEY
QUERY BY FORMULA(2)
QUERY(2)
REDRAW
_o_REDRAW LIST
REDRAW WINDOW
REGISTER CLIENT

REJECT
SAVE LIST
SCREEN COORDINATES
SCREEN DEPTH
Screen height
Screen width
Select folder
SELECT LIST ITEMS BY POSITION
SELECT LIST ITEMS BY REFERENCE
SELECT LOG FILE
Selected list items
Self
SET CURSOR
SET FIELD TITLES
Set group properties
SET LIST ITEM
SET LIST ITEM PROPERTIES
SET LIST PROPERTIES
SET PICTURE TO PASTEBOARD
SET SCREEN DEPTH
SET TABLE TITLES
SET TEXT TO PASTEBOARD
SET TIMER
Set user properties
SET WINDOW RECT
Shift down
SHOW PROCESS
SHOW WINDOW
SORT LIST
User in group
Validate password
Window kind
WINDOW LIST
Window process

(1) 第一引数が空の文字列の場合のみ
(2) シンタックスの結果がダイアログを表示する場合のみ (例: **ORDER BY**([Table]))

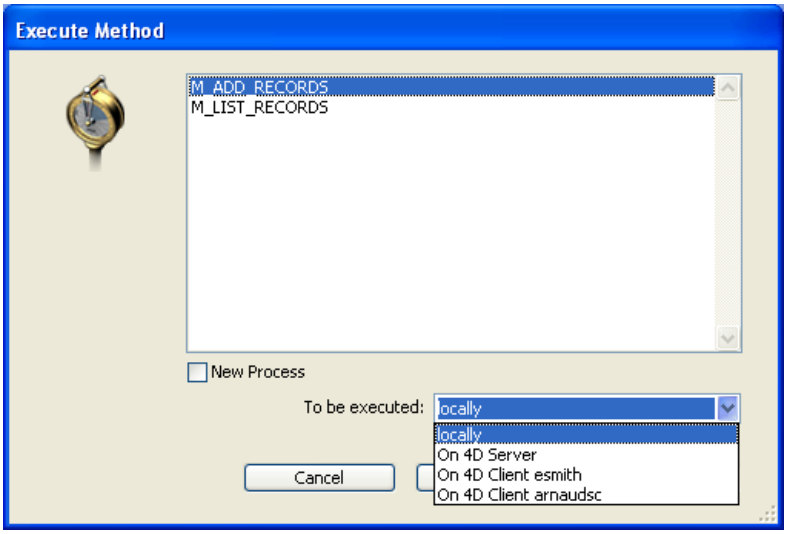
- **サーバー上で効果がないコマンド**

以下のコマンドはサーバー上のストアドプロシージャーで呼び出された場合、効果がありません。特定のエラーコードは返されません。

GRAPH
MESSAGES OFF
MESSAGES ON
SET MENU BAR
SHOW TOOL BAR

ストアドプロシージャを開始する

- 4Dからメソッド実行ダイアログボックスを使用して、手動でストアドプロシージャーを開始できます:



ここからメソッドを4D Serverまたは他の4Dクライアントマシン上で実行できます。このリストに4Dクライアントマシンを表示させるためには、まずそのマシンが登録されていなければならないことに留意してください (**クライアントマシン上でのストアドプロシージャ**と**REGISTER CLIENT**コマンドを参照)。

- また4D上で、**Execute on server**または**EXECUTE ON CLIENT**コマンドを使用してプログラムからストアドプロシージャーを開始できます。.

注: リモート4Dからサーバのストアドプロシージャに、**DELAY PROCESS**、**PAUSE PROCESS**そして**RESUME PROCESS**などのプロセス管理コマンドを使用することはできません。

- 4D Server上で実行されているメソッド (サーバデータベースメソッド、サーバ上で実行属性付きのメソッド、またはストアドプロシージャ) は**Execute on server**、**New process**、または**EXECUTE ON CLIENT**を使用してストアドプロシージャを開始できます。

ストアドプロシージャーとユーザプロセス間のインタープロセス通信について

ストアドプロシージャー間の通信には、次の方法を使用します:

- インタープロセス変数
- ローカルセマフォーまたはグローバルセマフォー
- レコード
- インタープロセスセットおよびインタープロセス命名セレクション
- **GET PROCESS VARIABLE**、**SET PROCESS VARIABLE**、**VARIABLE TO VARIABLE**コマンド

4D Language Referenceマニュアルで、関連する箇所を参照してください。4Dコマンドは、クライアントマシンのスコープ内で動作する場合と同様に、ストアドプ ロシージャーを実行するサーバーまたはクライアントマシンのスコープ内で動作することに注意してください。

注: **CALL PROCESS**および**Outside call**メカニズムは、サーバーマシン上では意味がありません。ストアドプロシージャーには、データ入力のためのユーザーインタフェースがないためです。

さらにもう1 つ重要な機能があります。クライアントユーザープロセス (クライアントマシンで実行されるプロセス) は、**GET PROCESS VARIABLE**、**SET PROCESS VARIABLE**、**VARIABLE TO VARIABLE**コマンドを使用して、ストアドプロシージャーのプロセス変数 (*) を読み込んだり、書き込むことができます

(*) サーバーマシンのインタープロセス変数も同様

重要: **GET PROCESS VARIABLE**、**SET PROCESS VARIABLE**、**VARIABLE TO VARIABLE**コマンドを使用して行う“マシン間”のプロセス通信は、クライアントからサーバーに対してのみ可能です。ストアドプロシージャーの変数を読み込んだり、書き込んだりするのは常にクライアントのプロセスです。

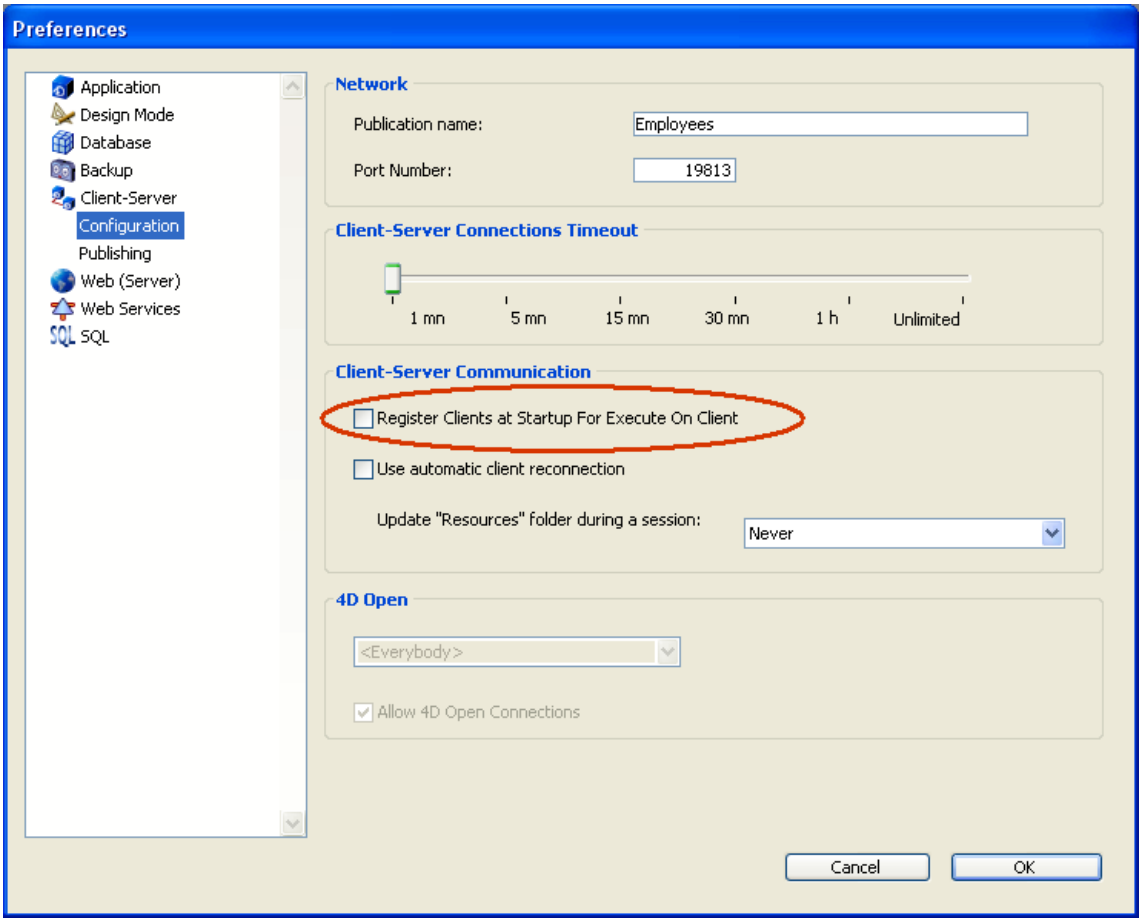
クライアントマシン上でのストアドプロシージャ

ストアドプロシージャを1つあるいは複数の4Dマシン上で実行できます。クライアント上のストアドプロシージャは、サーバ上のそれとおなじように実行されますが、クライアント上ではデータ入力を行うことができます。詳細は**ストアドプロシージャ**を参照してください。

サーバまたは他のクライアントマシンによって起こされた、どのようなクライアントマシンで実行されているストアードプロシージャも、明示的にこのセッションに登録されなければなりません。4D クライアントを登録するには2つの方法があります。接続時に自動登録またはプログラミングによる登録です。

4D Server へ接続する各4Dクライアントマシンを自動的に登録する

データベース設定のクライアント-サーバーページ中、ネットワークオプションタブの"Execute On Clientのために起動時にクライアント登録"チェックボックスを利用します。



このオプションにチェックを入れると、データベースに接続した各4Dクライアントマシンは、ストアードプロシージャを実行することができるクライアントとして、4D Server から自動的に参照されます。クライアントマシンに応じて名前がつけられた4Dクライアントタイプのプロセスは、サーバ上に作成されます。また対応するプロセスが、各クライアントマシン上に作成されます。

プログラミングによる4Dクライアントの登録

プログラミングを使って1つまたは複数の4Dクライアントマシンを登録することができます。この方法では登録に必要なクライアントマシンを選択し、登録名を決めることができます。
"プロセス"テーマの**REGISTER CLIENT**コマンドで、どのような名前でもクライアントマシンを登録することができます。

4Dクライアントの登録解除

クライアントマシンが登録された方法にかかわらず、"プロセス"テーマの**UNREGISTER CLIENT** コマンドで現在のセッションから指定したクライアントを登録解除することができます。クライアントごとに付けられた名前の登録プロセスはサーバマシンのユーザプロセスグループから消え、クライアント上の対応するプロセスも終了されます。

注: **GET REGISTERED CLIENTS**コマンドを使用して、セッションに登録されたクライアントのリストと実行待ちのメソッド数であるタスク割り当てを取得できます。

これらのコマンドの詳細は、4D Language Referenceマニュアルを参照してください。

SPベースの読み込み (例題)

次の例は、クライアント/サーバアーキテクチャにおいて、データの読み込みを飛躍的に高速化する方法を示しています。***Regular Import***メソッドでは、クライアント側で**IMPORT TEXT**コマンドを使用して、レコードの読み込みに要する時間を調べています:

```
  ` Regular Importプロジェクトメソッド
$vhDocRef:=Open document("")
If(OK=1)
   CLOSE DOCUMENT($vhDocRef)
   INPUT FORM([Table1];"Import")
   $vhStartTime:=Current time
   IMPORT TEXT([Table1];Document)
   $vhEndTime:=Current time
   ALERT("It took "+String(0+($vhEndTime-$vhStartTime))+" seconds.")
End if
```

通常のデータ読み込みでは、4Dはテキストファイルを解析した後、各レコードに対して新しいレコードを作成し、読み込んだデータをフィールドに入力し、レコードをサーバマシンに送信してデータベースに追加します。そのため、大量の要求がネットワーク上でやり取りされることになります。この操作を最適化する方法の1 つとして、ストアドプロシージャを使用し、サーバマシンでローカルにこの作業を実行するやり方があります。クライアントマシンではドキュメントファイルをBLOB にロードした後、ストアドプロシージャを開始し、引数としてこのBLOB を渡します。ストアドプロシージャではこのBLOB をサーバマシンのディスク内のドキュメントファイルに保存し、このドキュメントファイルをローカルに読み込みます。ネットワーク要求がほとんどなくなるため、データ読み込みはローカルに (ローカルバージョンの4Dと比較可能な速度で) 処理されます。

次に示すのは、***CLIENT IMPORT***プロジェクトメソッドです。このプロジェクトメソッドはクライアントマシンで実行され、後述の***SERVER IMPORT***ストアドプロシージャを呼び出します:

```
  ` CLIENT IMPORT Project Method
  ` CLIENT IMPORT ( Pointer ; Text)
  ` CLIENT IMPORT ( -> [Table] ; Input form )

C_POINTER($1)
C_TEXT($2)
C_TIME($vhDocRef)
C_BLOB($vxData)
C_LONGINT(spErrCode)

  ` 読み込むドキュメントを選択
$vhDocRef:=Open document("")
If(OK=1)
 `ドキュメントが選択されたら開いたままにしない
   CLOSE DOCUMENT($vhDocRef)
   $vhStartTime:=Current time
  ` メモリにロード
   DOCUMENT TO BLOB(Document;$vxData)
   If(OK=1)
  ` ドキュメントをBLOBに読み込めたら
  ` ストアドプロシージャを開始し、サーバマシン上でデータを読み込む
      $spProcessID:=Execute on server("SERVER IMPORT";32*1024;
      "Server Import Services";Table($1);$2;$vxData)
  ` この時点で、このプロセス内でこのBLOBはもう必要ない
      CLEAR VARIABLE($vxData)
  ` ストアドプロシージャの処理終了を待つ
      Repeat
         DELAY PROCESS(Current process;300)
         GET PROCESS VARIABLE($spProcessID;spErrCode;spErrCode)
         If(Undefined(spErrCode))
```

```
   ` Note: ストアドプロシージャが自身のspErrCode変数のインスタンスをまだ初期化していない場合、
   ` 未定義変数が返されるかもしれません。
            spErrCode:=1
         End if
      Until(spErrCode<=0)
   ` ストアドプロシージャに結果を取得したことを伝える
      spErrCode:=1
      SET PROCESS VARIABLE($spProcessID;spErrCode;spErrCode)
      $vhEndTime:=Current time
      ALERT("It took "+String(0+($vhEndTime-$vhStartTime))+" seconds.")
   Else
      ALERT("There is not enough memory to load the document.")
   End if
End if
```

以下はストアドプロシージャとして実行される***SERVER IMPORT***プロジェクトメソッドです:

```
   ` SERVER IMPORT Project Method
   ` SERVER IMPORT (Long ; Text; BLOB )
   ` SERVER IMPORT (Table number ; Input form ; Import Data )

C_LONGINT($1)
C_TEXT($2)
C_BLOB($3)
C_LONGINT(spErrCode)

   ` 処理がまだ終わっていない, spErrCodeを1に設定
spErrCode:=1
$vpTable:=Table($1)
INPUT FORM($vpTable->;$2)
$vsDocName:="Import File "+String(1+Random)+".txt"
DELETE DOCUMENT($vsDocName)
BLOB TO DOCUMENT($vsDocName;$3)
IMPORT TEXT($vpTable->;$vsDocName)
DELETE DOCUMENT($vsDocName)
   ` 処理が終了した, spErrCodeを0に設定
spErrCode:=0
   ` リクエストもとのクライアントからの通知を待つ
Repeat
   DELAY PROCESS(Current process;1)
Until(spErrCode>0)
```

Note: ***On Windows*** プロジェクトメソッドは4D Language Referenceマニュアルの**C_TEXTシステムドキュメント**で紹介しています。

これら2 つのプロジェクトメソッドがデータベースに実装された後で、例えば次のように、“ストアドプロシージャベース”のデータ読み込みを実行できます:

```
CLIENT IMPORT(->[Table1];"Import")
```

ベンチマークテストを何度か行うと、このメソッドを使用した場合には、通常のデータ読み込みの場合と比べて最高で60倍も速くレコードを読み込めることがわかります。

SPベースのサービス (例題)

SPベースの読み込み (例題)で説明している例では、ストアドプロシージャは、データ読み込み処理が要求されるたびに開始され、終了されています。次の例では、ストアドプロシージャはサーバデータベースが起動されると自動的に開始され、サーバデータベースに接続している任意の4Dから随時に終了する、または再開することができます。ストアドプロシージャは実行されるとすぐに、データベースに接続しているクライアントから送られる複数の要求に対して、非同期的に応答することができるようになります。

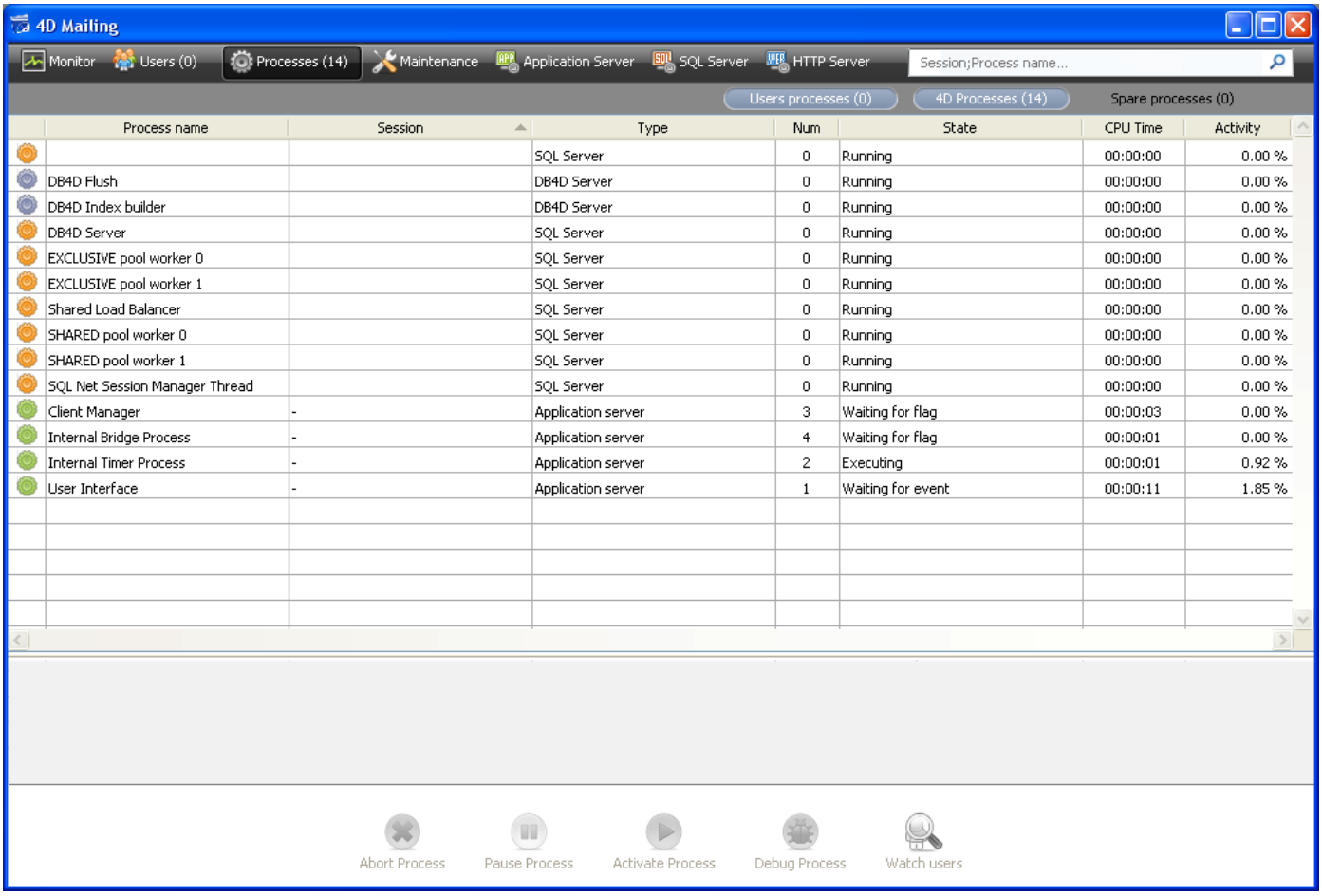
SPベースの読み込み (例題)の節では、4D Server で提供される既存のサービスを飛躍的に最適化する方法について説明していますが、この例ではすべての4Dクライアントマシンで使用できる、新しいサービスやカスタムサービスを実現する方法について説明します。さらにこの例は、独自のサービスを実現するためのテンプレートとしても使用することができます。

ストアドプロシージャを自動起動する

ストアドプロシージャは**On Server Startupデータベースメソッド**によって自動的に開始されます:

```
` On Server Startup Database Method
START SP SERVICES
```

On Server Startupデータベースメソッドが、***SP SERVICES***プロジェクトメソッドをストアドプロシージャとして開始するため、実際にクライアントがサーバデータベースに接続しているかどうかに関わらず、4D Server でデータベースが開かれるとすぐに***SP SERVICES***が実行されます。次の図では、クライアントがまだ接続していない状態で、ストアドプロシージャが実行されている様子が4D Server の管理ウィンドウに表示されています。



Process name	Session	Type	Num	State	CPU Time	Activity
		SQL Server	0	Running	00:00:00	0.00 %
DB4D Flush		DB4D Server	0	Running	00:00:00	0.00 %
DB4D Index builder		DB4D Server	0	Running	00:00:00	0.00 %
DB4D Server		SQL Server	0	Running	00:00:00	0.00 %
EXCLUSIVE pool worker 0		SQL Server	0	Running	00:00:00	0.00 %
EXCLUSIVE pool worker 1		SQL Server	0	Running	00:00:00	0.00 %
Shared Load Balancer		SQL Server	0	Running	00:00:00	0.00 %
SHARED pool worker 0		SQL Server	0	Running	00:00:00	0.00 %
SHARED pool worker 1		SQL Server	0	Running	00:00:00	0.00 %
SQL Net Session Manager Thread		SQL Server	0	Running	00:00:00	0.00 %
Client Manager	-	Application server	3	Waiting for flag	00:00:03	0.00 %
Internal Bridge Process	-	Application server	4	Waiting for flag	00:00:01	0.00 %
Internal Timer Process	-	Application server	2	Executing	00:00:01	0.92 %
User Interface	-	Application server	1	Waiting for event	00:00:11	1.85 %

ストアドプロシージャの開始と終了

START SP SERVICES プロジェクトメソッドは以下のとおりです。:

```
    ` START SP SERVICES プロジェクトメソッド
<>vlSPServices:=Execute on server("SP SERVICES";32*1024;"SP SERVICES";*)
```

Execute on serverコマンドはサーバマシンから呼ばれたときは**New process**と同様に動作するので、同じメソッド (***START SP SERVICES***) をサーバマシンおよびクライアントマシンから使用して、ストアドプロシージャとして***SP SERVICES***メソッドをサーバマシン上で実行できます。

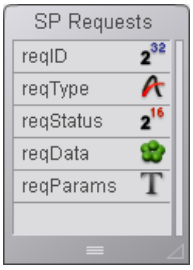
STOP SP SERVICESプロジェクトメソッドは***SP SERVICES***プロジェクトメソッドに停止するよう通知します。

```
    ` STOP SP SERVICES Project Method
SET PROCESS VARIABLE(<>vlSPServices;vbStopSPServices;True)
```

SP SERVICESプロジェクトメソッドが開始されると、vbStopSPServicesプロセス変数がFalseに設定され、このブール変数がTrueになるまでループします。コマンド**SET PROCESS VARIABLE**を使用して、サーバあるいはクライアント上で実行されているユーザプロセスからvbStopSPServices変数の値を変更し、ストアドプロシージャを停止させられます。

ストアドプロシージャとの通信

ストアドプロシージャは、任意の時間に任意の順序で非同期的にクライアントのリクエストを受信し、応答できる必要があります。この通信を保証する簡単な方法はテーブルを使用することです。



[SP Requests] テーブルには、次のフィールドが含まれています:

- [SP Requests]reqIDは、**Sequence number**コマンドを使用して設定されます。このフィールドによって各リクエストを識別します。
- [SP Requests]reqTypeはリクエストのタイプを示します。
- [SP Requests]reqStatusは次の値のうちいずれかになります:

値	説明
1	リクエストは送られたが、まだ処理されていない。
0	リクエストは正常に処理された。
< 0	リクエストは処理されたが、エラーが発生した。

Note: これらの値は、この例題のため任意に選ばれたものであり、4D から与えられた値ではありません。

- [SP Requests]reqDataは、リクエストデータを格納しているBLOB です。リクエスト元から送られたデータやストアドプロシージャからリクエスト元に返されるデータが含まれています。
- [SP Requests]reqParamsには、オプションとしてリクエスト元がストアドプロシージャに送った引数の値が含まれています。

なぜテーブルを使用するのか?

クライアントプロセスとストアドプロシージャの間の通信は**GET PROCESS VARIABLE**、**SET PROCESS VARIABLE**、**VARIABLE TO VARIABLE**コマンドを使用して実現できます。**SPベースの読み込み (例題)**の節や、前述の***STOP SP SERVICES***プロジェクトメソッドで使用したソリューションがこの例です。

今回の場合は、ストアドプロシージャがさまざまな量のデータを送受信できるようにシステムが設定されていなければなりません。テキスト配列やピクチャ配列等の配列を使用することもできますが、次の2つの理由からテーブルを使用します:

- レコードを使用してリクエストを処理するアルゴリズムの方が、より容易に実装できます。クライアントマシンからリ

クエストを送る処理は、テーブルにリクエストを追加する処理だけで構成されています。ストアドプロシージャ内からリクエストに応答する処理は、このリクエストレコードを修正する処理だけで構成されています。

- リクエストはテーブルに格納されるため、ディスク上に保存されます。したがって、リクエストは (配列に格納されるデータの場合とは異なり) メモリには保持されず、リクエストのサイズが大きい場合でも問題にはなりません。

クライアントマシンからリクエストを送る

Client post request プロジェクトメソッドは、リクエストを送るための汎用的なメソッドです:

```
   ` Client post request プロジェクトメソッド
   ` Client post request ( String { ; Text } ) -> Long
   ` Client post request ( Request type { ; Parameters } ) -> Request ID
CREATE RECORD([SP Requests])
[SP Requests]reqID:=Sequence number([SP Requests])
[SP Requests]reqType:=$1
[SP Requests]reqStatus:=1
If(Count parameters>=2)
   [SP Requests]reqParams:=$2
End if
SAVE RECORD([SP Requests])
$0:=[SP Requests]reqID
```

このメソッドからリクエストID 番号が返されますが、**Sequence number**コマンドを使用することにより、この番号は必ずユニークになります。レコードが[SP Requests]テーブルに追加された後、クライアントはフィールド[SP Requests]reqStatus を調べ、ストアドプロシージャが完全にリクエストを処理するまで待機します。

リクエストステータスの検査とクライアントマシンでの結果の取得

Client get resultプロジェクトメソッドは、リクエストステータスを調べるための汎用的なメソッドです。前述したように、[SP Requets]redStatusフィールドが1以外の値になるとすぐに、クライアントはストアドプロシージャがリクエストを処理したことが (成功しても失敗しても) 分かります。

```
   ` Client get result プロジェクトメソッド
   ` Client get result ( Long ; ->BLOB {; Long } ) -> Long
   ` Client get result ( Request ID ; ->Data {; Delay } ) -> Error Code
C_LONGINT($0;$1;$vlDelay)
$0:=1
$vlDelay:=0
If(Count parameters>=3)
   $vlDelay:=$3
End if
READ ONLY([SP Requests])
Repeat
   QUERY([SP Requests];[SP Requests]reqID=$1)
   If(Records in selection([SP Requests])>0)
      If([SP Requests]reqStatus?#?1)
         $2->:=[SP Requests]reqData
         READ WRITE([SP Requests])
         While(Locked([SP Requests]))
            DELAY PROCESS(Current process;$vlDelay)
            LOAD RECORD([SP Requests])
         End while
         DELETE RECORD([SP Requests])
         $0:=[SP Requests]reqStatus
      End if
   Else
   ` リクエストレコードが失われた!
   ` これは発生すべきではないが、とにかくerrorを-2に設定 (任意の値)
      $0:=-2
   End if
   ` リクエストはまだ処理されていない
   If($0=1)
      WAITING LOOP($vlDelay)
   End if
```

```
   Until($0?#?1)
   READ ONLY([SP Requests])
```

リクエストがストアドプロシージャにより正常に処理された場合、このメソッドはレコードからBLOB へ結果をコピーします(ある場合)。BLOB へのポインタは引数として渡されます。次に、呼び出し元であるメソッドでリクエストタイプに応じ、BLOB データが解析されます。リクエストの処理が終了したら、[SP Requests]レコードの削除を行うのはクライアントである点に注意してください。

小さな***WAITING LOOP***プロジェクトメソッドは、指定したtick数が経過するまでループします:

```
   ` WAITING LOOP Project Method
   ` WAITING LOOP ( Long )
   ` WAITING LOOP ( Delay in ticks )
C_LONGINT($1)
$vlStartTicks:=Tickcount
Repeat
   IDLE
Until((Tickcount-$vlStartTicks)>=$1)
```

Reminder: ***WAITING LOOP*** プロジェクトメソッドは、クライアントマシンのユーザ環境プロセスからリクエストされた場合でも、必要なだけ時間を待つために使用されています。

ストアドプロシージャとサブルーチン

SP SERVICESプロジェクトメソッドは、サーバマシン上でストアドプロシージャとして実行されるメソッドです。疑似コードを次に示しますが、総体的なアーキテクチャは簡単です:

```
   "stop" 変数の初期化
   以下繰り返す
      [SP Requests]reqStatus フィールドが1であるリクエストを検索
      リクエストごとに
         リクエストのタイプに応じて、サブルーチンを呼び出し、
            [SP Requests]reqData フィールドに結果を格納する
         リクエストのステータスを変更し、処理の終了をクライアントに通知
      リクエストごとの繰り返しここまで
      再開始するまで少々の時間停止する
   "stop" 変数がTrueになるまで
```

以下は実際のコードです:

```
   ` SP SERVICES プロジェクトメソッド
   ` ストアドプロシージャの開始
vbStopSPServices:=False
   ` ストアドプロシージャはテーブルに対し読み書きアクセスを必要としない
READ ONLY(*)
   ` ただし[SP Requests] テーブルを除く
READ WRITE([SP Requests])
Repeat
   ` まだ処理していないリクエストを検索
   QUERY([SP Requests];[SP Requests]reqStatus=1)
   ` これらのリクエストをひとつづつ処理
   For($vlRecord;1;Records in selection([SP Requests]))
   ` リクエストレコードがロックされていれば、ロック解除まで待つ
      While(Locked([SP Requests]))
   ` 再試行まで1秒待つ
         DELAY PROCESS(Current process;60)
   ` 読み書きアクセスを試行
         LOAD RECORD([SP Requests])
      End while
   ` 処理が成功したと仮定する
      [SP Requests]reqStatus:=0
      Case of
         :([SP Requests]reqType="Server Information")
            SP DO SERVER INFORMATION
         :([SP Requests]reqType="Volume List")
```

```
            SP DO VOLUME LIST
         :([SP Requests]reqType="Browse Directory")
            SP DO BROWSE DIRECTORY([SP Requests]reqParams)
   ` ...
   ` 他のリクエストタイプをここに追加可能!
   ` ...
         Else
   ` 未知のリクエストタイプ, エラー -1を返す (任意の値)
            [SP Requests]reqStatus:=-1
      End case
   ` リクエストステータスが1でないようにする
   ` (サブルーチンが1にしてしまった場合)
      If([SP Requests]reqStatus=1)
         [SP Requests]reqStatus:=-3
      End if
   ` リクエストレコードを更新
      SAVE RECORD([SP Requests])
   ` 次の未処理レコードに移動
      NEXT RECORD([SP Requests])
   End for
   ` 最後に処理したレコードをアンロード
   UNLOAD RECORD([SP Requests])
   ` 再び処理を開始する前に1秒待つ
   DELAY PROCESS(Current process;60)
   ` 停止を指示されるまでループする
Until(vbStopSPServices)
```

SP SERVICESプロジェクトメソッドは、データベースに新しいサービスを実現するためのテンプレートとして使用することができます。この節では、***SP DO SERVER INFORMATION***サブルーチンおよび***SP DO VOLUME LIST***サブルーチンの詳細について説明します。***SP DO BROWSE DIRECTORY*** ([SP Requests]reqParamsフィールドに納めて送られた引数を引数として取得するサブルーチン) の詳細については、このドキュメントでは説明されていません。

リクエストのタイプによって***SP SERVICES***プロジェクトメソッドは、結果データを[SP Requests]reqDataフィールドに保存する処理を行うサブルーチンを呼び出します。レコードの保存やリクエストステータスの変更は、***SP SERVICES***プロジェクトメソッドによって実行されます。

次に示すのは***SP DO SERVER INFORMATION***サブルーチンです。このサブルーチンはサーバ関連の情報をBLOB に保存します。別のプロジェクトメソッドを使用して、クライアントマシン上でBLOB データを取り出します。

```
   ` SP DO SERVER INFORMATION プロジェクトメソッド
TEXT TO BLOB(Application version(*);[SP Requests]reqData;UTF8 C string)
TEXT TO BLOB(Structure file;[SP Requests]reqData;UTF8 C string;*)
TEXT TO BLOB(Data file;[SP Requests]reqData;UTF8 C string;*)
PLATFORM PROPERTIES($vlPlatform;$vlSystem;$vlMachine)
VARIABLE TO BLOB($vlPlatform;[SP Requests]reqData;*)
VARIABLE TO BLOB($vlSystem;[SP Requests]reqData;*)
VARIABLE TO BLOB($vlMachine;[SP Requests]reqData;*)
```

次に示すのは***SP DO VOLUME LIST***サブルーチンです。このサブルーチンは、ボリューム関連の情報をBLOB に保存します。別のプロジェクトメソッドを使用して、クライアントマシン上でBLOB データを取り出します。

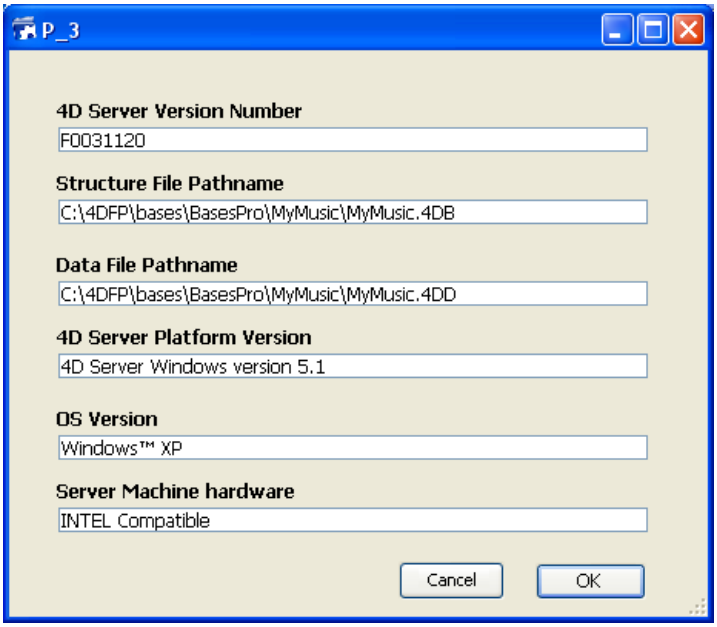
```
   ` SP DO VOLUME LIST プロジェクトメソッド
VOLUME LIST($asVName)
$vlSize:=Size of array($asVName)
REAL($arVSize;$vlSize)
REAL($arVUsedSpace;$vlSize)
REAL($arVFreeSpace;$vlSize)
For($vlElem;1;$vlSize)
   VOLUME ATTRIBUTES($asVName{$vlElem};$arVSize{$vlElem};$arVUsedSpace{$vlElem}
   ;$arVFreeSpace{$vlELem})
End for
VARIABLE TO BLOB($asVName;[SP Requests]reqData)
VARIABLE TO BLOB($arVSize;[SP Requests]reqData;*)
VARIABLE TO BLOB($arVUsedSpace;[SP Requests]reqData;*)
VARIABLE TO BLOB($arVFreeSpace;[SP Requests]reqData;*)
```

サーバ情報をクライアントマシン上に表示する

汎用的な***Client post request***と***Client get result***プロジェクトメソッドを使用して、***M_SERVER_INFORMATION***プロジェクトメソッドはストアドプロシージャより返されたサーバ情報をクライアントマシン上に表示します。このメソッドは、メニューに割り当てる、あるいはボタンのオブジェクトメソッドで呼び出してもいいでしょう。

```
  ` M_SERVER_INFORMATION
C_BLOB(vxData)
C_LONGINT($vlReqID;$vlErrCode;$vlOffset)
  ` リクエストを送信
$vlReqID:=Client post request("Server Information")
  ` リクエストステータスを見て、結果を取得
$vlErrCode:=Client get result($vlReqID;->vxData;60)
  ` リクエストが正しく処理されていれば、結果を表示
If($vlErrCode=0)
  ` BLOBから情報を取り出し
   $vlOffset:=0
   vsServerVersion:=BLOB to text(vxData;UTF8 C string;$vlOffset)
   vsStructureFile:=BLOB to text(vxData;UTF8 C string;$vlOffset)
   vsDataFile:=BLOB to text(vxData;UTF8 C string;$vlOffset)
   BLOB TO VARIABLE(vxData;$vlPlatform;$vlOffset)
   BLOB TO VARIABLE(vxData;$vlSystem;$vlOffset)
   BLOB TO VARIABLE(vxData;$vlMachine;$vlOffset)
  ` プラットフォームプロパティを解析
   vs4DPlatform:="Unknown 4D Server Version"
   vsSystem:="Unknown System Version"
   vsMachine:="Unknown Machine"
  `...
  ` $vlSystem と $vlMachineを取得するコードをここに記述
  ` (PLATFORM PROPERTIES コマンドの例題参照)
  ` ...
  ` 結果を表示
   DIALOG([SP Requests];"SERVER INFORMATION")
Else
   ALERT("Request error "+String($vlErrCode))
End if
  ` BLOBは必要ない
CLEAR VARIABLE(vxData)
```

以下は実行された[SP Requests];"SERVER INFORMATION"フォームです:



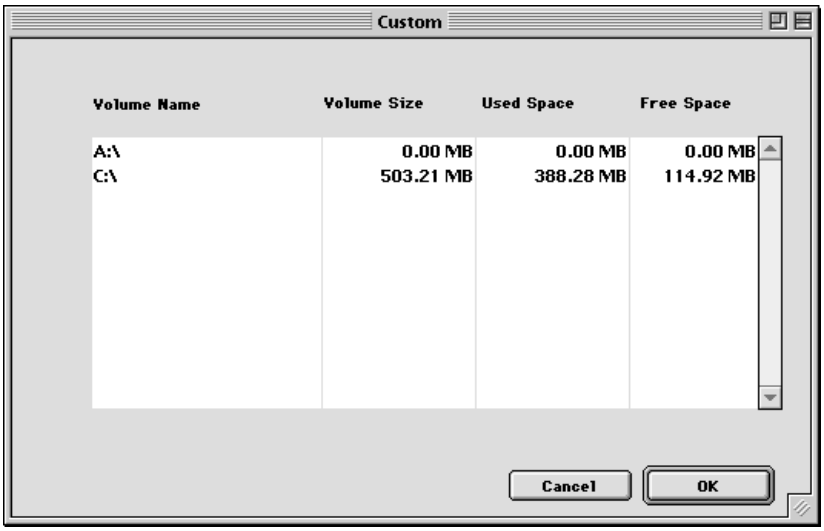
サーバマシンのボリューム一覧をクライアントマシン上に表示する

汎用的な***Client post request***と***Client get result***プロジェクトメソッドを使用して、***M_SERVER_INFORMATION***プロ

ジェクトメソッドはストアドプロシージャより返されたボリューム一覧をクライアントマシン上に表示します。このメソッドをメニューに割り当てたり、あるいはボタンのオブジェクトメソッドで呼び出してもいいでしょう:

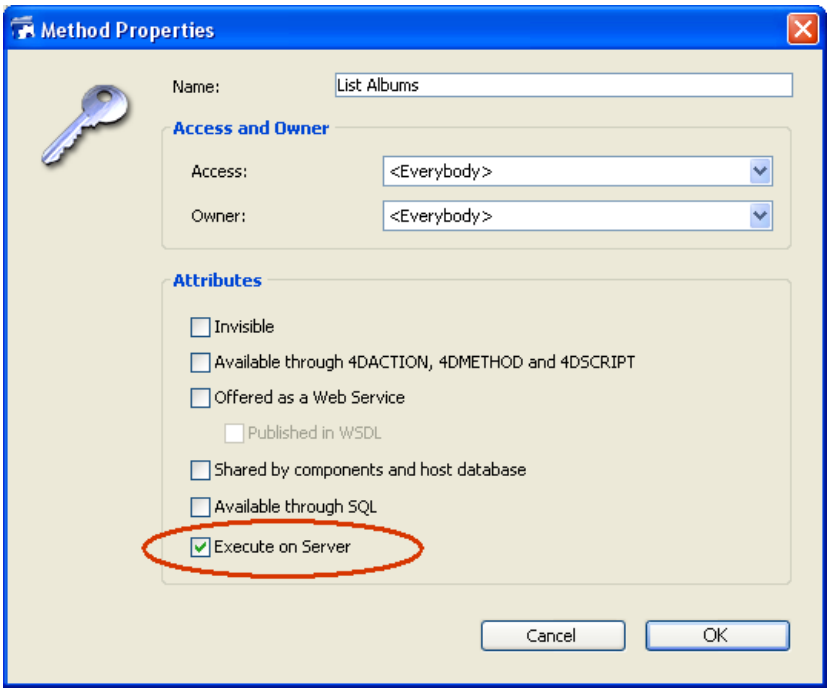
```
  ` M_SERVER_VOLUMES
C_BLOB(vxData)
  ` リクエストを送信
$vlReqID:=Client post request("Volume List")
  ` リクエストステータスを見て、結果を取得
$vlErrCode:=Client get result($vlReqID;->vxData;120)
  ` リクエストが正しく処理されていれば、結果を表示
If($vlErrCode=0)
  ` BLOBから情報を取り出し
   $vlOffset:=0
   BLOB TO VARIABLE(vxData;asVName;$vlOffset)
   BLOB TO VARIABLE(vxData;arVSize;$vlOffset)
   BLOB TO VARIABLE(vxData;arVUsedSpace;$vlOffset)
   BLOB TO VARIABLE(vxData;arVFreeSpace;$vlOffset)
   For($vlElem;1;Size of array(arVSize))
  ` バイトをMBに変換
      arVSize{$vlElem}:=arVSize{$vlElem}/1048576
      arVUsedSpace{$vlElem}:=arVUsedSpace{$vlElem}/1048576
      arVFreeSpace{$vlElem}:=arVFreeSpace{$vlElem}/1048576
   End for
  ` 結果を表示
   DIALOG([SP Requests];"VOLUME LIST")
Else
   ALERT("Request error "+String($vlErrCode))
End if
  ` BLOBは必要ない
CLEAR VARIABLE(vxData)
```

以下は実行された[SP Requests];"VOLUME LIST"フォームです:



サーバー上で実行属性

"サーバー上で実行"プロジェクトメソッド属性はメソッドプロパティダイアログや属性の一括設定ダイアログを使用して設定できます:



このオプションがチェックされていると、プロジェクトメソッドは、それが呼ばれる方法に関わらず、常にサーバー上で実行されます。

注: この属性はクライアント/サーバーモードの4Dアプリケーションでのみ考慮されます。

実行コンテキスト

この属性が選択されている場合、そのプロジェクトメソッドの実行コンテキストはトリガーのそれと似たようなものになります (**4D Serverと4Dランゲージ**参照)。つまり、サーバー上で実行されるメソッドは、レコードロックやトランザクションについてクライアント側の対応する同じデータベースコンテキストを共有しますが、ランゲージコンテキスト (プロセス変数、セット、カレントセレクション) は共有しません。トリガーと異なり、サーバー上で実行されるメソッドはクライアントコンテキストのカレントレコードを共有しない点に留意してください (トリガーはトリガーテーブルのカレントレコードを共有します)。
メソッドのすべての引数 ($1, $2等) はサーバに送信され、$0が (使用されていれば) クライアントに返されます。

Execute on serverコマンドと異なり、このオプションはサーバ上にプロセスを作成しません。4D Serverは、実行をリクエストしたクライアントプロセスに対応するサーバ側のクライアントプロセスを使用します。
さらにこのオプションは、普通のメソッド呼び出しのように、引数の交換が両方向で自動化されるので、サーバにメソッドの実行を移譲することが簡単になります。
Execute on serverコマンドの場合、これは非同期で動作するため、追加のプログラミングと、結果を読み込むための待ち合わせが必要です。

使用できるコマンド

"サーバ上で実行"属性を持つメソッドは、4Dランゲージコマンドの利用においては、ストアドプロシージャと同じルールが適用されます。サーバ上ではいくつかのコマンドの実行は許可されず、いくつかは推奨されません。詳細は**ストアドプロシージャ**"(サーバ上で実行される) ストアドプロシージャが行わないこと"を参照してください。

ポインター

変数へのポインタ (変数、配列、または配列要素) を渡した場合、参照先の値もサーバーに送信されます。サーバー上でメソッドにより参照先の値が更新されると、更新された値がクライアントに返され、クライアント側の対応する変数が更新されます。
テーブルやフィールドのポインタは参照として送られます (テーブル番号、フィールド番号)。カレントのレコード値は自動では交換されません。

注: このオプションはインタプリターモードとコンパイルモードで同じように動作します。

例題

以下は"サーバ上で実行"属性が設定されたプロジェクトメソッドのコードです:

```
C_POINTER($1)  `テーブルへのポインタ
C_POINTER($2)  `フィールドへのポインタ
C_POINTER($3)  `配列へのポインタ
C_TEXT($4)  `検索する値
C_LONGINT($0)  `結果

  `検索を行いレコード毎の結果を返す
QUERY($1->;$2->=$4)
While(Not(End selection($1->)))
   APPEND TO ARRAY($3->;myFormula($1))
   NEXT RECORD($1->)
End while
UNLOAD RECORD($1->)
$0:=Records in selection($1->)
```

クライアント側で、メソッドは以下のように呼び出されます:

```
ARRAY TEXT(myArray;0)
$vlnum :=MyAppli(->[Table_1] ;->[Table_1]Field_1 ;->myArray;"to find")
```