4th Dimension 2003

アップグレード

Windows® and Mac™OS

4D

4th Dimension 2003 アップグレード

Windows® and Mac™ OS

このマニュアルに記載されている事項は、将来予告なしに変更されることがあり、いかなる変更に関しても4D SAは一切の責任を負いかねます。このマニュアルで説明されるソフトウェアは、本製品に同梱のLicense Agreement（使用許諾契約書）のもとでのみ使用することができます。

注意

このソフトウェアの使用に際し、本製品に同梱のLicense Agreement（使用許諾契約書）に同意する必要があります。ソフトウェアを使用する前に、License Agreementを注意深くお読みください。

目次

4th Dimension 2003へようこそ！4th Dimension 2003は4th Dimensionおよび4D Server開発環境の最新バージョンです。

このバージョンから、新しいバージョン番号システムが採用され、リリース年度に基づいてメインバージョン番号が付けられるようになりました。これにより、プロダクトラインのバージョン管理は、よりシンプルに、わかりやすくなります。

さらに、4th Dimension 2003では数多くの新しい機能や最適化が提供されています。

このマニュアルについて

本マニュアルは、バージョン2003の4th Dimensionおよび4D Serverに導入された新しい機能と変更点についてすべて詳しく説明します。マニュアルは、以下の章に分かれています。

■ **デザインモード**：この章では、4Dアプリケーションの開発環境に関する新機能および変更点について説明します。データベースの新しいオープン方法、「環境設定」ダイアログボックス、ロックされたデータファイルの検出、メニュー管理の変更点、4Dエクスプローラからの HTML ドキュメントへのアクセス、新しい「メソッドエディタ」について説明しています。

■ **ユーザモード**：この章では、データ読み込みと書き出し時の XML フォーマットのサポートについて説明します。

■ **コンパイラとアプリケーションビルダ**：この章では、新しいデータベースコンパイル機能について説明します。この機能は、今までは4D Compilerで提供されていましたが、4Dアプリケーションに統合されるようになります。さらに、新しいアプリケーション構築機能についても説明しています。

■ **Web サービス**：この章では、サーバおよびクライアントの両モードにおける4th Dimension 2003の Webサービスサポートについて説明します。

■ **ランゲージ**：この章では、4th Dimension言語に関する新機能および変更点（新しいコマンドや変更されたコマンド）について説明します。

■ Web サーバ：この章では、4D Web サーバの変更点について説明します。特に、4D Client における Web サーバのセットアップについて詳しく述べています。

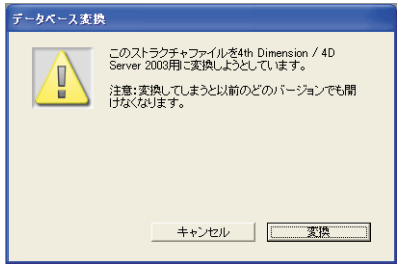
■ 最適化：この章では、4D の内部的なオペレーションレベルにおける変更点（データキャッシュ書き込みの最適化、マウスホイールの導入）について説明します。

前バージョンとの互換性

バージョン 6.7.x や 6.8.x の 4th Dimension や 4D Server を使用して作成された 4D データベースは、4th Dimension 2003 のアプリケーションと完全に互換します（ストラクチャファイルおよびデータファイル）。

しかし、4D 2003 でデータベースを開くとストラクチャファイルが変換されるため、以前のバージョンの 4D では開くことができなくなります。

前バージョンの 4D を使用して作成されたデータベースを 4D 2003 で開くと、ダイアログボックスが表示され、ストラクチャファイルが変換されることを知らせます。



ただし、データファイルは変換されないため、バージョン 6.8.x の 4th Dimension や 4D Server を使用して開くことができます。

動作環境

4Dバージョン2003プロダクトラインのアプリケーションには、次のような動作環境が必要です。

	Windows	MacOS 9	MacOS X
コンピュータ	Pentium IIプロセッサを搭載したPC互換機	Macintosh iMac	
OS	Windows 98SE、 Windows ME、 Windows 2000、 Windows XP	4th Dimension、4D Server、 4D Runtime Classic： OS 9.2以上 4D Clientおよびその他製品： OS 9.1以上 CarbonLib：バージョン1.4以上（バージョン1.5推奨）	バージョン10.1以上
最小メモリ	64 MB	64 MB	128 MB
推奨メモリ	Windows 98/ME： 128 MB Windows 2000/XP： 256 MB	128 MB	256 MB
スクリーン解像度	800*600ピクセル		

注：4Dデータベースでプラグインを使用する場合、最適なパフォーマンスでアプリケーション（4th Dimension、4D Client、4D Runtime、実行形式アプリケーション）を実行するために、さらに2MB程の利用可能なメモリが必要になります。この値はデータベースで使用するプラグインに応じて変わります。

デザインモード　2

4th Dimensionの「デザイン」モードには、数々の新しい機能と変更が加えられています。これらの新機能は、以下の事柄に関するものです。

■ データベースのオープンと作成

■ ロックされたデータファイルの検出

■ 新しい「環境設定」ダイアログボックス

■ メニュー管理

■「検索」コマンドの場所

■ MacOS X上でのQuartzレンダリング

■ エクスプローラからのHTMLドキュメントへのアクセス

■ 新しい「メソッド」エディタ

■ メソッド属性の一括設定

データベースのオープンと作成

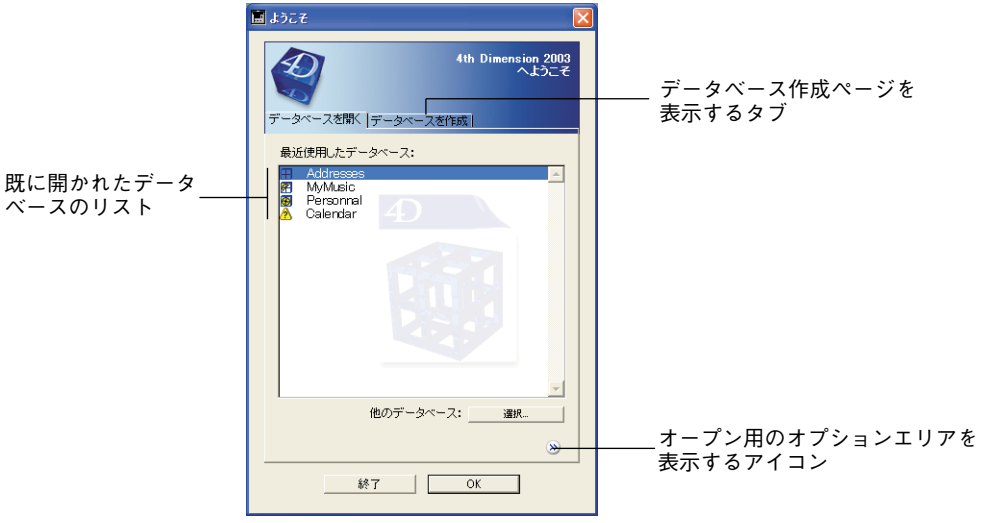
4th Dimension 2003では、データベースのオープンと作成に関する2つの重要な新機能が提供されています。

■ データベースのオープンと作成のためさらにオプションが追加された、よりユーザフレンドリーなダイアログボックス

■ 定義済の“テンプレート”を用いて、いつでも使用可能なデータベースの作成

データベースのオープンと作成用の新しいダイアログボックス

バージョン2003の4th Dimensionと4D Serverでは、データベースのオープンと作成用のダイアログボックスが変更されました。デフォルトでは、次のようなダイアログボックスが表示されます。



作成とオープン

リスト上に表示されていない既存の4Dデータベースを開くには、「選択...」ボタンをクリックして標準のドキュメントを開くダイアログボックスを表示します。

データベースの作成は、このダイアログボックスの2番目のページで実行されるようになりました。これを行うには、「データベースを作成」タブを使用します（このページについては、後述の「データベーステンプレートの使用」の節で説明します）。

データベースのリスト

最近使用したデータベースの一覧は、ファイルのタイプに応じて異なるアイコンで表示されます。

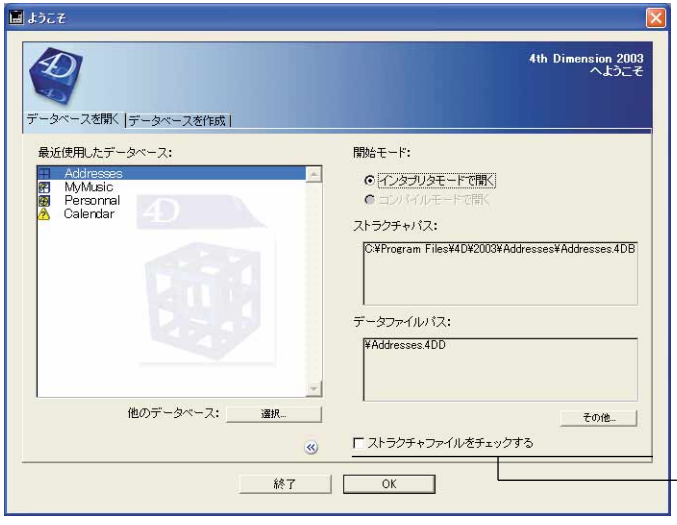
- インタプリタ版だけのストラクチャファイル。
- データベースのインタプリタ版とコンパイル版のコードを含むストラクチャファイル[1]。選択した開始モードオプション（次節参照）に従って、データベースはインタプリタ版またはコンパイル版のいずれかで実行される。
- コンパイル版だけのストラクチャファイル。

1. バージョン2003の4th Dimensionでのコンパイルに関する新しい原則についての詳細は、後述の「コンパイラとアプリケーションビルダ」の章を参照してください。

■ 少なくとも1つのファイル（ストラクチャ、データ、またはリソース）が不足しているデータベース（移動、名称変更、削除...）。

コンパイル版を開く

ダイアログボックスの右側部分（デフォルトでは隠れている）には、作成およびオープン用のオプションがあります。



新しいオプション「開始モード」を利用できます。データベースのインタプリタ版とコンパイル版のコードが両方とも含まれているファイルを選択した場合に、このオプションを使用します。「インタプリタモードで開く」または「コンパイルモードで開く」ラジオボタンを使用して、開始モードを選択することができます。

そのデータベースで前回使用されたモードが、デフォルトのオプションとして選択されます。

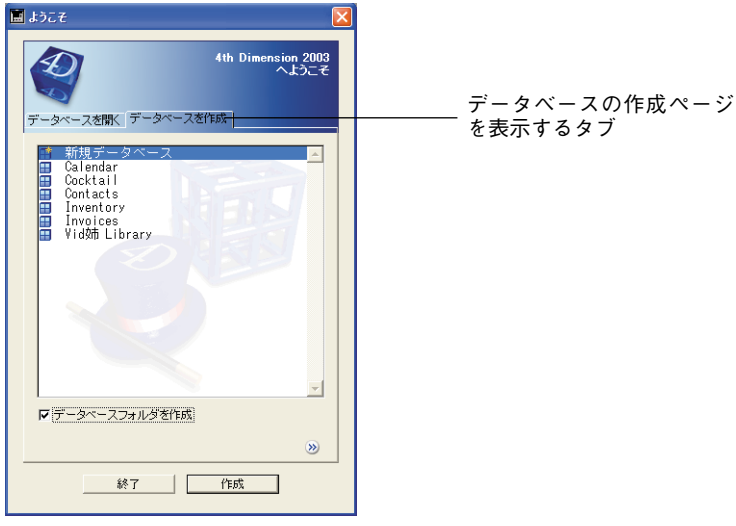
注：また、4Dの使用中にインタプリタモードからコンパイルモード（あるいはその逆）への切り替えを行うことができます。詳細については、後述の「異なるモードへの切り替え」を参照してください。

2つのモードのいずれかが利用できない場合（データベースがコンパイルされていない、または変更後のコンパイルが行われていない、コンパイル済ファイルのみ等）、対応するラジオボタンは使用不可となります。

データベーステンプレートの使用

4th Dimension 2003では、新しくデータベースを作成する際にデータベーステンプレートを利用できるようになりました。これらのテンプレートには、サンプルデータが含まれ、ダイレクトに操作可能です。また、各テンプレートはインタフェース、テーブル、フィールド、フォーム、メソッド等を備えており、独自の開発においてひな形として利用することができます。テンプレートを利用してデータベースを作成できるため、4th Dimensionの能力をより実感しやすくなるでしょう。

データベーステンプレートは、データベースのオープンと作成用ダイアログボックスの「データベースを作成」ページに一覧で表示されます。



「新規データベース」を選択すると、以前のバージョンの4Dと同様に、空のデータベースを作成することができます。

リスト上にある他の名前は、4th Dimensionより提供されるデータベーステンプレートです。新しいテンプレートを追加することもできます（後述）。

テンプレートを選択するには、その名前をクリックします。すると、ダイアログボックスの右側部分に、インタフェースのプレビューと機能に関する簡単な説明が表示されます。

選択したテンプレートを使用して新規のデータベースを作成するには、「作成」ボタンをクリックします。標準の「ファイル保存」ダイアログボックスが表示され、データベースの名前や保存場所を選択することができます。

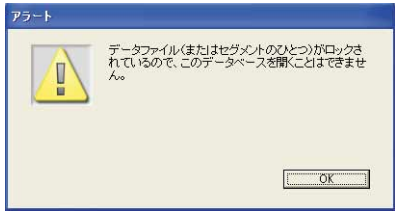
テンプレートの場所

データベーステンプレートは、「4D Templates」という名前のフォルダ内に保存されます。必ずこのフォルダは、4Dアプリケーションの.exeファイル（Windowsの場合）か、ソフトウェアパッケージ（MacOSの場合）と同じ階層に配置してください。

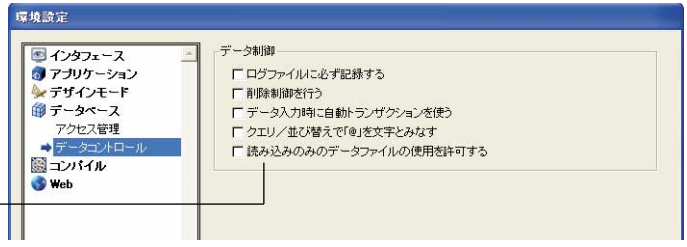
4D社からは定期的にテンプレートがさらに提供されます。これをテンプレート一覧上に表示するには、上記のフォルダに新しいテンプレートを追加してください。

ロックされたデータファイルの検出

バージョン2003より、4th Dimensionにはデータファイルやいずれかのセグメントがロックされているデータベースのオープンを自動的に回避するメカニズムが導入されました。この検出オプションが有効であり、ロックされている場合、4Dは警告メッセージを表示し、そのデータベースのオープンは行いません。



この操作の定義は、「環境設定」ダイアログボックスの「データ制御」ページにある「読み込みのみのデータファイルの使用を許可する」オプションを使用して行います。



ロックされたデータファイルを開くためのオプション

このオプションをチェックしない場合、データファイルがロックされているデータベースを開くことができません（バージョン2003以降の4Dを使用して作成されたデータベースでは、これがデフォルトとなります）。このオプションが適用されるのは、開こうとするデータベースだけであり、4Dアプリケーションで開かれたすべてのデータベースではないという点に注意してください。

注：互換性上の理由から、バージョン2003より前の4Dで作成されたデータベースに対しては、このオプションがデフォルトで選択されています。

さらに、新しい**Is data file locked**コマンドを使用し、開くデータファイルの現在の状態をプログラムで調べることができます（詳細は、後述の**Is data file locked**コマンドの説明を参照してください）。

ロックされたファイルについて

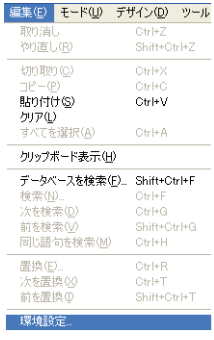
ロックされたファイルは読み込めますが、その内容を変更することはできません。例えば、上書きがサポートされていない媒体上（CD-ROM）に保存されている場合や、このタイプの媒体から再コピーされた場合には、ファイルがロックされています。4th Dimensionはロックされたデータファイルを使用して透過的に作業を行えるため、CD-ROM上に保存されたデータベースを使用することができます。

ただし、この操作では変更が保存されないという点で、ロックされたデータファイルの不用意な使用につながる危険性があります。以前のバージョンの4th Dimensionでは、ワークセッションの最初にデータファイルのロック状態を調べる作業はユーザに委ねられていました。

新しい「環境設定」ダイアログボックス

バージョン2003の4th Dimensionより、「ファイル」メニューの「データベースプロパティ」ダイアログボックスがなくなり、その代わりとして「環境設定」ダイアログボックスを使用します。

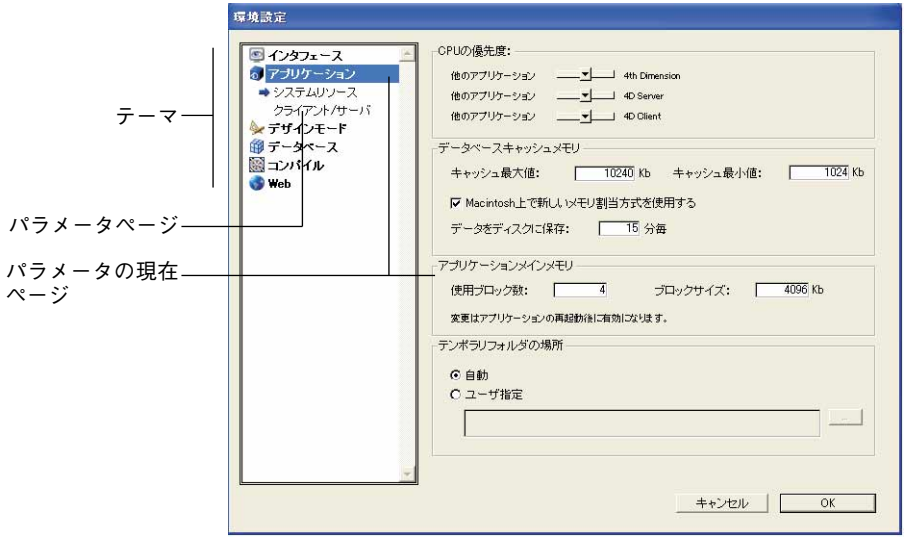
標準的なインタフェースに合わせるため、このダイアログボックスは「編集」メニュー（WindowsまたはMacOS 9）またはアプリケーションメニュー（MacOS X）を使用し、「デザイン」モードからアクセスします。



注：「環境設定...」コマンドは、「デザイン」ウインドウのコンテキストメニューでもアクセスすることができます（任意のオブジェクトの外側をクリック）。

パラメータの再編成

新しい「環境設定」ダイアログボックスは、18ページから成るパラメータ群で構成され、6つのテーマに分かれています（インタフェース、アプリケーション、デザインモード、データベース、コンパイル、Web）。



パラメータページを右側のウインドウに表示するには、対応するテーマを展開し、左側のリストからページ名を選択します。キーボードの「矢印」キーを使い、テーマを展開または縮小したり、選択することができます。パラメータページの値を順次選択するには、「タブ」キーを使用します。

「環境設定」ページのパラメータ構成は、データベースプロパティのものとは異なります。いくつかのプロパティは削除され、さらにプロパティが追加されたものもあります。次の表は、主な「環境設定」と以前の「データベースプロパティ」ダイアログボックスの同等のものとを比較しています。

環境設定（4D 2003）			データベースプロパティ（2003以前）
テーマ	ページ	環境設定	タブ（番号）
インタフェース 	Look	プラットフォーム	ユーザインタフェース（3）
		ドラッグ＆ドロップ点滅	一般（1）
		進捗インジケータ	一般（1）
		ツールバー	一般（1）
	フォーマット	表示フォーマットと入力フィルタ	フォーマット＆フィルタ（6）
	スタイルシート	スタイルシートの編集	ユーザインタフェース（3）
アプリケーション 	システムリソース	アプリケーションの優先度	システム設定（5）
		データベースキャッシュメモリ	システム設定（5）
		メインメモリ	システム設定（5）
		テンポラリフォルダ	システム設定（5）
	クライアント／サーバ	接続タイムアウト	接続（8）
		接続セキュリティ	接続（8）
		4D Open	データ制御とデータアクセス権（2）
デザインモード 	フォント	通常フォント	ユーザインタフェース（3）
	メソッドエディタ	フォント	ユーザインタフェース（3）
		デフォルト表示	4D 2003の新機能
		オプション	4D 2003の新機能
	ストラクチャエディタ	ストラクチャエディタ	デザイン環境（4）
	オプション	起動時モード	一般（1）
		デザインモードを終了してから移動...	4D 2003の新機能
		自動フォーム作成	一般（1）
		互換性	デザイン環境（4）
	ドキュメント	ドキュメントアクセス...	4D 2003の新機能
	コメント	自動コメント	コメント（7）
データベース 	アクセス管理	データアクセス権	データ制御＆アクセス権（2）
		ユーザモードアクセス権	データ制御＆アクセス権（2）
	データ制御	データ制御	データ制御＆アクセス権（2）
コンパイル 	設定	コンパイルオプション	4D 2003の新機能
		コンパイラメソッド...	4D 2003の新機能
Web 	Web公開	ポート、アドレス、開始	Webサーバ I（9）
		WebサーバにSSLを許可する	接続設定（8）
		Webパスワード	Webサーバ II（10）
		開始時のモード	Webサーバ I（9）
		デフォルトHTMLパス	Webサーバ I（9）
	構成	キャッシュ	Webサーバ II（10）
		Webプロセス	接続設定（8）
		テキスト変換	Webサーバ II（10）
		互換性	Webサーバ II（10）
		オプション	サーバ II（10）
	4D WebSTAR	4D WebSTARの接続を許可する...	Webサーバ I（9）
	Webサービス	サーバ	4D 2003の新機能
		クライアント	4D 2003の新機

パラメータの変更

「環境設定」の再編成にともない、以下の点が変更されました。

■「高速スクリーン更新（メモリが余計に必要です）」オプションが削除されました。時代遅れとなってしまったこのオプションは、これまでデータベースプロパティの「一般」ページに含まれていました。

■「メソッド」エディタのオプション「フローチャート式」は削除されました。これは、4th Dimension 2003では、このエディタがサポートされなくなったためです（後述の「フローチャートエディタに関する注意」の節を参照）。

■「デザイン環境」テーマの「オプション」ページにある「カスタムモードに移動する前に、デザインモードを終了する」オプションにより、「カスタム」モードへ移動する際に「デザイン」モードのウインドウが閉じられるようになりました。このオプションを選択しない場合、「カスタム」モードの背後に「デザイン」モードのウインドウが表示されたままになります。

以前のバージョンの4Dでは、「ファイル」メニューの「デザインモード終了」コマンドを使用した場合に、この機能が利用できました（バージョン2003からは削除されています）。

注：「環境設定」でこのオプションが選択されているかどうかに関わらず、Shiftキーを押したまま、「モード」メニューの「ユーザ」コマンドまたは「カスタム」コマンドを選択することにより、「デザイン」モードのウインドウを閉じることができます（後述の「異なるモードを移動する」の節を参照）。

■「データベース」テーマの「データ制御」ページに、新しく「読み込みのみのデータファイルの使用を許可する」オプションが追加されています。このオプションを使用して、ロックされたデータファイルを開く際のアプリケーション操作を設定することができます。このオプションに関しては、前述の「ロックされたデータファイルの検出」の節で詳しく説明しています。

■「デザインモード」テーマの「ドキュメント」ページを使用し、オンラインの4Dドキュメントへのアクセスモードを設定することができます。この新しい機能に関しては、後述の「エクスプローラからHTMLドキュメントをアクセスする」の節で説明しています。

■「コンパイル」テーマ（シングルユーザ用の4th Dimension）には、データベースのコンパイルに関して定義するあらゆる設定がまとめられています。これらの設定は、今まで4D Compilerアプリケーションに含まれていました。

コンパイラの統合に関しては、後述の「コンパイラとアプリケーションビルダ」の章で述べられています。

■「Webサービス」ページには、データベースのWebサービスの公開と接続オプションが含まれています。これらのサービスについては、後述の「Webサービス」の章で説明しています。

ライセンス更新

ライセンスを管理するダイアログボックスは、「ヘルプ」メニューから「ライセンス更新...」コマンドを使用してアクセスできるようになりました。

以前のバージョンの4Dでは、「データベースプロパティ」の「一般」ページにある「ライセンス...」ボタンを使用して、このダイアログボックスにアクセスしていました。

Serial番号のアンインストール

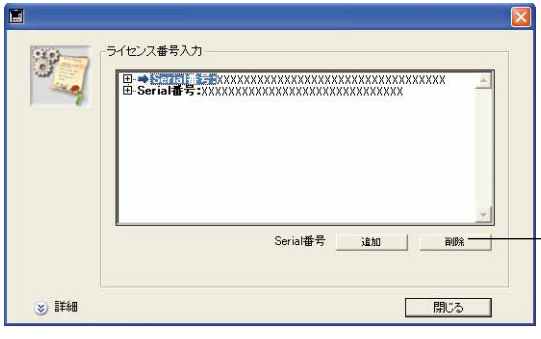
バージョン2003の4Dでは、Serial番号のアンインストールが可能になりました。

▼Serial番号のアンインストールを行うには、次の手順に従ってください。

1 4Dの「ヘルプ」メニューの「ライセンス更新...」コマンドを選択する。

ライセンスを管理するダイアログボックスが表示されます。

2 アンインストールを行いたいSerial番号をクリックし、「削除」をクリックする。



すると、確認ダイアログボックスが表示されます。この選択を確定すると、ライセンスを管理するダイアログボックスからSerial番号が即座に削除されます。

そのライセンス番号に関連付けられたExpansion Serial番号があれば、それも削除されます。

注：直接Expansion Serial番号を削除することはできません。

メニューの管理

バージョン2003の4th Dimensionでは、「カスタム」モードでのアプリケーション開発がより柔軟に使いやすくなるよう、メニュー管理方法が変更されました。

これらの変更は、以下の機能に関する事柄です。

■ デザイン／ユーザ／カスタムモード間の移動

■ デフォルトメニューバー

■「編集」メニューの管理

■ 標準のアクションと「カスタムメニュー」コマンドとの関連付け

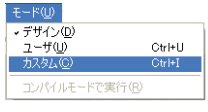
■ メニューバーの名前

異なるモードを移動する

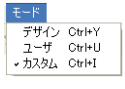
以前のバージョンの4Dでは、一定の条件のもとでのみ「モード」メニューの「カスタム」コマンドが利用可能でした。つまり、「ユーザ」モードから「カスタム」モードへの切り替えだけが可能であり、少なくとも1つのカスタムメニューバーが既に作成されていなければなりませんでした。その上、「カスタム」モードから「デザイン」モードへ直接戻ることはできず、「デザイン」コマンドをアクセスするには「ファイル」メニューから「終了」コマンドを選択する必要がありました。

バージョン2003の4Dからは、デザイン、ユーザ、カスタムモード間の移動が簡単になります。

■「デザイン」モードから直接「カスタム」モードへアクセスすることができます。



■ デフォルトでは、「モード」メニューを使用して「カスタム」モードから「デザイン」モードや「ユーザ」モードへ直接移動することができます。



実際のところ、「モード」メニューはデフォルトメニューバーに含まれています（次節を参照）。

注：「デザイン」モードにおいて、Shiftキーを押しながら「ユーザ」または「カスタム」コマンドを選択すると、「デザイン」モードのウインドウがすべて一斉にクローズされます。この操作は、「環境設定」を使用して設定することもできます（前述の「新しい「環境設定」ダイアログボックス」の節を参照）。

デフォルトメニューバー

バージョン2003の4th Dimensionでは、デフォルトメニューバーに関する新しい機能がいくつか導入されました。

■ 新規データベースでデフォルトメニューバーを作成

4th Dimensionは各新規データベースにデフォルトメニューバー（バー番号1）を自動作成します。したがって、データベースを作成した時点から「カスタム」モードにアクセスすることができます。

■ デフォルトメニューバーの変更

デフォルトメニューバー（新しいバーを作成すると生成される）に、「ファイル」、「編集」、「モード」という3つのメニューが含まれるようになりました。

■ **ファイル**：デフォルトとして、このメニューには「終了」コマンドだけが含まれています。ここで留意すべきなのは、以前のバージョンの4Dでは、デフォルトとしてこのコマンドにメソッドが関連付けられていなかったため、「カスタム」モードを終了すると「ユーザ」モードへ戻っていた点です。バージョン2003の4Dからは、「Quit」自動アクションがこのコマンドに関連付けられ、アプリケーションは実際に終了します。

■ **編集**（新しいメカニズム）：バージョン2003の4th Dimensionにおいて、「編集」メニューが標準で用意され、すべて変更可能になります。詳細については、後述の「編集メニューの管理」を参照してください。

以前に作成したデータベースに関しては、このメニューの今までの動作は変わらずに保持されます。

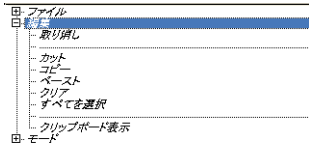
■ **モード**：デフォルトとして、「デザイン」、「ユーザ」、「カスタム」コマンドが「モード」メニューに含まれます。このメニューを使用し、「カスタム」モードから直接、ほかの4D環境へアクセスすることができます。

注：デフォルトメニューバーのコマンドは、自動メニューアクションを使用します（4D 2003の新機能、後述の「カスタムメニューコマンドに標準アクションを関連付ける」の節を参照）。

編集メニューの管理

「編集」メニューの管理やカスタマイズを行えるようになりました。以前のバージョンの4Dでは、このメニューの管理は4Dアプリケーションに委ねられ、変更することができませんでした。

新しくメニューバーを作成すると、「メニューバー」エディタ上には「編集」メニューが用意されます。このメニューのコマンドは標準的なものです。



デフォルトでは、このメニューの各コマンドは標準メニューアクションに関連付けられています。標準アクションにより、コンテキストに応じて4Dはコマンドのアクティブ化／非アクティブ化を行います。

もちろん、このメニューにコマンドを追加して独自のメソッドを使用することもできます。

■ 今までの「編集」メニューのメカニズム

以前のバージョンの4Dで作成され、バージョン2003で開かれたデータベースでは、追加オプションである「旧バージョンの編集メニューを使用」を利用することができます。既存のメニューバーに対して、このオプションがデフォルトとして選択されます。これにより、「編集」メニューの以前の操作を維持することができます（アプリケーションによって管理される）。このオプションを選択すると、以前のバージョンの4Dと同様に、4th Dimensionは修正不可の「編集」メニューをメニューバーに追加します。このオプションを選択しない場合には、カスタマイズしたメニューバーの「編集」メニューを自分で追加、管理しなくてはなりません（新しいメカニズム）。

既存のデータベースとの互換性

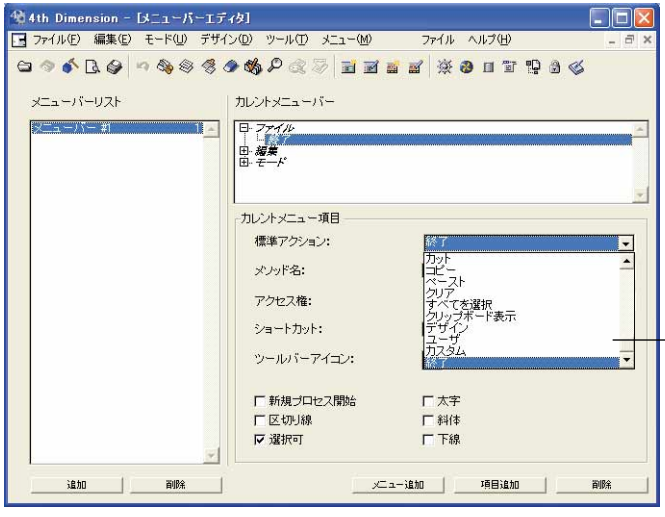
前述の各節で説明した原則は、バージョン2003以降の4th Dimensionで作成された新しいメニューバーにデフォルトとして適用されます。

以前のバージョンの4th Dimensionで作成されたアプリケーション内の既存のメニューバーに関しては、その操作は従来通りです。

カスタムメニューコマンドに標準アクションを関連付ける

バージョン2003の4th Dimensionでは、カスタムメニューコマンドに標準アクションを関連付けることができます。今までは、メニューコマンドに関連付けることができるのは、プロジェクトメソッドだけでした。

カスタムメニューコマンドに標準アクションを関連付けるには、「メニューバー」エディタの「標準アクション」のポップアップメニューからアクションを選択します。



注：標準アクションに関連付けられているデフォルトのキーボードショートカットは変更しないようお勧めします。

メニューに提供される標準アクションのリストは、ボタンのものと同様です（4Dのフォームウィザード上でボタンのプロパティのパレットを用いてアクセス可能）。実際のところ、大半のアクションは両方の状況で使用できます。

しかし「サブレコード編集」、「サブレコード削除」、「サブレコード追加」、「自動スプリッタ」アクションはメニューコマンドに関連付けることができません。したがって、これらのアクションはポップアップメニューには表示されません。これとは逆に、ボタンに対してはすべての標準アクションを使用することができます。

標準アクションのカスタムメニューコマンドへの関連付けは、任意の作業です。標準アクションまたはプロジェクトメソッドのいずれをコマンドに関連付けるかを選択することができます。

さらに、標準アクションとプロジェクトメソッドの両方をメニューコマンドに関連付けることもできます。この場合、標準アクションは実行されませんが、4th Dimensionはこのアクションを使用し、状況に応じてメニューコマンドを有効、または無効にします。メニューコマンドが無効である場合、関連付けられたプロジェクトメソッドも実行されません。

新しい標準アクション

バージョン2003の4th Dimensionでは、新しい標準アクションが提供されています。

- **取り消し**：実行された前回の動作をキャンセルします（「編集」メニューの「取り消し」コマンドと同じ）。取り消しとキャンセルとを混同しないようにしてください（キャンセルは、レコードの表示中に行われた変更をすべてキャンセルし、出力フォームに戻ります）。
- **カット**：選択範囲を削除して、それをクリップボードに配置します。
- **コピー**：選択範囲をクリップボードにコピーします。
- **ペースト**：クリップボードの内容をカーソルのある場所に挿入します。
- **クリア**：選択範囲を削除します。何も選択されていない場合、カーソルが置かれたエリア全体を消去します（入力エリアのみ）。
- **すべてを選択**：コンテキスト内の選択可能な要素をすべて選択します。
- **クリップボード表示**：新しいウインドウを開き、クリップボードの現在の内容を表示します。
- **デザイン**：4th Dimensionの「デザイン」モードのウインドウとメニューバーを前面に配置します。
- **ユーザ**：4th Dimensionの「ユーザ」モードのウインドウとメニューバーを前面に配置します。
- **カスタム**：4th Dimensionの「カスタム」モードのウインドウとメニューバーを前面に配置します。
- **終了**：「よろしいですか？」という確認のダイアログボックスを表示し、これが確定されると4Dアプリケーションを終了します。確定されない場合には、この操作はキャンセルされます。

  オブジェクトメソッドも関連付けられているボタンに、このアクションが割り当てられると、その実行手順は次のようになります。まず、確認ダイアログボックスが表示され、これが確定されると4Dはオブジェクトメソッドを実行します。実行終了後、アプリケーションが終了します。

MacOS Xに関する注意：MacOS Xにおいては、カスタムメニューコマンドが「終了」アクションに関連付けられている場合、データベースを「カスタム」モードで実行すると、コマンドは自動的にアプリケーションシステムメニュー上に配置されます。このメカニズムによって、MacOS X上の「終了」コマンドの管理が簡単になります。

メニューバーの命名

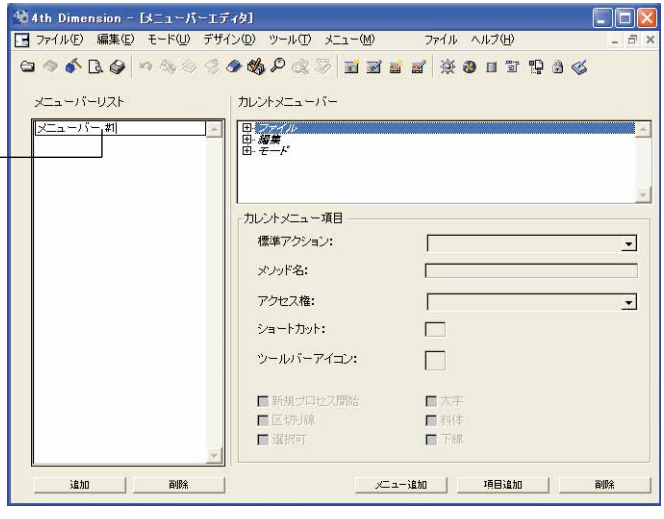
「カスタム」モードの各メニューバーに対して、独自の名前を割り当てられるようになりました。今までは、リスト上のメニューバーの位置に相当する名前がデフォルトとして付けられ（バー#1、バー#2等）、変更することができませんでした。

この操作により、「デザイン」モードのさまざまなダイアログボックスにおいて、メニューバーの識別が容易になります。

注：この新しい機能を反映するために、MENU BARコマンドのシンタックスが変更されました。詳細については、後述するMENU BARコマンド（「メニュー」テーマ）の説明を参照してください。

もちろん、メニューバーを番号で参照することも可能です。

メニューバーの名前を変更するには、Ctrlキー（Windows）またはCommandキー（MacOS）を押したまま、「メニューバー」エディタのバーのタイトルをクリックします。すると、名前が入力可の状態になります。

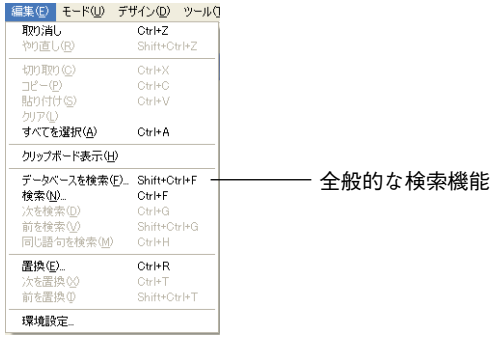


メニューバー名には31桁までの文字を指定でき、ユニークな名前でなくてはなりません。

注：コンポーネントのステータスに関わらず（パブリック、プロテクト、プライベート）、コンポーネントによりインストールされたメニューバーの名前を変更することはできません。

検索コマンドの場所

4th Dimensionの全般的な検索機能が、4th Dimension 2003の「編集」メニューへ移されました（以前のバージョンの4Dでは「ツール」メニューの「検索...」コマンドを用いて利用可能でした）。さらに、「データベースを検索...」という名称に変わりました。



この機能の用途は、今までのバージョンの4Dと同じです。

注：他の「検索」と「置換」コマンドもこのメニュー上にあり、4Dの「メソッド」エディタで使用します（後述の「検索と置換」の節を参照）。

MacOS X上でQuartzを使用する

4th Dimension 2003では、4DアプリケーションとMacOS Xインタフェースとの統合がさらに進められています。4Dは、Quartzグラフィックレンダリングエンジンを使用して、データベースのすべてのテキストを表示します（プラグインのテキストも含む）。

バージョン6.8.xの4Dにおいて、テキストタイプのフィールドや変数、およびプラグインは、Quartzレンダリングの対象とならず、“従来の”MacOSのレンダリングエンジンであるQuickDrawを用いて表示されていました。

一方で、Quartzレンダリングによってテキストにアンチエイリアスが適用されるため、テキストがより美しく表示されるようになり、他方では、文字間隔がより正確に計算されます。

Zoo “従来の”表示（QuickDraw）

Zoo Quartz 表示

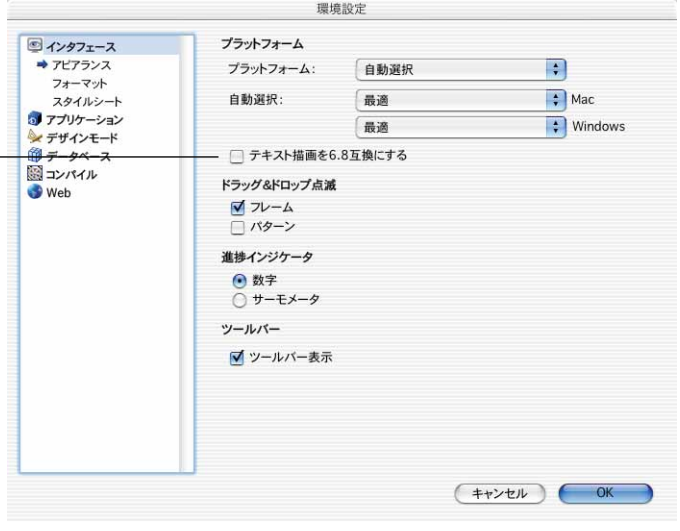
注：

・“シャドウ”、“アウトライン”属性等、特定のテキストスタイルのなかにはQuartzによってサポートされないものがあります。したがって、これらのスタイルが割り当てられたオブジェクトの表示は、自動的にQuickDrawにより処理されます。

・Quartzエンジンは、表示目的にのみ使用されます。印刷は、標準のモードで実行されます。

文字間隔が修正されると、バージョン6.8.xで開発されたアプリケーションのインタフェースの外観が変わってしまうため、そのアプリケーションの「環境設定」（「インタフェース」テーマの“アピアランス”ページ）にある「テキスト描画の6.8互換」オプションを使用して、“フル”Quartz表示モードを無効にすることができます。

MacOS X上のレンダリングオプション



以前のバージョンの4th Dimensionで作成されたアプリケーションには、このオプションがデフォルトとして選択されます。逆に、バージョン2003以降の4th Dimensionで作成された新しいデータベースでは、このオプションはデフォルトとして未選択になります。

注：Quartzレンダリングを表示するためには、データベースをMacOS Xプラットフォーム上で実行し、「Macテーマ」プラットフォームインタフェースを使用しなければなりません。

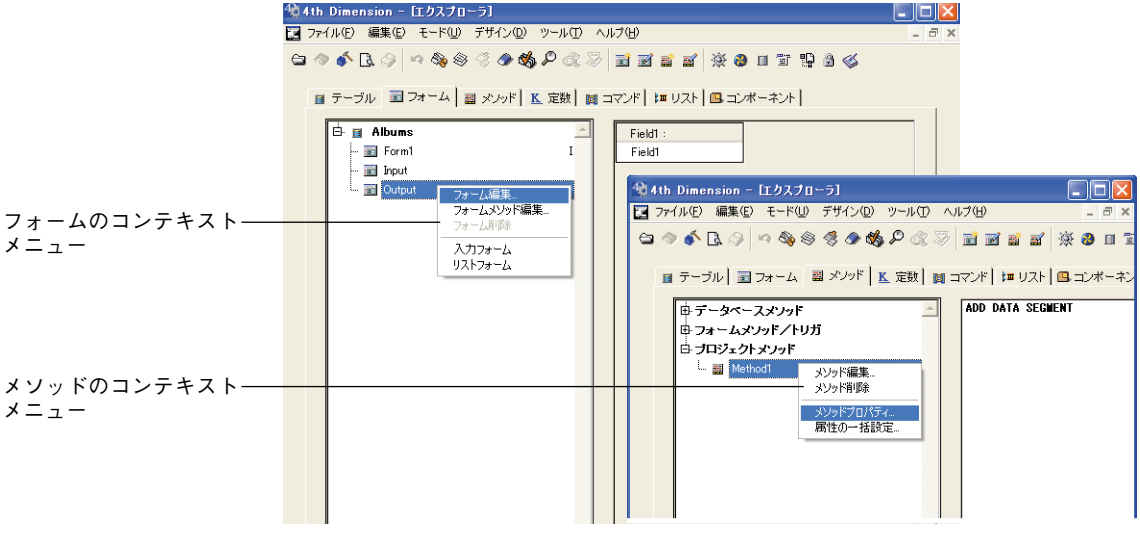
エクスプローラのコンテキストメニュー

4th Dimension 2003のエクスプローラの「フォーム」および「メソッド」ページではコンテキストメニューが提供され、各種管理機能に直接アクセスすることができます。

エクスプローラのコンテキストメニューを表示するには、次のようにします。

■ Windows上では、階層リストの項目上で右クリック。

■ MacOS上では、階層リストの項目上でcontrol + クリック。



これらのコンテキストメニューのコマンドについて、次に説明します。

「フォーム」ページ

フォーム名をクリックすると、メニューが表示されます。

■ **フォーム編集...**：「フォーム」エディタのウインドウ上にそのフォームを表示します。

■ **フォームメソッド編集**：「メソッド」エディタのウインドウ上にフォームメソッドを表示します。

■ **フォーム削除**：フォームを削除します（フォームがカレント入力フォームや出力フォームではない場合）。

■ **入力フォーム／リストフォーム**：そのフォームをテーブルのカレント入力フォームや出力フォーム（または両方）に指定します。リストの下側にあるチェックボックスを使用して、この機能を使用することもできます。

「メソッド」ページ

プロジェクトメソッド名をクリックするか、またはプロジェクトメソッドが選択されている場合にプレビューエリアをクリックすると、メニューが表示されます。

- **メソッド編集...**：「メソッド」エディタのウインドウ上にプロジェクトメソッドを表示します。
- **メソッド削除**：プロジェクトメソッドを削除します。
- **メソッドプロパティ...**：「メソッドプロパティ」ダイアログボックスを表示します。
- **属性の一括設定**：「メソッド属性」ウインドウを表示します（後述の「メソッド属性の一括設定」の節を参照）。

エクスプローラから HTML ドキュメントにアクセスする

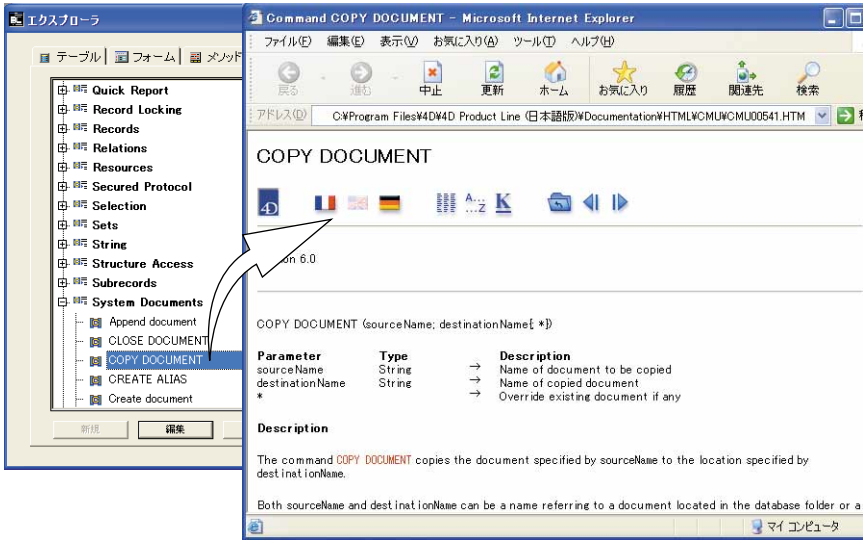
4Dエクスプローラから、4th Dimensionのオンライン（HTML）ドキュメントに直接アクセスできるようになりました。4Dコマンド名をダブルクリックすると、対応するHTMLページがお使いのブラウザ上に表示されます。

さらに（MacOSのみ）、エクスプローラウインドウ上にページを直接表示することもできます。

HTMLドキュメントページは、CD-ROMやお使いのハードディスク上に保存されたものでも、あるいは4D S.A. Webサイトから直接読み込まれたものでも構いません。この操作は、アプリケーションの「環境設定」で設定します。

ダブルクリックによるページ表示

4Dエクスプローラ上で、コマンド名をダブルクリックするか、またはコマンドの選択後「編集」ボタンをクリックすると、お使いのブラウザ上にそのコマンドの説明が表示されます。

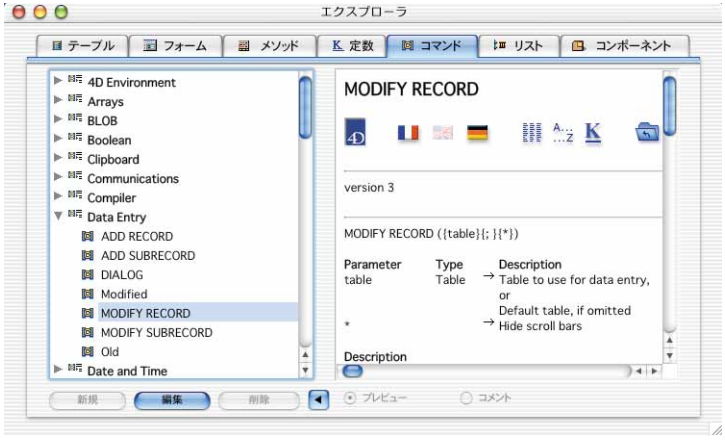


ブラウザが既に開かれている場合、今まで表示されていたページの代わりにこのページが表示されます。ブラウザが開かれていない場合、4th Dimension はページを表示する前に、デフォルトのブラウザを起動します。

ページのロード先となる場所の設定は、アプリケーションの「環境設定」の「エクスプローラからのドキュメントアクセス」パラメータを使用して行います。

エクスプローラでページを表示する（MacOS のみ）

MacOS 上では、エクスプローラで 4D コマンドを選択すると、そのコマンドの HTML 形式での説明がプレビューエリアに表示されます。



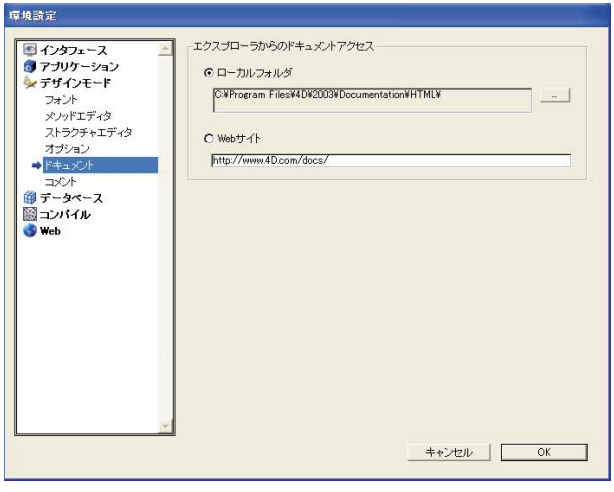
表示される各ページ上の移動用ボタンや、ハイパーテキストリンクは有効です。一方で、標準的なWebナビゲーション機能（戻る／進む）は使用できず、また情報をコピーすることもできません。

ページのロード先となる場所は、アプリケーションの「環境設定」で指定したパラメータに応じて変わります（後述の「アクセス設定」の節を参照）。

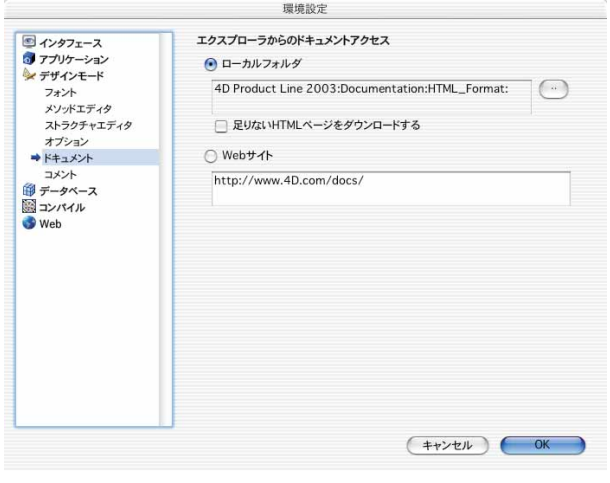
アクセス設定

4Dの「環境設定」の「ドキュメント」ページで、4Dエクスプローラからのオンラインドキュメントへのアクセスを設定することができます。

Windows



MacOS



■ ローカルフォルダ／Webサイト：このオプションを使用して、HTMLドキュメントをロードしてくる場所を指定します。

■ ユーザがエクスプローラ上でコマンドをダブルクリックした場合（WindowsおよびMacOS）

■ エクスプローラ上でコマンドが選択された場合（MacOSのみ）

「ローカルフォルダ」オプションを選択した場合、4th Dimensionは指定したフォルダ内でHTMLページを探します。デフォルトでは、この場所は4Dアプリケーションと同じ階層にある「Documentation」フォルダになります（MacOS上では4Dソフトウェアパッケージと同じ階層）。

指定されたアクセスパスは、4Dアプリケーションと関連しています。この場所は自由に変更することができます。つまり、HTMLドキュメントの場所は、別のボリューム上やCD-ROM等であっても構いません。他の場所を指定するには、入力エリアの横にあるボタンをクリックし、ドキュメントのルートフォルダを選択します（「4DDOCFR.HTM」、「4DDOCUS.HTM」、「4DDOCGM.HTM」ファイルを含むフォルダ）。

「Webサイト」オプションを選択した場合には、4th Dimensionは指定したURL上でHTMLページを探します。デフォルトのURLは、4D, Inc.のインターネットサイト上にある標準の4D 2003ドキュメント用URLです。このURLは自由に変更することができます。

■「足りないHTMLページをダウンロードする」（MacOSのみ）：このオプションは、“ローカルフォルダ”内にページが見つからない場合、インターネットからHTMLドキュメントページを自動的にダウンロードする処理を有効（または無効）にするために使用します。この処理によって、お使いのドキュメントに新しい4Dコマンドを取り込んだり、必要に応じて徐々にローカルフォルダの内容を増やしてゆくことができます。

デフォルトでは、このオプションが選択されていません。「ローカルフォルダ」からドキュメントをロードするよう選択した場合、4Dコマンドに対応するページがこのフォルダ内に見当たらなければ、4th Dimensionはエラーページを表示します。一方で、「足りないHTMLページをダウンロードする」オプションを選択している場合、4th Dimensionは「Webサイト」エリアに指定されたURLを元に、そのコマンドに関するページをダウンロードします。ダウンロードが行われると、そのページはローカルフォルダに保存されます。

4D Server：オンラインドキュメントへのアクセス設定は、各クライアントマシン毎に行います。

新しい「メソッド」エディタ

バージョン2003の4th Dimensionの「メソッド」エディタには、数多くの新機能や改良点が盛り込まれています。これらの新機能は、インタフェース、機能、データの入力と編集、検索と置換、読み込みや書きだし等、エディタのあらゆる面に関連しています。

新しい4D「メソッド」エディタは、特に以下の操作のために使用することができます。

■ プログラム構造の展開や縮小（If／Else／End if等）、編集ウインドウの分割、スタイルのカスタマイズ

■ 選択行のコメント付けやコメントの削除、複数レベルでの変更の取り消し、複数のクリップボードの使用、項目のドラッグやドロップ

■ マクロを作成、使用して、繰り返し使用するコードの入力を迅速化

■ 表示される4Dオブジェクトのリスト（コマンド、フォーム、リスト、メソッド等）の内容とその数をカスタマイズ

■ メソッドの読み込みと書き出し

■ カレントメソッドやすべてのデータベースメソッド内の式の検索と置換

■ 2GBまでのテキストを含むメソッドの入力

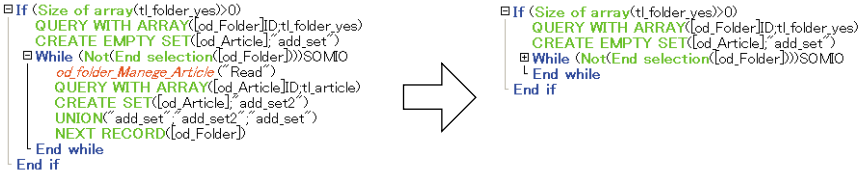
これらの新機能はすべて、次の節で説明します。

インタフェース

「メソッド」エディタの全体的なエルゴノミクスを改善する目的で、数々の変更が加えられています。

展開／縮小

メソッドが見やすくなるように、ループ式や条件式内にある4Dコードを縮めたり、展開することができるようになりました。



この新機能の操作は階層リストと同様です。つまり、⊟アイコンをクリックするとループ式や条件式内にあるコードが縮小されます。また、⊞アイコンは縮められているコード部分を示し、このアイコンをクリックすると縮められたコードが展開されます。

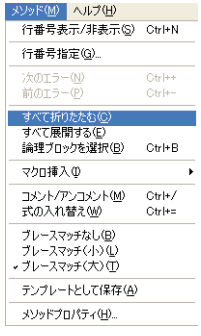
コードが展開されると、画面上に表示されるキーワードの開始と終了の間に垂直線が引かれ、コードの構造がより見やすくなる点に注目してください。

コードの一部が縮小されていると、その部分を変更することができません。条件式やループ式に関連する文字の入力や削除を行うと、縮小部分は自動的に展開されます。

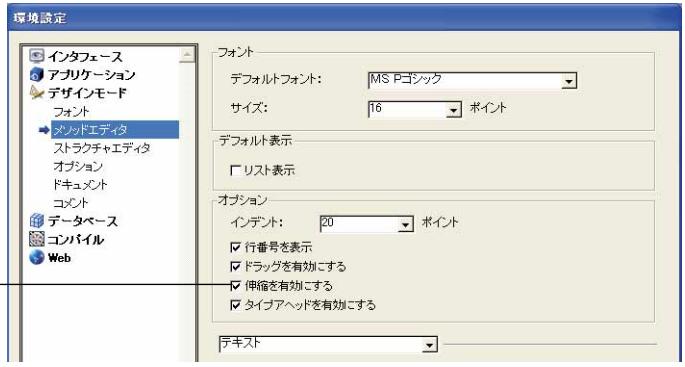
縮められたコード部分の選択やコピー、ペースト、削除を行うことができます。間に含まれるすべての行のコピー、ペースト、削除がそれぞれ行われます。

注：コードの一部をペーストすると、その部分は自動的に展開されます。

「メソッド」メニューまたは「メソッド」エディタのコンテキストメニューの「すべて縮める」コマンドや「すべて拡げる」コマンドを使用すると、メソッドのループ式や条件式をすべて縮小／展開することができます。



ループ式や条件式を階層リストの形式で表示するこの機能は、データベースの「環境設定」ダイアログボックスの「メソッドエディタ」ページ（「デザインモード」テーマ）にある「伸縮を有効にする」オプションを使用して、有効または無効にすることができます。



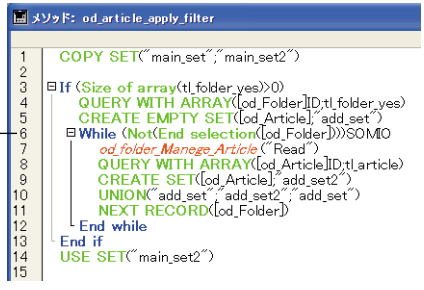
コードの階層形式表示オプションのアクティブ化

デフォルトでは、このオプションが選択されています。

注：メソッドの印刷時には、この表示モードが考慮されません。つまり、今までのバージョンの4Dと同様に、ソースコードは常に展開され、ループ式や条件式には垂直線が引かれます。

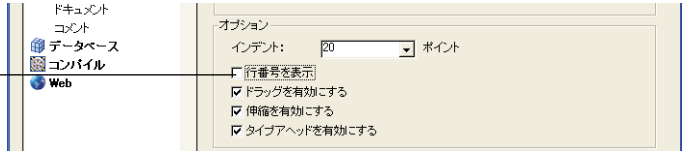
行番号を表示

「メソッド」エディタの各ウインドウに行番号を表示できるようになりました。



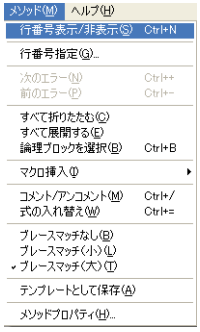
行番号

行番号を表示するこの機能は、データベースの「環境設定」ダイアログボックスの「メソッドエディタ」ページ（「デザインモード」テーマ）にある「行番号を表示」オプションを使用して、デフォルトとして有効または無効にすることができます。



行番号を表示するオプション

「メソッド」メニューの「行番号表示／非表示」コマンドを使用すると、「メソッド」エディタの各ウインドウに対して個別に、この表示を変更することもできます。

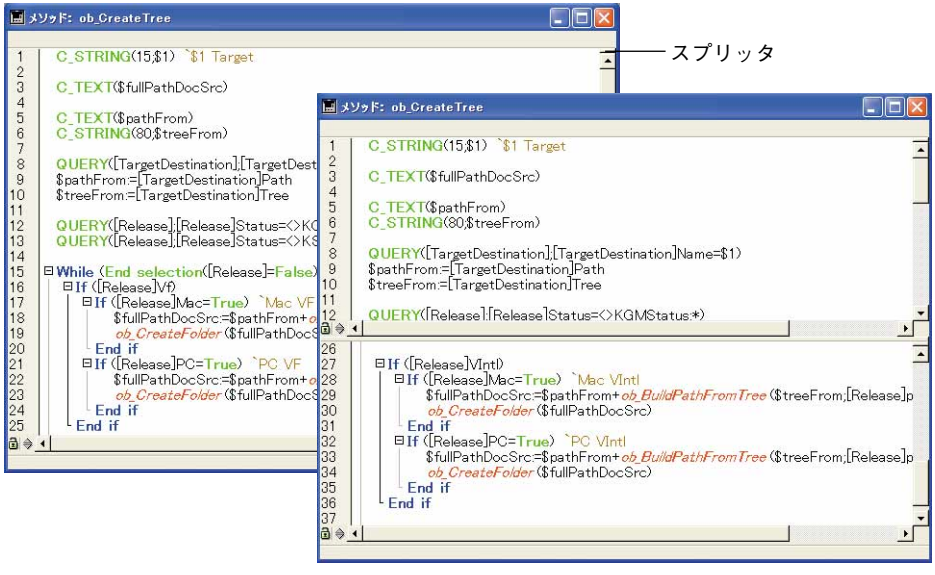


エディタウインドウの分割

4D 2003の「メソッド」エディタでは、編集エリアを複数の水平区画に分割することができます。編集エリアを分割すると、他の部分から切り離して同一メソッドの各部分を表示したり、スクロールすることができます。例えば、一般的にメソッドの説明や、コメント、変数の宣言が含まれるメソッドのヘッダ部分を画面上に表示しておきたい場合には、この機能が役立ちます。また、同一メソッドの離れた位置にあるエリアを複数同時に表示することもできます。

■ 区画の作成

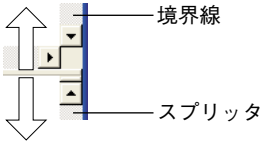
表示区画を作成するには、ウインドウの一番上にあるスプリッタをクリックし、それを下側へスライドさせます。



複数の区画を作成するには、この操作を必要な分だけ繰り返します。スプリッタをクリック（一回）してそれを下側にドラッグすると、既存の区画の下側に区画を作成することができます。

■ 区画サイズの変更

区画サイズを変更するには、境界線のひとつを上下方向にスライドさせます。区画の境界線とスプリッタを混同しないよう注意してください。



区画の高さが十分ではない場合、スプリッタは表示されません。

■ 区画の削除

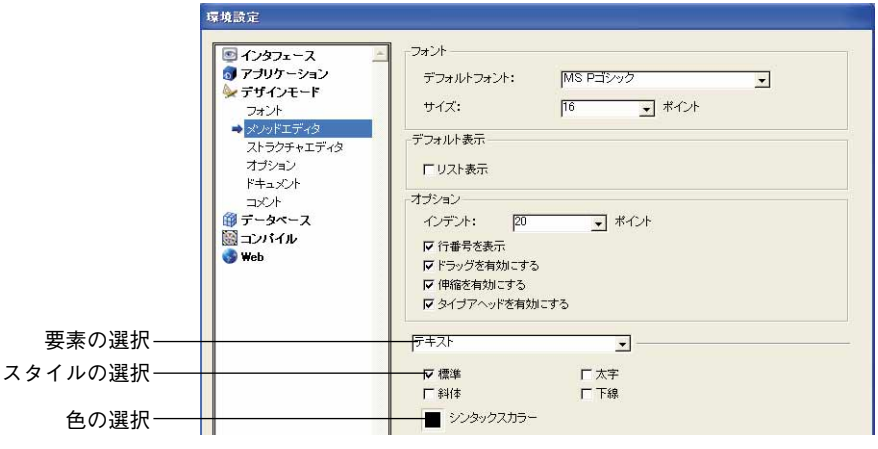
表示している区画を削除するには、下側の境界線をウインドウの上部までスライドするか、またはダブルクリックします。

シンタックス要素のスタイルと色をカスタマイズする

4Dでは、4Dランゲージの要素タイプごとに（フィールド、テーブル、変数、パラメータ等）特定の色を割り当てることができます。

さらに、新しい「メソッド」エディタを使い、それぞれの要素に対して特定のスタイルを割り当てることもできます。さまざまな色とスタイルの組み合わせをメソッド要素に割り当てると、コード管理の際に非常に役立ちます。

スタイルや色のカスタマイズは、アプリケーションの「環境設定」ダイアログボックスの「メソッドエディタ」ページ（「デザインモード」テーマ）で定義します。



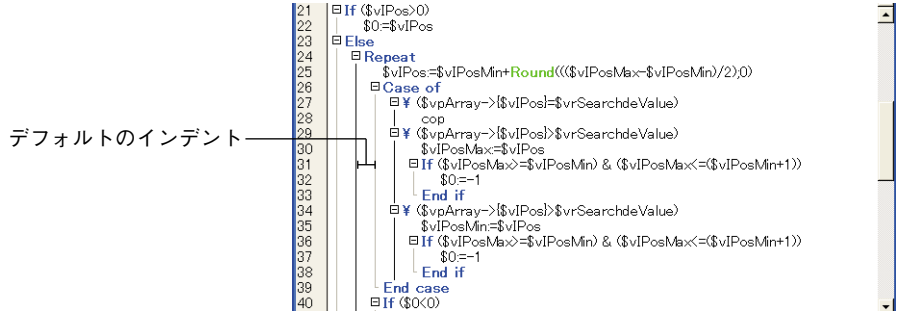
注：新しく加わった「テキスト」要素タイプは、定義された他のタイプに属さないテキストすべて（記号、句読点、リテラル定数等）を示します。

次のスタイルを使用することができます。

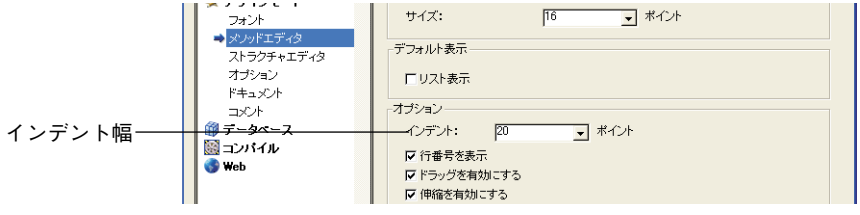
- 標準
- 太字（デフォルトとして4Dコマンド、キーワード、プラグインコマンドに使用されます）
- 斜体（デフォルトとしてプラグインコマンドとメソッドに使用されます）
- 下線（デフォルトとして定義済定数に使用されます）
- アウトラインとシャドウ（MacOS上でのみ表示されるスタイル）

インデント幅

コード構造を分かりやすくするために、4Dコードは自動的にインデントされます。



アプリケーションの「環境設定」ダイアログボックスの「メソッドエディタ」ページ（「デザインモード」テーマ）にある新しいオプションを使用して、インデント幅を変更できるようになりました。



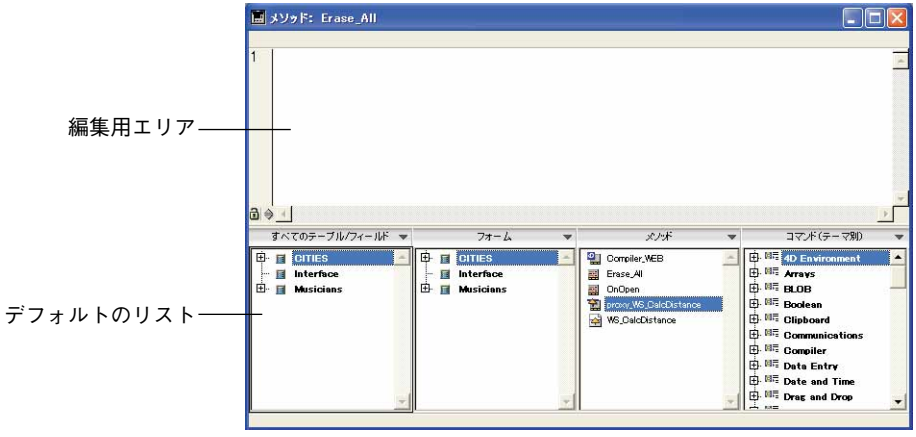
この幅は必ずポイント単位で定義します（デフォルトでは20ポイント）。

メソッドが何レベルもの入れ子構造を持つ複雑なアルゴリズムを含む場合は、このデフォルト値を変更すると便利です。横スクロールを避けるために、より小さいインデント幅を使用することができます。

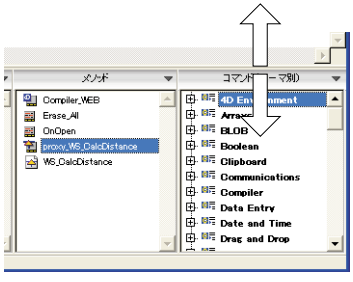
リストの表示とカスタマイズ

4th Dimension 2003の「メソッド」エディタでは、メソッド作成に必要となる項目リストの表示をカスタマイズすることができます（コマンド、定数、フォーム等）。ウインドウに表示するリストの数や内容を選択することが可能です。

デフォルトとして、「メソッド」エディタには編集用のエリアと4種類のリストが表示されます。

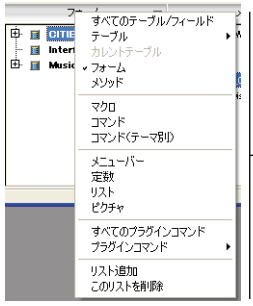


各リストエリアの幅は、その区切り線をドラッグすることにより拡張や縮小を行えます。また、編集エリアとリストの間にある区切り線をドラッグし、編集エリアのサイズと相対的にリストエリアのサイズを調整することもできます。



リスト上の項目をダブルクリックすると、編集エリア上のカーソル位置にそのコマンドが挿入されます。

■ リストの内容を変更するには、変更したいリストのタイトルエリアをクリックします。すると、ポップアップメニューが表示され、表示する項目タイプを選択することができます。



表示できる項目は、以下の通りです。

■ **すべてのテーブルとフィールド**：データベースのテーブル名とフィールド名を階層リスト形式で表示。

■ **テーブル（サブメニュー）**：サブメニューを用いて選択されたテーブルのフィールド名を表示。

■ **カレントテーブル**：カレントテーブルのフィールド名を表示（トリガ、フォームメソッド、オブジェクトメソッドにおいて使用可能）。

■ **フォーム**：データベースのテーブル名とフォーム名を階層リスト形式で表示。

■ **メソッド**：データベースのプロジェクトメソッド名を表示。

■ **マクロ**：データベースで定義されたマクロ名を、マクロファイルの順序で分類して表示。

注：今までのバージョンの4Dで提供されていたキーワードリストは、マクロを用いて処理されるようになりました（詳細は、後述する「マクロの作成と使用」の節を参照）。

■ **コマンド**：4th Dimensionのランゲージコマンドをアルファベット順に表示。

■ **コマンド（テーマ別）**：4th Dimensionのランゲージコマンドをテーマ別に階層リスト形式で表示。

■ **メニューバー**：メニューバー名と番号を表示。

■ **定数**：4th Dimensionの定数と任意のプラグインをテーマ別に階層リスト形式で表示。

■ **リスト**：リスト名を表示。

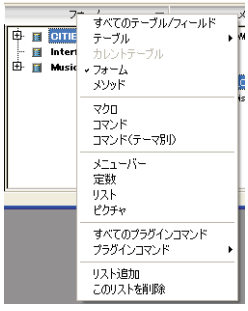
■ **ピクチャ**：4Dピクチャライブラリに保存されているピクチャの名前を表示。

■ **すべてのプラグインコマンド**：データベースにインストールされているすべてのプラグインのコマンドをテーマ別に分類して階層リスト形式で表示。

■ **プラグインコマンド（サブメニュー）**：サブメニューを用いて選択された特定のプラグインのコマンドを表示。

注：「マクロ」項目を除き、リストはすべてアルファベット順になります。

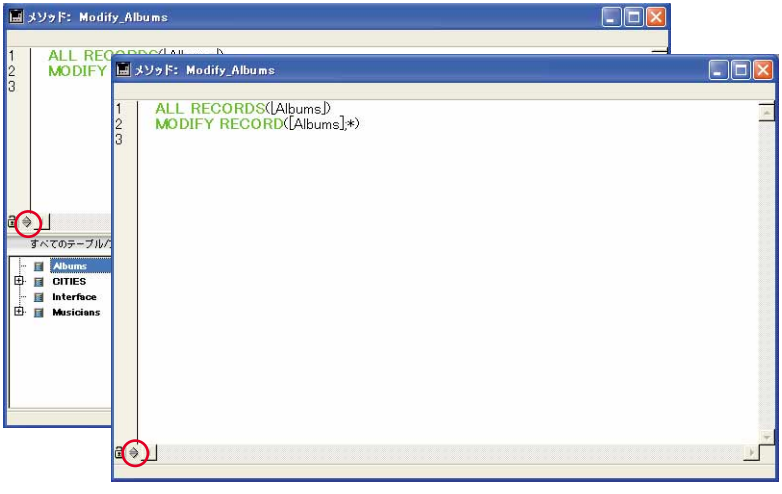
■ リストの追加や削除を行うには、いずれかのリストのタイトルエリアをクリックし、ポップアップメニューから対応するコマンドを選択します。



リストの追加や削除

エディタウインドウ上には、少なくとも1つのリストが表示されていなくてはならない点に注意してください。最後に残されたリストをクリックすると、「このリストを削除」コマンドは使用不可になります。リストをすべて隠したい場合には、必ずデータベースの「環境設定」ダイアログボックスを使用してください（後述）。

■「メソッド」エディタウインドウの編集エリアだけを表示することができます。これを行うには、編集エリアの左隅にある アイコンをクリックします。すると、リストが隠されます。リストを再度表示するには、このアイコンを再びクリックします。



■ デフォルトとして、このリストを隠しておくこともできます。これを行うには、データベースの「環境設定」ダイアログボックス（「デザインモード」テーマ）の「メソッドエディタ」ページにある「リスト表示」オプションの選択を解除します。

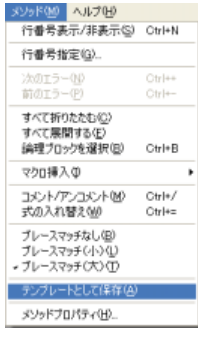
「環境設定」ダイアログボックスで行った変更を反映させるには、まず開かれているメソッドをすべて閉じてから、再び開き直してください。

■ **テンプレートとして保存：**「メソッド」エディタウインドウで指定した各パラメータを“テンプレート”の形で保存することができます。一度テンプレートを保存すると、新しく「メソッド」エディタウインドウを開くたびに、テンプレートに定義した設定が使用されます。

次のパラメータはテンプレートに保存することができます。

- 編集エリアとリストエリアの相対サイズ
- リスト数
- 各リストの位置と内容
- 各リストの相対幅

「メソッド」エディタウインドウをテンプレートとして保存するには、「メソッド」メニュー、または「メソッド」エディタのコンテキストメニューから「テンプレートとして保存」コマンドを選択します。



「メソッド」エディタのコンテキストメニュー

すると、テンプレートが即座に保存されます（ダイアログボックスは表示されません）。テンプレートは、4Dアプリケーションの初期設定に保存されます。テンプレートが既に存在する場合には置き換えられます。

データ入力アシスタント

4D 2003の「メソッド」エディタの数々の新機能は、コードの入力や編集を円滑にする目的のために導入されました。

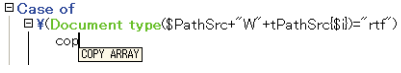
さらに、メソッド内でマクロを作成し使用することにより、繰り返し使用するコードの入力が非常に迅速に行えるようになります。この新しい機能に関しては、後述する「マクロの作成と使用」の節で解説しています。

先行入力機能（タイプアヘッド）

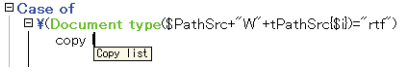
「メソッド」エディタに“先行入力”機能が導入されました。

このタイプの機能は、今までのバージョンの4Dで“ワイルドカード（@）”記号を用いて、以前より利用することができました（ただし、より限定的な形式で）。今バージョンの4th Dimensionから、入力された最初の数桁の文字を元にして、候補が表示されるようになりました。

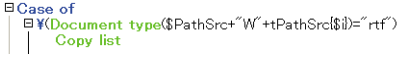
以下の例題では、“cop”という文字列を入力すると、この文字列で始まる4Dコマンドのうち最初のもの（アルファベット順による）を納めたヒントが表示されます。



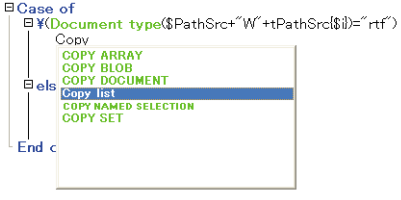
さらに文字を入力すると、ヒントに提示される値が更新されます。



入力した文字が、考えられる唯一の値に達した時に「タブ」キーを押すと、その値が挿入されます。

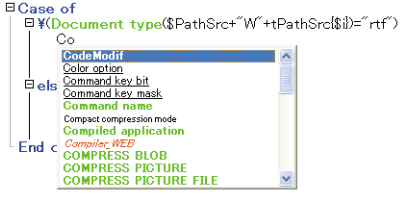


これ以外の場合に「タブ」キーを押すと、入力した文字で始まるすべてのワードのリストが表示されます。



このリストはアルファベット順に表示されます。値をダブルクリックするか、または上下矢印キーを使用して各値をスクロールした後で改行キーを押すことにより、リストの値を選択することができます。「Esc」キーを押して、さらに入力を続けることもできます。

入力した値が、さまざまなタイプのオブジェクトに対応する場合、各オブジェクトはそのカレントスタイルでリスト上に表示されます。



次のタイプのオブジェクトが表示されます。

■ 4Dコマンド

■ ユーザメソッド

■ テーブル名

■ フィールド名

■ 定数

■ プラグインコマンド

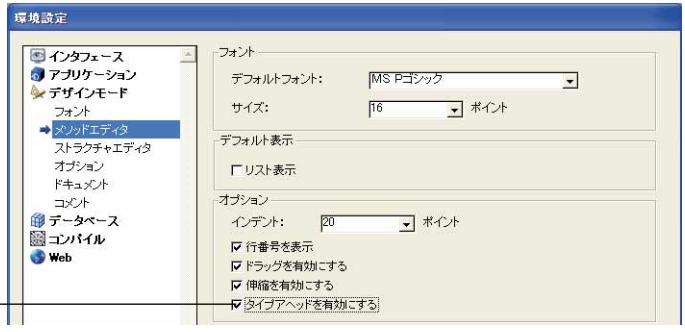
■ 4Dキーワード

■ マクロ

注：マクロ名は“< >”で囲まれて表示されます。マクロに関する詳細は、後述する「マクロの作成と使用」の節を参照してください。

■ 先行入力機能の取り消し

アプリケーションの「環境設定」ダイアログボックスの「メソッドエディタ」ページ（「デザインモード」テーマ）にある「タイプアヘッドを有効にする」オプションを使用して、先行入力機能を有効／無効に設定することができます。



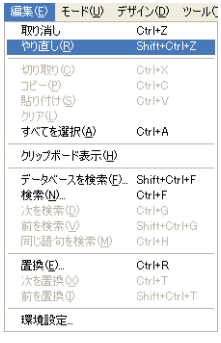
先行入力機能を有効／無効にするオプション

複数レベルの取り消し／やり直し

4D 2003の「メソッド」エディタでは、複数レベルでの取り消し／やり直し操作を行えるようになりました。実行したあらゆる動作（テキスト入力、削除、コピー／ペースト等）は、順次メモリ上に保存され、その実行とは逆の順序で取り消すことができます。これと同様に、取り消した各動作はやり直すことができます。

このようにして4th Dimensionは、実行された動作のうち直前までの20回分を保存します。

「編集」メニューや「メソッド」エディタのコンテキストメニュー、およびツールバーより、「取り消し／やり直し」コマンドを使用することができます。



複数レベルのコピー＆ペーストとクリップボードの番号付け

標準のコピー＆ペースト操作（従来通り）に加えて、4th Dimension 2003では新しく2種類の機能が提供され、さまざまなクリップボードの内容を使用して作業を行えるようになりました。

■ 4Dは、カレントセッション中に「メソッド」エディタ上で実行された“コピー”や“ペースト”動作のうち、直前までの10回分をメモリ上に保存します。このようにして保存された各種内容は、いつでも再使用することができます。

これを行うには、「メソッド」エディタのコンテキストメニューから「クリップボード履歴」コマンドを使用します。

コピー、またはカットされた項目のうち、最初の数項目が表示されます。ある項目を選択すると、現在のカーソル位置にその項目が挿入されます。

■ 10種類のカレントクリップボードに番号が振られ、キーボードショートカットを用いて直接使用することができます。

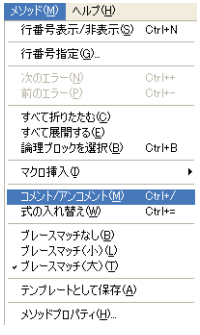
■ 特定のクリップボードに選択項目をコピーするには、Ctrl+Shift+1から9（Windows）またはcontrol+Shift+1から9（MacOS）を使用します。

■ 特定のクリップボードの内容をペーストするには、Ctrl+1から9（Windows）またはcontrol+1から9（MacOS）を使用します。

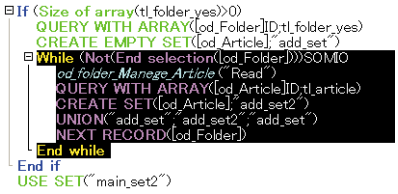
コメント／アンコメント

新しい「コメント／アンコメント」コマンドを使用して、選択された一連のコード行をコメントとして印付けすることができます。あるいは、これとは逆に、選択範囲からコメント記号を削除することもできます。

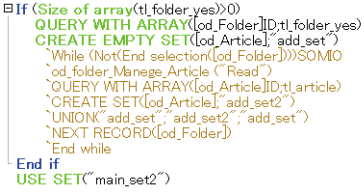
「コメント／アンコメント」コマンドは、「メソッド」メニューおよび「メソッド」エディタのコンテキストメニューより使用することができます。



このコマンドを使用するには、コメント文として印を付けるコードを選択し、次に「コメント／アンコメント」コマンドを選びます。



```
If (Size of array(tl_folder_yes)>0)
   QUERY WITH ARRAY([od_Folder]ID;tl_folder_yes)
   CREATE EMPTY SET([od_Article];"add_set")
   While (Not(End selection([od_Folder])))SOMO
      od_folder_Manege_Article ("Read")
      QUERY WITH ARRAY([od_Article]ID;tl_article)
      CREATE SET([od_Article];"add_set2")
      UNION("add_set";"add_set2";"add_set")
      NEXT RECORD([od_Folder])
   End while
End if
USE SET("main_set2")
```



```
If (Size of array(tl_folder_yes)>0)
   QUERY WITH ARRAY([od_Folder]ID;tl_folder_yes)
   CREATE EMPTY SET([od_Article];"add_set")
   `While (Not(End selection([od_Folder])))SOMO
   `od_folder_Manege_Article ("Read")
   `QUERY WITH ARRAY([od_Article]ID;tl_article)
   `CREATE SET([od_Article];"add_set2")
   `UNION("add_set";"add_set2";"add_set")
   `NEXT RECORD([od_Folder])
   `End while
End if
USE SET("main_set2")
```

選択範囲内にアクティブなコードだけが含まれている場合、「コメント」コマンドが適用されます。

注：4th Dimension 2003において、コメント長の制限は80桁ではなく、行の最大サイズである32,000桁までになります。

選択範囲内にアクティブなコードとコメント行とが混在している場合、コメント行にコメント記号（`）がさらに追加されます。このように、連続して行に“アンコメント”処理を行っても最初のコメントされた状態が維持されます。

選択範囲内にコメント行だけが含まれている場合、「アンコメント」コマンドが適用されます。

注：「コメント／アンコメント」コマンドは、行全体に対してのみ作用します。つまり、行の一部だけをコメント付けするために使用することはできません。

ドラッグ＆ドロップ

4th Dimension 2003では、メソッド作成時にドラッグ＆ドロップ動作を行うことができます。次の場合に、項目をドラッグ＆ドロップすることができます。

■ 同一メソッド内

■ 2つのメソッド間

注：以前のバージョンの4Dと同様に、テーブル名やフィールド名、フォーム名、プロジェクトメソッド名、コマンド名、定数名をメソッドに挿入するために、4Dエクスプローラからドラッグ＆ドロップを行うこともできます。

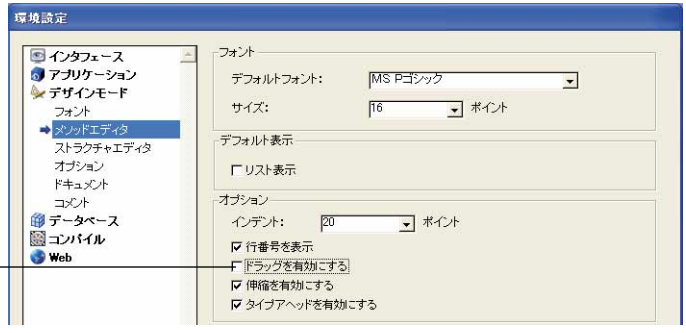
「メソッド」エディタにおいて、テキストの一部が選択されると即座にドラッグ＆ドロップ機能が有効になります。カーソルは図のような形 に変わり、選択範囲がドラッグ＆ドロップ可能であることを示します。

■ 選択範囲のコピー

デフォルトとして、ドラッグ＆ドロップ機能は選択したテキストを移動します。テキストをコピーするには、Ctrl キー（Windows）または Option キー（MacOS）を押しながらドラッグ＆ドロップしてください。

■ ドラッグ＆ドロップを無効にする

アプリケーションの「環境設定」ダイアログボックスの「メソッドエディタ」ページ（「デザインモード」テーマ）にある「ドラッグを有効にする」オプション（デフォルトでは選択済）を使用して、1つのメソッド内または2つのメソッド間のドラッグ＆ドロップ機能を有効／無効にすることができます。



ドラッグ＆ドロップを無効にする

注：「メソッド」エディタ内でドラッグ＆ドロップを無効にした場合でも、4D エクスプローラからオブジェクトをドラッグ＆ドロップして挿入することができます。

論理ブロックの選択

新しい「論理ブロックを選択」機能を使用して、カーソルポイントを含むコードの“論理ブロック”を選択することができます。論理ブロックは、次の要素で定義されます。

■ 引用符

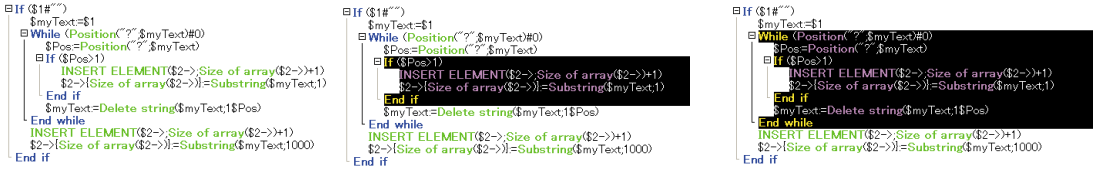
■ 括弧

■ 論理構造（If/Else/End if、While/End while、Repeat/Until、Case of/End case）

■ 大括弧

テキストブロックが既に選択されている場合、この機能はその次に高いレベルの論理ブロックを選択し、メソッド全体が選択されるまでその操作を続けます。

次の例は、目的のコードブロックを選択するために、2回続けて「論理ブロックを選択」機能を適用しています。



Ctrl+Shift+B（Windows）またはCommand+Shift+B（MacOS）を押すと、この操作が逆方向に働き、直前に選ばれた論理ブロックの選択が解除されます。

注：カーソルポイントが“If”または“Else”タイプの構造の中にある場合、論理ブロックは“If”文または“Else”文のうち、カーソルが位置するいずれか一方となります。

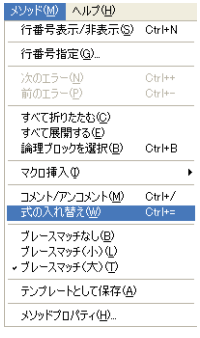
式の入れ替え

新しい機能である「式の入れ替え」を使用して、値を割り当てる式の項を入れ替えることができます。例えば、

variable1:=variable2
は以下のようになります。

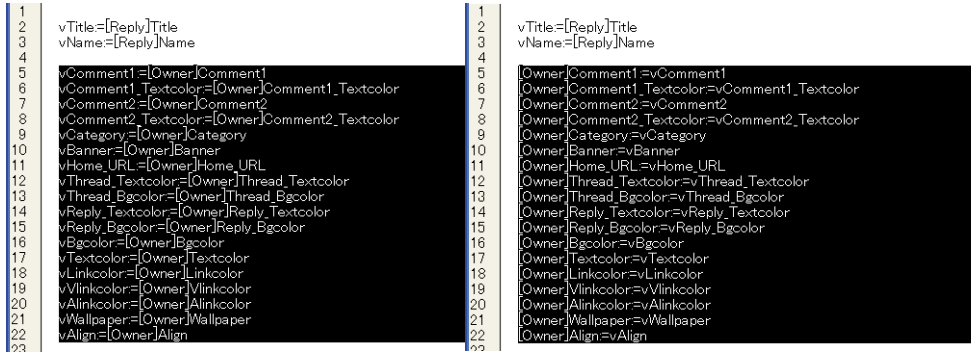
variable2:=variable1
この機能は、プロパティの取得や設定を行ったり、入力エラーを訂正するために使用した一連の割り当て値を逆転させる場合に非常に役立ちます。

この機能を使用するには、変更する行（複数）の選択後、「メソッド」メニュー、またはそのエリアのコンテキストメニューから「式の入れ替え」コマンドを選びます。



選択範囲の中で、値を割り当てている行だけが変更されます。

次に示す読み込みメソッドでは、メソッドの先頭にある変数割り当てエリアがコピーされ、次に「式の入れ替え」コマンドを使用して、そこに含まれている式が入れ替えられています。



```
vTitle:=[Reply]Title
vName:=[Reply]Name

vComment1:=[Owner]Comment1
vComment1_Textcolor:=[Owner]Comment1_Textcolor
vComment2:=[Owner]Comment2
vComment2_Textcolor:=[Owner]Comment2_Textcolor
vCategory:=[Owner]Category
vBanner:=[Owner]Banner
vHome_URL:=[Owner]Home_URL
vThread_Textcolor:=[Owner]Thread_Textcolor
vThread_Bgcolor:=[Owner]Thread_Bgcolor
vReply_Textcolor:=[Owner]Reply_Textcolor
vReply_Bgcolor:=[Owner]Reply_Bgcolor
vBgcolor:=[Owner]Bgcolor
vTextcolor:=[Owner]Textcolor
vLinkcolor:=[Owner]Linkcolor
vVlinkcolor:=[Owner]Vlinkcolor
vAlinkcolor:=[Owner]Alinkcolor
vWallpaper:=[Owner]Wallpaper
vAlign:=[Owner]Align
```

```
vTitle:=[Reply]Title
vName:=[Reply]Name

[Owner]Comment1:=vComment1
[Owner]Comment1_Textcolor:=vComment1_Textcolor
[Owner]Comment2:=vComment2
[Owner]Comment2_Textcolor:=vComment2_Textcolor
[Owner]Category:=vCategory
[Owner]Banner:=vBanner
[Owner]Home_URL:=vHome_URL
[Owner]Thread_Textcolor:=vThread_Textcolor
[Owner]Thread_Bgcolor:=vThread_Bgcolor
[Owner]Reply_Textcolor:=vReply_Textcolor
[Owner]Reply_Bgcolor:=vReply_Bgcolor
[Owner]Bgcolor:=vBgcolor
[Owner]Textcolor:=vTextcolor
[Owner]Linkcolor:=vLinkcolor
[Owner]Vlinkcolor:=vVlinkcolor
[Owner]Alinkcolor:=vAlinkcolor
[Owner]Wallpaper:=vWallpaper
[Owner]Align:=vAlign
```

冗長な文字列の管理

4th Dimension 2003の「メソッド」エディタでは、文字列の長さ制限は80桁のままですが、冗長な文字列の入力が自動的に管理されるようになりました。行の妥当性チェックの際に、エディタは長すぎる文字列を分割し、必要なシンタックス要素を挿入します。

入力

```
ALERT("80バイト以上の長い文字列の記述が可です。リテラルストリング内についても勝手に削除されることはなくなりました。")
```

妥当性確認後

```
ALERT("80バイト以上の長い文字列の記述が可です。リテラルストリング内についても勝手に削除"+"されることはなくなりました。")
```

エスケープ・シーケンスの使用

4D 2003の「メソッド」エディタでは、エスケープ・シーケンス（エスケープキャラクタとも呼ばれる）を使用することができます。エスケープ・シーケンスは一続きの文字で、“特殊な”文字を置き換える際に使用されます。

シーケンスは、バックスラッシュ（JISでは¥）とその後に続く1文字で構成されています。例えば、“¥t”はTabキャラクタのエスケープ・シーケンスです。エスケープ・シーケンスにより、特殊文字の入力をスムーズに行うことができます。先の例では、Character(Tab)の入力に代わって“¥t”を使用しています。

4th Dimensionにおいて、以下のようなエスケープ・シーケンスを使用することができます。

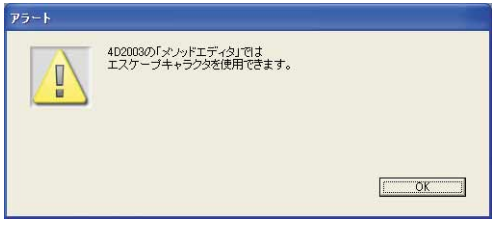
エスケープキャラクタ	置き換えられる文字
¥n	LF（ラインフィード）
¥t	HT（タブ）
¥r	CR（キャリッジリターン）
¥¥	¥（バックスラッシュ）
¥"	"（引用符）

注：エスケープ・シーケンスには、大文字と小文字のいずれでも使用することができます。

次の例では、以下に示すダイアログボックスを表示するために、構文中にキャリッジリターン文字（エスケープ・シーケンス“¥r”）を挿入しています。

```
ALERT("4D2003の「メソッドエディタ」では¥rエスケープキャラクタを使用できます。")
```

今までのバージョンの4th Dimensionでは、同じ構文が次のように書かれていました。



警告：Windowsにおいて、¥（円）記号はアクセスパスのセパレータとして使用されています。通常、4th Dimensionは「メソッド」エディタで入力されたWindowsのアクセスパスを正しく解釈するために、シングルの円記号（¥）をダブルの円記号（¥¥）に置き換えます。例えば、“C:¥Folder”は“C:¥¥Folder”となります。

しかし“C:¥MyDocuments¥New”と記述した場合4th Dimensionは“C:¥¥MyDocuments¥New”と表示します。この場合、二番目の円記号（¥）は“¥N”（既存のエスケープ・シーケンス）として誤って解釈されてしまいます。したがって、4th Dimensionが認識するエスケープ・シーケンスのいずれかで用いられる文字の前に円記号を使用したい場合には、必ずダブル円記号（¥¥）を入力してください。

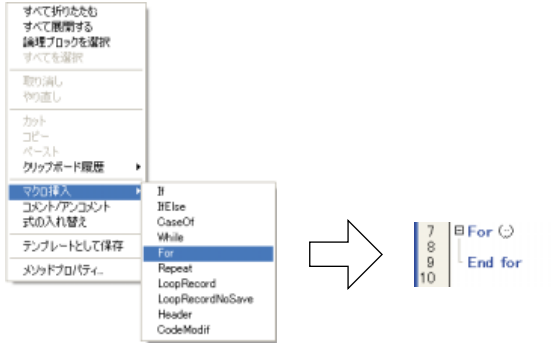
マクロの作成と使用

4th Dimension 2003では、メソッド内でマクロコマンドを使用することができます。マクロコマンドを使用すると、メソッド入力においてかなりの時間が節約できます。

マクロとは？

マクロコマンドは4Dコードの一部分であり、常に利用可能で、開かれているデータベースのタイプに関わらずメソッドのいずれの場所にでも挿入することができます。マクロに含めることができるのは、あらゆるタイプの4Dテキスト、コマンド、定数、およびマクロ挿入時にメソッドのコンテキストから算出された値で置き換えられる特殊なタグなどです。例えば、あるマクロに<method_name/>;というタグが含まれる場合、このタグはマクロ挿入時に現在のプロジェクトメソッド名で置き換えられます。

マクロ挿入の例



マクロは、XMLフォーマットの（テキスト）ファイルに保存されます。マクロは「メソッド」エディタのリストに配置したり、エディタのコンテキストメニューや先行入力機能を使用して呼びだすこともできます。

4th DimensionのマクロはXMLフォーマットで記述されます。4Dのデフォルトのマクロファイルを“そのまま”使用したり、あるいはこれを変更することもできます。

マクロの場所

マクロは各マシンに特定されており、そのマシン上で実行されるすべての4D 2003アプリケーションから利用することができます。マクロはすべて“Macros.xml”という名前のテキストファイルに保存され、そのマシンのアクティブな4Dフォルダ内に置かれます。

アクティブな4Dフォルダの場所は、OSによって異なります。

システム	アクティブな4Dフォルダの場所
MacOS 9	{Disk}:システムフォルダ:アプリケーションサポート:4D
MacOS X	{Disk}:Library:Application Support:4D
Windows 98 Windows Millennium	{Disk}:¥{System folder}¥All users¥Application Data¥4D
Windows 2000 Windows XP	{Disk}:¥Documents and Settings¥All Users¥Application Data¥4D
	4D Client: {Disk}:¥Document and Settings¥Current user¥Application Data¥4D （"Current user"はWindowsセッションを開いたユーザの名前）

注：Windows上では通常、デフォルトとして「Application Data」フォルダが非表示になっています。その内容を見るためには、OSレベルでこのファイルを表示させなくてはなりません。

デフォルトのマクロ

4Dより一連のデフォルトマクロが提供されますが、これは以前のバージョンの4Dにおけるキーワードリストに相当します。これらのマクロはデフォルトの「Macros.xml」ファイルに納められ、4D 2003を初めて起動した時にマシン上のアクティブな4Dフォルダ内に作成されます。

このファイルは、後から自由に変更することができます（後述の節を参照）。このファイルに問題が発生した場合、それを削除すれば次回の4Dの起動時にファイルが再作成されます。

カスタマイズしたマクロの追加

標準のテキストエディタやプログラムを用いて、「Macros.xml」ファイルにカスタマイズしたマクロを追加することができます。

4Dの使用中でもマクロファイルを開くことができます。4Dをアクティブにする度に、利用可能なマクロのリストが更新されます。例えば、テキストエディタを前面に移動してマクロファイルを変更した後、4Dメソッドへ戻ると、「メソッド」エディタ上では新しいマクロが使用できるようになります。

空のマクロやエラーがあるマクロは表示されません。

■ カスタマイズしたマクロの構文チェック

4Dマクロの定義に使用する言語はXMLであるため、Webブラウザで「Macros.xml」ファイルを開くだけで、マクロに構文エラーがないかどうかを確認することができます。Webブラウザは階層リスト形式でXMLファイルの内容を表示し、その構文を解析します。エラーが検出されれば（例えば、終了タグが抜けている）、ブラウザはエラー箇所を指摘します。

注：「Macros.xml」ファイルの“妥当性確認”、つまり、4Dから提供されるDocument Type Declaration（DTD）との整合性をチェックすることも可能です。詳細については、後述の付録B「マクロのDTD」を参照してください。

4Dマクロのシンタックス

4Dマクロは“要素”と呼ばれる、カスタマイズされたXMLタグを用いて構築されます。

タグには定義の開始と終了を示すものもあり（<tag> </tag>タイプのダブルタグ）、挿入内容の値で置き換えられるものもあります（<tag/>）。

XML仕様に準拠して、いくつかの要素タグには属性を含めることができます。特に指定されないかぎり、これらの属性は任意であり、省略されると省略値が使用されます。属性付き要素のシンタックスは、次の通りです。

■ ダブルタグ：<tag attribute="value"> </macro>

■ シングルタグ：<tag attribute="value"/>

要素が複数の属性を受け入れる場合、同じコマンド行にそれらの属性をまとめて記述し、スペースで区切ります。

<tag attribute1="value" attribute2="value" attribute3="value"... >
タグの一覧とその使用モードを以下に示します。

要素タグ	説明
<macros> </macros>	マクロファイルの開始と終了（必須タグ）
<macro> </macro>	マクロ定義の開始と終了とその属性
	属性： ・name：メニューおよび「メソッド」エディタのリスト上に表示されるマクロ名**（必須属性） ・type_ahead_text：先行入力機能*を用いたマクロ呼び出しの際に入力する文字列** in_menu：コンテキストメニュー*を用いたマクロ呼び出しが可能かどうかを示すブール値。値 = “true”（デフォルト）または“false” ・type_ahead：先行入力機能*を用いたマクロ呼び出しが可能かどうかを示すブール値。値 = “true”（デフォルト）または“false”
<selection/>	マクロ挿入時に選択テキストで置き換えられるタグ。選択範囲は空でも構わない。
<text> </text>	メソッドに挿入しなければならない開始コードと終了コード。コードの前後にはキャリッジリターンが追加される。
<caret/>	コード上でのマクロ挿入後のカーソルポイントの位置
<user_4D/>	カレント4Dユーザ名で置き換えられるタグ。
<user_os/>	カレントシステムユーザ名で置き換えられるタグ。
<method_name/>	カレントプロジェクトメソッド名で置き換えられるタグ。

要素タグ	説明
<date/>	現在日付で置き換えられるタグ。
	属性： ・format：日付表示に使用する4Dフォーマット。フォーマットが定義されていない場合、デフォルトのフォーマットが使用される。値 = 4Dフォーマットの番号（0～ 8）
<time/>	現在時刻で置き換えられるタグ。
	属性： ・format：時刻表示に使用する4Dフォーマット。フォーマットが定義されていない場合、デフォルトのフォーマットが使用される。値 = 4Dフォーマットの番号（0～ 6）
<clipboard/>	クリップボードの内容で置き換えられるタグ。
	属性： ・index：ペーストするクリップボード。値 = クリップボード番号（0～9）

* マクロは「メソッド」エディタのコンテキストメニューや先行入力機能を使用して呼びだすことができます（後述の節を参照）。

** XML言語仕様に準拠したい場合には、拡張文字（アクセント文字、引用符等）を使用してはいけません。

マクロ定義の例を以下に示します。

マクロ内容	コメント
<macros>	マクロXMLファイルの開始
<macro name="RecordLoop">	マクロ定義の開始とマクロ名
<text>	マクロコードの開始
For($i;1;Records in selection(<Selection/>))	<Selection/>タグは、マクロの挿入時に4D
SAVE RECORD(<Selection/>)	メソッド内で選択したコードによって
NEXT RECORD(<Selection/>)	置き換えられる（例：テーブル名）
End for	
</text>	マクロコードの終了
</macro>	マクロ定義の終了
</macros>	マクロXMLファイルの終了

マクロの呼び出し

デフォルトでは、「メソッド」エディタのコンテキストメニューや先行入力機能、「メソッド」エディタウインドウの下側にある特定のリストを使用して、マクロを呼び出すことができます。

マクロ毎に、コンテキストメニューや先行入力機能によるマクロ呼び出しを禁止できる点に注意してください。

■ コンテキストメニュー

デフォルトでは、「メソッド」エディタのコンテキストメニューの「マクロ挿入」階層コマンドを用いて、すべてのマクロを呼びだすことができます。<macro>タグの「in_menu」属性を使用して、このメニュー上にマクロを表示するかどうかを定義することができます。

コンテキストメニュー上では、「Macros.xml」ファイル内の順序でマクロが表示されます。したがって、このファイルを修正すると表示順を変えることができます。

■ 先行入力

デフォルトでは、先行入力機能（前述の「先行入力機能」の節を参照）を使用して、すべてのマクロにアクセスすることができます。入力したテキストは、マクロで置き換えられます。

<macro>タグの「type_ahead」属性を使用して、このタイプの操作からマクロを対象外にすることができます。

注：マクロに<selection/>タグが含まれる場合、そのマクロは先行入力機能のポップアップウインドウ上に表示されません。

マクロと他のタイプのオブジェクトを区別するため、先行入力機能のポップアップウインドウにおいてマクロ名は“< >”記号で括られます。

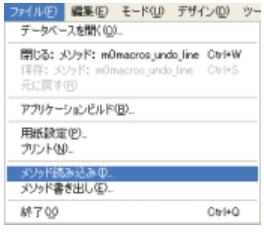
■「メソッド」エディタのリスト

マクロは「メソッド」エディタのリスト上に表示することができます（前述の「リストの表示とカスタマイズ」の節を参照）。マクロを呼びだすには、このリストからマクロ名をダブルクリックします。

このリストから特定のマクロを除外することはできません。

メソッドの読み込みと書き出し

4th Dimension 2003では、データベースメソッドやプロジェクトメソッド、オブジェクトメソッド、およびトリガの読み込みと書き出しをファイル形式で行うことができます。これらのコマンドは、「ファイル」メニューに含まれています。



「メソッド書き出し」コマンドを選択すると、標準の「ファイル保存」ダイアログボックスが表示され、ファイル名や保存場所、書き出しファイルのフォーマット（後述）を選択することができます。

印刷時と同様に、メソッド書き出しの際にはコード構造が縮小されていても考慮されず、コード全体が書き出されます。

「メソッド読み込み」コマンドを選択すると、「メソッド読み込み」ダイアログボックスが表示され、読み込むファイルを指定することができます。

読み込んだデータにより、メソッド内の選択部分のテキストが置き換えられます。読み込まれたメソッドで既存のメソッドを置き換えるには、読み込みを実行する前にメソッド内容全体を選択しておきます。

注：読み込み／書き出し機能はマルチプラットフォーム対応です。MacOS上で書き出されたメソッドは、Windowsに読み込むことが可能で、その逆も同様です。4Dは必要があればキャラクタの変換を行います。

ファイル形式

4Dは、2種類の形式でメソッドの書き出しや読み込みを行います。

■ 4Dメソッド（Windowsでは拡張子“.c4d”）：この形式では、暗号化形式でメソッドの書き出しが行われます。オブジェクト名は暗号化されます。この形式は、特に4th Dimensionアプリケーションとプラグイン間の異なる言語でのメソッド交換の際に使用されます。しかし、この形式のファイルはテキストエディタで表示することができません。

■ テキスト（Windowsでは拡張子“.txt”）：この形式では、テキストオンリー形式でメソッドの書き出しが行われます。

この場合、標準的なテキストエディタを使用してメソッドを読むことができます。しかし、書き出しと読み込みに使用する4Dアプリケーションの言語は同じでなければなりません。

例えば、テキストモードで4Dから書き出された次のメソッドの場合、

```
C_STRING(10;$1)

If (Count parameters=0)
   <>pr_Document:=New process("List_Documents";64*1024;"List of Documents";"Open";*)
   SHOW PROCESS(<>pr_Document)
   BRING TO FRONT(<>pr_Document)
Else
   If ($1="Open")
      MENU BAR(2)
      message_for_document:=""
      page_document:=Load list("od_Page_Document")

      COPY ARRAY(<>Document_types;Document_types)

      ALL RECORDS([od_Document])

      INPUT FORM([od_Document];"Input")
      OUTPUT FORM([od_Document];"List")
      $ref:=Open form window([od_Document];"Input")

      MODIFY SELECTION([od_Document];*)
      CLOSE WINDOW
      <>pr_Document:=0
   End if

   If (Is a list(page_document))
      CLEAR LIST(page_document;*)
   End if

End if
```

テキストエディタ上では次のように表示されます。

```
C_STRING(10;$1)

If (Count parameters=0)
<>pr_Document:=New process("List_Documents";64*1024;"List of Documents";"Open";*)
SHOW PROCESS(<>pr_Document)
BRING TO FRONT(<>pr_Document)
Else

If ($1="Open")
MENU BAR(2)
message_for_document:=""
page_document:=Load list("od_Page_Document")

COPY ARRAY(<>Document_types;Document_types)

ALL RECORDS([od_Document])

INPUT FORM([od_Document];"Input")
OUTPUT FORM([od_Document];"List")
$ref:=Open form window([od_Document];"Input")

MODIFY SELECTION([od_Document];*)
CLOSE WINDOW
<>pr_Document:=0
End if
```

検索と置換

4th Dimension 2003では、メソッドに関する検索と置換機能が変更されました。

■「検索／置換」コマンドは「編集」メニューに配置されています。

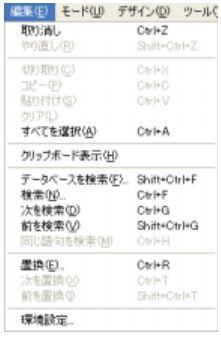
■ メソッドの検索は、4D標準の検索用ダイアログボックスと同様のダイアログボックスを用いて実行されます。

■「検索／置換」ダイアログボックスが機能強化されました。

■ 新しい「検索」コマンドが使用できます。

「検索／置換」コマンドへのアクセス

標準的なインタフェース基準に準拠して、メソッド（およびデータベーストラクチャ全体）の「検索／置換」コマンドは、4th Dimensionの「編集」メニュー上に配置されるようになりました。



「検索」および「置換」機能

「データベースを検索...」コマンドを使用すると、4Dデータベースを全体的に検索できます。このコマンドは「メソッド」エディタだけに限定されません。このコマンドに関する詳細は、前述の「検索コマンドの場所」ならびに4Dドキュメントの『デザインリファレンス』マニュアルを参照してください。

検索

「検索」コマンドを選択すると、次の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で検索条件を指定すると、前面に配置されているメソッド内で検索が実行されます。

■「検索語句：」エリアには、検索対象とする文字列を入力することができます。このエリアはコンボボックスであり、4Dセッション中に検索や置換の対象となった最近の15の文字列が保存されています。

■「ワード単位」オプションを使用すると、対象が検索語句と完全に一致した語句だけに限定されます。このオプションを選択すると、例えば“client”を検索する場合には“clients”や“myclient”は対象となりません。デフォルトでは、このオプションが選択されていません。したがって、“var”を検索すると、“Myvar”や“variation”等が対象語句であるとみなされます。

「データベースを検索...」コマンド（データベース全体を検索）のダイアログボックスの「タイプ：すべて」オプションとは異なり、「ワード単位」オプションではオブジェクト名が検索対象にならない点に注意してください。例えば、このオプションを使用して、メソッド中の“My”という文字列を検索した場合、変数“My Variable”がヒットします。しかし、「タイプ：すべて」オプションを使用して全体的な検索を行っても、前述の例の場合では同じ結果になりません。これは、（前に見つかった変数の）オブジェクト名全体が“My Variable”であり、入力した文字列（“My”）と完全に一致しないためです。

■「大文字小文字の区別」オプションを使用すると、「検索語句」エリアに入力した文字の大文字小文字の状態が考慮されます。例えば、“MyVar”を検索した場合には“myVar”は対象とみなされません。

■「検索方向：前／次」ラジオボタンを使用すると、検索を実行する方向を指定することができます。カーソルのある位置から開始して、現在のメソッドの先頭、または終わりに向かって検索を行います。

「OK」ボタンをクリックし、検索を実行します。定義した条件に一致する最初の項目が「メソッド」エディタウインドウ上で選択されます。この後、「編集」メニューの「次を検索」および「前を検索」コマンドを使用して、検索を継続することができます。

同じ語句を検索

「編集」メニューの「同じ語句を検索」コマンドを使用すると、選択された語句と一致する文字列を検索することができます。このコマンドは、「メソッド」エディタ上で少なくとも1つの文字が選択されている場合にのみ有効になります。

検索の実行は、カレントメソッドで「次を検索」した場合と同様に処理されます。

検索／置換

4Dの「編集」メニューの新しい「置換」コマンドにより、次のダイアログボックスが表示されます。

■「検索語句：」エリアを使用し、検索対象とする文字列または式を指定します。「検索」ダイアログボックスと同様に、このエリアはコンボボックスであり、検索の対象となった最近の15の文字列が保存されています。

■「置換語句：」エリアを使用し、上で指定した語句を置き換える文字列を定義します。このエリアもまた、コンボボックスであり、検索や置換の対象となった最近の15の文字列が保存されています。

■「ワード単位」オプションを使用すると、入力した文字列と完全に一致した語句だけが検索／置換対象となります。この場合、例えば、“client” を検索する場合には “clients” や “myclient” は対象となりません。

■「大文字小文字の区別」オプションを使用すると、入力された文字列の大文字小文字の状態が同じである語句だけが検索／置換対象となります。例えば、“MyVar”を検索した場合には“myVar”は対象とみなされません。

■「検索」ダイアログボックスと同様に、「検索方向：前／次」ラジオボタンを使用すると、検索を実行する方向を指定することができます。カーソルのある位置から開始して、現在のメソッドの先頭、または終わりに向かって検索を行います。

「置換」ボタンを使用して検索を開始し、検出された最初の語句を置換します。この後、「編集」メニューの「次を置換」および「前を置換」コマンドを使用して、検索／置換処理を継続することができます。

「一括置換」ボタンは、開かれているメソッド内で検索条件に一致する語句をすべてダイレクトに置換する際に使用します。

パフォーマンスと互換性

4th Dimension 2003の新しい「メソッド」エディタでは、以前のエディタでの制限が撤廃され、また互換性に優れています。

コメント長

コメントの最大長の制限は80桁ではなくなりましたが、コマンド行の最大長に準じます（理論上は32,000桁まで）。

メソッドサイズ

メソッドの最大サイズはテキスト32KBまでではなく、2GBまたはコマンド行が32,000行までに制限されます。この制限を超えると、これを超える行が表示されないことを知らせる警告メッセージが表示されます。

埋め込み構造

4th Dimension 2003では、プログラム構造（If/While/For/Case of）を512レベルの“深さ”まで埋め込むことができます。

インターナショナルスクリプト

4Dの新しい「メソッド」エディタは、2バイト文字セットを受け入れます。

ワイルドカード（@）

ワイルドカード（@）による操作は、新しい「メソッド」エディタでも維持されます。しかし、提供される機能は、先行入力機能のものよりも少し劣ります。

上位互換性と下位互換性

4D 2003の「メソッド」エディタは、以前のバージョンの4Dで記述されたメソッドと完全に互換します。

同様に、4D 2003で作成されたメソッドをバージョン6.8.xの4Dで再オープンすることができます。もちろん、この場合には4D 2003で導入された新しい機能を使用することはできません。

フローチャートエディタに関する注意

4th Dimension 2003では、フローチャートエディタが保持されていません。このエディタを使用してメソッドを作成することはできなくなりました。

既存のフローチャートメソッドのオープンや修正は可能ですが、これらのメソッドをリスト形式のエディタで再作成されることを強くお勧めします。次のバージョンの4th Dimensionからは、フローチャートエディタが完全に廃止されます。

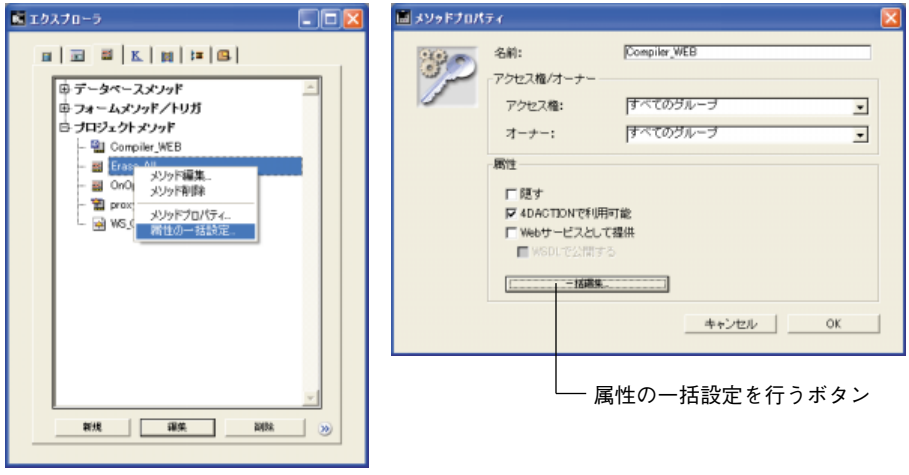
メソッド属性の一括設定

4th Dimension 2003では新しいダイアログボックスが提供され、ここでデータベースプロジェクトメソッド全体、またはその一部の属性（非表示、4DACTIONで利用可能など）を一回の操作で変更することができます。

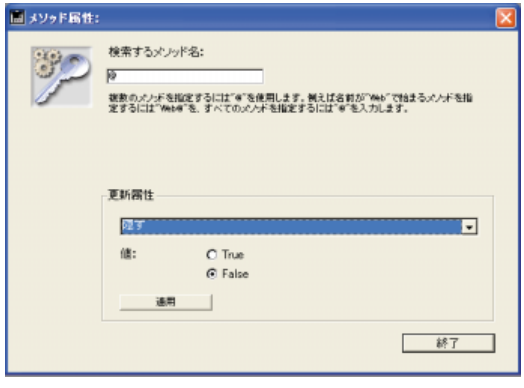
この新しい機能は、以前のバージョンの4Dから変換したデータベースのプロジェクトメソッド属性を変更する際に特に役立ちます。また、開発中にこの機能を使用して、同じようなメソッド群に対して、共通する属性を速やかに適用することもできます。

▼ メソッド属性の一括設定を行うには、次の手順に従ってください。

1 4Dの「エクスプローラ」において、プロジェクトメソッドを右クリック（Windows）またはcontrol+クリック（MacOS）し、「属性の一括設定...」コマンドを選択する。または、「メソッドプロパティ」ウインドウの「一括編集...」ボタンをクリックする。



次のダイアログボックスが表示されます。



2「検索するメソッド名：」エリアに、一括で修正しようとするメソッド名を指定する。

メソッド名を検索する条件として、文字列を使用します。一連のメソッドを指定できるようにワイルドカード記号（@）を使用してください。

- ある文字で始まる名前のメソッドを指定するには、文字列の最後に“@”を入力します。例：web@
- ある文字を含む名前のメソッドを指定するには、文字列の中に“@”を入力します。例：web@write
- ある文字で終わる名前のメソッドを指定するには、文字列の最初に“@”を入力します。例：@web
- すべてのメソッドを指定するには、このエリアに“@”だけを入力します。

注：

・検索では大文字小文字が考慮されません。

・例えば、dtro_@web@pro@ のように、文字列中に複数の“@”記号を入力することができます。

3「更新属性」エリアにおいて、ドロップダウンリストから属性を選び、適用する値に相当するラジオボタンをクリックする。

各種メソッド属性はすべて、所定の一括処理で変更することができます。

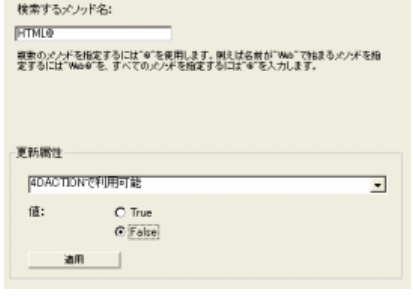
■ 隠す

■ 4DACTIONで利用可能（後述の「4DACTIONで利用可能」の節を参照）

■ Webサービスとして提供、またはWSDLで公開する（後述の「4th Dimensionを使用したWebサービス公開」の節を参照）

注：「WSDLで公開する」属性が真（True）に設定された場合、「Webサービスとして提供」属性が設定済であるプロジェクトメソッドにのみ適用されます。

この例では、“HTML”で始まる名前を持つすべてのメソッドに対し「4DACTIONで利用可能」オプションのチェックが解除されます。



4「適用」をクリックする。

入力した文字列で指定されたすべてのプロジェクトメソッドに対して、即座に変更が適用されます。

5 一連のメソッド毎、または適用する属性毎に、この処理を繰り返す。

6「終了」をクリックし、ダイアログボックスを閉じる。

4th Dimension 2003では、「ユーザ」モードにおいて標準の読み込み／書き出しダイアログボックスを使用することにより、XMLフォーマットでのデータ読み込みと書き出しがサポートされます。

注：4D 2003の新しいランゲージコマンドを使い、XML構造の内容を解析することができます（後述の「XML」の節を参照）。

XMLフォーマットでの読み込み／書き出し

4th Dimension 2003では、XMLフォーマットでのデータの読み込みと書き出しを行うことができます。

XML（eXtensible Markup Language）は、データ変換を行う際の標準言語です。この言語はタグの使用を基本とし、このタグにより交換データおよびその構造に関する正確な記述が可能となります。XMLファイルはテキスト形式ファイルで、その内容はデータを読み込むアプリケーションによって解析されます。今日では、数多くのアプリケーションがこのフォーマットをサポートしています。

4th Dimension 2003では、XMLを使用して他の4Dデータベースや他のアプリケーションから書き出されたXMLデータをテーブル内に読み込むことができます。また、レコード内容をXMLフォーマットのファイルに書き出すこともできます。

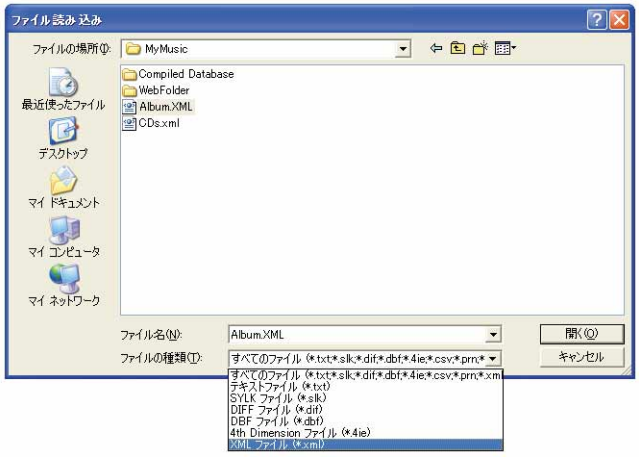
XMLという用語に関する詳細は、付録Aの「XML用語集」を参照してください。

XMLに関する詳細は、http://xml.orgおよび、http://www.w3.orgのサイトを参照してください。

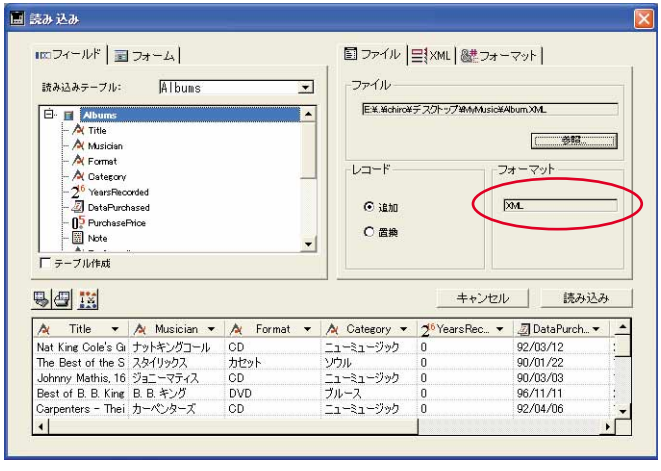
XMLフォーマットでのデータ読み込み

XMLフォーマットでデータを読み込むには、「ファイル」メニュー（「ユーザ」モード）の「データ読み込み...」コマンドにより表示される、4th Dimension標準の「ファイル読み込み」ダイアログボックスを使用します。

このコマンドを選択すると、標準の「ファイル読み込み」ダイアログボックスが表示されます。ファイルフォーマットを選択するメニューから、「XMLファイル」オプションを選べるようになりました。



XMLファイルを開くと、4th Dimensionのデータ読み込み用ダイアログボックスが表示されます。「フォーマット」エリア（入力不可）には読み込むファイルのタイプが示されます。

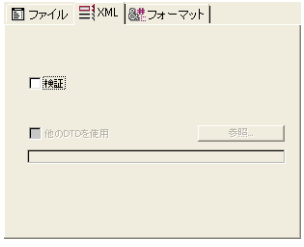


XMLファイルを読み込む際のこのダイアログボックスの操作は、他のファイルフォーマットの場合と同じです。つまり、4th Dimensionは読み込むフィールドを、選択されたテーブルの4Dフィールドに自動的に割り当て、それぞれ推定されるタイプに応じてデータのフォーマットを行います。この自動割り当ての結果はプレビューエリアに表示され、ユーザはこれを変更することができます。

フォーミュラでのデータ読み込みや、読み込みデータの受け入れ先となるテーブルの作成も可能です。

XMLタブ

XMLタブと呼ばれる特殊なタブを使用して、読み込まれるXML内容の解析モードを設定します。



データ読み込み時に、4th Dimensionは情報を取り出すためXML文書の内容を解釈します。デフォルトでは、特定の妥当性検証を行わずにこの操作を実行します。XML文書は“整形式”、つまりその構造は正しく、解釈における曖昧さがないものと見なされます。

ただし、「検証」オプションを選択すると、読み込み時に文書の“妥当性検証”を要求することができます。この場合、4th DimensionはそのDTD（Document Type Definition）に基づいて文書の内容を解析し、この定義に対応しているかどうかを調べます。文書が妥当である場合にのみ、読み込みが行われます。

注：DTDに関する詳細は、後述する「DTDオプション」の節を参照してください。

読み込まれる文書のDTDがその文書自体に含まれず、別のファイルに納められている場合や、あるいは他のDTDを使用して文書の検証を行いたい場合には、「替わりのDTDを参照」オプションの選択後、「参照」ボタンを用いてDTDを含むファイルを指定します。

読み込みが終了すると、ダイアログボックスが閉じられ、データが読み込まれたテーブルがカレントテーブルになります。

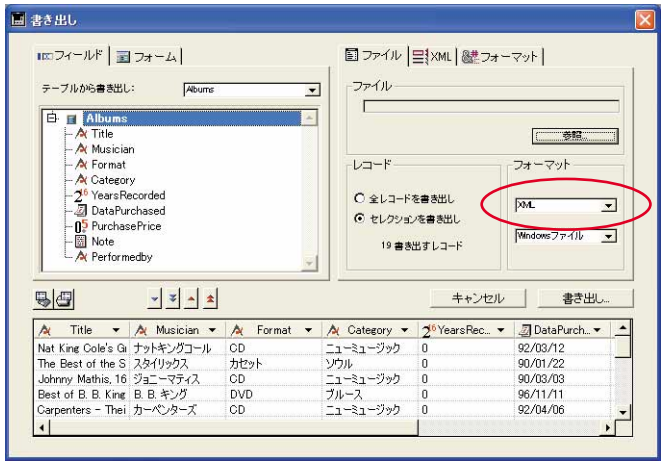
XMLデータの書き出し

4th Dimension 2003を使用して、XMLフォーマットでデータを書き出すことができます。XMLファイルはテキストファイルであるため、XMLフォーマットでのデータ書き出し方法は、他の書き出し用フォーマットの場合と同じです。例えば、書き出されるデータに対して表示フォーマットを指定することができます。

データ書き出し全般に関する情報は、4th Dimensionの『ユーザリファレンス』マニュアルを参照してください。

XMLフォーマットでデータを書き出すには、「ファイル」メニュー（「ユーザ」モード）の「データ書き出し...」コマンドにより表示される、4th Dimension標準の「書き出し」ダイアログボックスを使用します。

このコマンドを選択すると、標準のファイル書き出し用ダイアログボックスが表示されます。ファイル書き出し用フォーマットの選択メニューから、「XML」オプションを選べるようになりました。



このフォーマットを選択すると、新しい設定用タブである「XML」がダイアログボックスの右上に現われます。このページのパラメータを使用して、書き出されるXMLファイルの内容を定義することができます。

符号化方式（エンコード）

このポップアップメニューを使用し、XML文書に用いる符号化方式（すなわち、文字セット）を選択します。書き出されるデータ内容とそれを使用するアプリケーションにより、選択する符号化方式は異なります。

デフォルトでは、“ISO-8859-1(ISO Latin 1, West European)”コードが選択されます。

DTD オプション

XML フォーマットでの書き出しを行う際、4th Dimension では DTD（Document Type Declaration）を生成するかどうかを選択することができます。DTD には、XML が従わなくてはならない具体的な規定とプロパティ一式が記録されています。特に、これらの規定により、各タグの名前と内容、およびそのコンテキストが定義されます。

要素を形式的に定義することにより、XML 文書が“妥当”であることを確認でき、特に XML 文書内に繰り返し使用されるタグがある場合はこの規定が役立ちます。ただし、DTD は無くても構わないという点に注意してください。

DTD のハンドリングを定義するために、次の 3 つのオプションのいずれかを必ず選択してください。

- DTD なし（デフォルトオプション）：このオプションを選択すると、書き出し中に DTD が生成されません。
- DTD 生成：書き出し中に DTD を生成します。このオプションを選択すると、次のサブセットのラジオボタンにより、DTD を生成する場所を指定することができます。
  - XML 書き出しの中で：DTD は XML ファイル内に含まれます（内部 DTD）。したがって、生成された XML ファイルは独立しています。
  - 特定のドキュメントとして：DTD は別ファイル内に生成されます（外部 DTD）。外部 DTD は複数ユーザ間で共有できるので、異なるソースから生成された XML 文書構造を統一することができます。

注：4th Dimension では、テーブルとフィールドに同じ名前を付けることができます。しかし、XML 言語では異なる要素に対して同じ名前を使用することが禁止されています。したがって、「DTD 生成」オプションを使用する場合、書き出される 4D データに同じ名前を持つテーブルとフィールドが含まれてはいけません。さもなければ、生成された XML ファイルは妥当ではなくなり、XML パーサーで開けなくなります。

- 既存の DTD を参照：関連する「参照」ボタンを使用し、このオプションにより既存の外部 DTD ファイルを指定することができます。4th Dimension は書き出しファイル内にこの DTD への参照を含めます。

XSL ドキュメント

XSL（eXtensible Stylesheet Language）により、XML 文書で定義された各要素を視覚的に表現することができます。

この言語を使用して、XSL 文書の内容の処理や表示に用いるスタイルシートを概略的に定義します。

「既存のXSLを参照」オプションにより、関連する「参照」ボタンを使用して、XSLファイルを指定することができます。この結果、書き出されたXMLファイルには、このXSLファイルへの参照が含まれます。

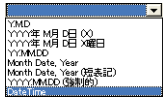
空白文字の置換

「空白を置換」オプションを選択すると、生成されたXMLファイルの値名フィールドにある“空白”文字が下線（“_”）で置き換えられます。XMLの値名フィールドには空白文字が許可されないため、デフォルトとして、このオプションが選択されています。

ただし、特別な目的のため、必要に応じてこのオプションの選択を解除することができます。もちろんこの場合、生成後のファイルは、XMLに関してW3Cが定義した全般的な構文規則に準拠しなくなります。

DateTime フォーマット

日付や時間タイプのデータの読み込みと書き出しに、新しい表示フォーマットである“DateTime”を使用することができます。



このフォーマットは、XMLの日付と時間の表現規格（ISO8601フォーマット）に対応しています。例えば、このフォーマットでは、“May 31, 2003 at 1:20 p.m.”という日付や時間は、“2003-05-31T13:20:00”と表現されます。

4th Dimensionでは、一つのフィールド内に日付と時間の両方を保存することは許可されません。しかし、次の方法でこのフォーマットを使用することができます。

- XML規格に従って日付や時間が保存されるように、このフォーマットでデータを書き出すことができます。日付の書き出しを行う場合、書き出された値のスタイルは“2003-05-31T00:00:00”のようになります。一方で、時間を書き出す場合には、そのスタイルは“0000-00-00T13:20:55”になります。
- このフォーマットで保存されたXMLデータを読み込むことができます。保存したい情報に応じて、データは日付または時間タイプのいずれかのフィールドに保存することができます。

4th Dimension 2003では、今まで4D Compilerアプリケーションを用いて利用していたコンパイル機能が組み込まれています。したがって、4th Dimensionアプリケーションを終了しなくても、データベースのコンパイルやコンパイル後のコードの検証を行えるようになります。

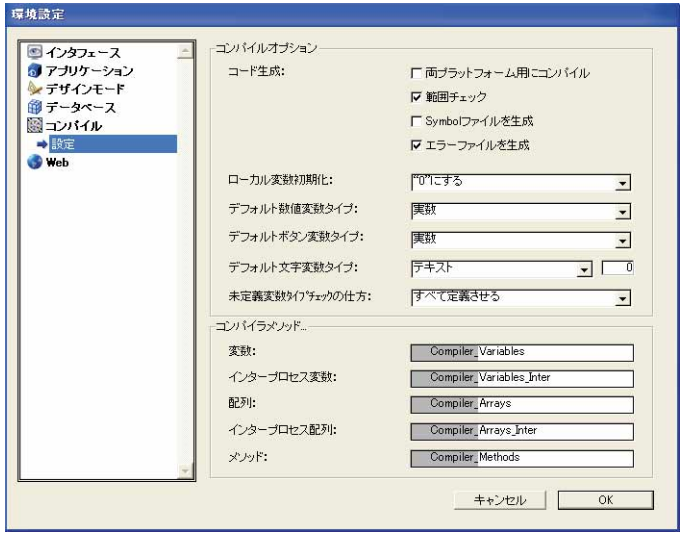
この原則を完全にするため、直接4Dからコンパイル済データベース（4D Engine組み込み済、または組み込まない）の作成が可能となり、この新しい機能によってデータベースの配備はよりスムーズに行えるようになります。

統合されたコンパイラ

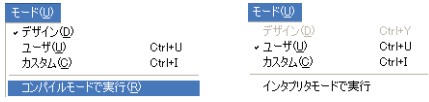
はじめに

4th Dimension 2003には、アプリケーションのコンパイル機能がそのまま統合されています。データベースのコンパイルは、4Dの「デザイン」モードに新しく追加されたダイアログボックスを使用して行います。

以前、4D Compilerアプリケーションで利用されていた全般的なコンパイルオプションの設定は、4Dの「環境設定」ダイアログボックスで行えるようになります。



データベースをコンパイルすると、「モード」メニューを使用して、インタプリタモードとコンパイルモードをいつでも切り替えることができ、4Dアプリケーションを終了する必要はありません。



また、データベースを開く際のダイアログボックスでは、データベースの起動時のモードとして、インタプリタモードまたはコンパイルモードのいずれかを選択することができます（前述の「データベースのオープンと作成」の節を参照）。

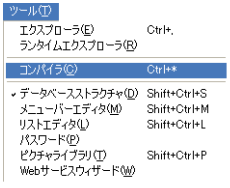
以前のバージョンとは異なり、4D 2003ではコンパイルコードとインタプリタコードを別々のファイルに分ける必要はありません。1つのストラクチャファイル内に、インタプリタコードだけ（今までのストラクチャファイルと同じ）、またはインタプリタコードとコンパイルコードの両方を納めることができます。

もちろん、コンパイルコードだけが含まれるアプリケーションを提供することもできます。この機能は、4th Dimensionのアプリケーションビルダ（後述の「アプリケーションビルダ」の節を参照）によって処理されるようになります。

4D Server：4D Serverや4D Clientからデータベースをコンパイルすることはできません（4D Clientでは「シンタックスチェック」機能だけが使用できます。後述の「コンパイラウインドウ」の節を参照してください）。コンパイルは、シングルユーザ版の4th Dimensionを使用した場合にのみ実行することができます。

コンパイラウインドウ

4th Dimension 2003では、コンパイルの実行は、「コンパイル」ダイアログボックスを使用して行います。このダイアログボックスには、「ツール」メニュー（「デザイン」モード）の新しいコンパイラコマンドを使用してアクセスすることができます。



注：データベースにメソッドが1つも存在しない場合、新しいコンパイラコマンドはグレー表示されます。

このウインドウを使用して、データベースのコンパイルの開始や（シングルユーザの4th Dimensionを使用した場合のみ）、メソッドのシンタックスチェックを行います。さらに、提供されるオプションを用いて、警告の表示や非表示、データベースの変数定義メソッドの生成や再生成、およびコンパイル済コードの削除を設定することができます。

注：

・データベースのコンパイルには、適切なライセンスが必要です。このライセンスが定義されていない場合、コンパイルを実行することができません（「コンパイル」ボタンはグレー表示されます）。ただし、シンタックスチェックと変数定義メソッドの生成は行うことができます。

・4D Clientでは、「シンタクッスチェック」と「生成」（変数定義メソッド）ボタンだけがアクティブになります。

■ シンタックスチェック：このボタンは、シンタックスチェックフェーズの実行を開始します。チェックが終了すると、検出されたエラーがすべて情報エリアに表示されます。

次の節で説明するように、エラー行をダブルクリックして、対応するメソッドを表示することができます。

■ コンパイル：このボタンは、即座にデータベースコンパイルプロセスを開始します。

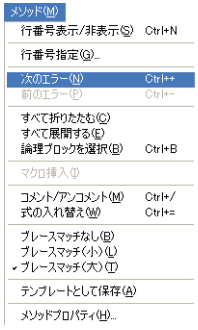
まず初めに、各種パスが実行され、「環境設定」ウインドウで定義された設定に基づいて、チェックや変数定義、初期設定を行います。

エラーが検出されなければ、実際のコンパイルが開始します。

エラーが検出されると、プロセスが中止され、ウインドウの情報エリアに問題となるメソッド名と行番号が階層リスト形式で表示されます。

問題となるメソッドを直接4Dの「メソッド」エディタ上で開くには、検出された各エラーをダブルクリックします。すると、エラーを含む行が高輝度で表示されます。

エディタの「メソッド」メニューから「前のエラー／次のエラー」コマンドを選択すると、エラーを含む行の間を移動することができます。



■ **警告表示**：4th Dimension 2003では、常に警告が有効になります。このオプションを使用して、コンパイラウインドウの情報エリアに警告を表示したり、または隠すことができます。

このオプションを選択すると、ウインドウには他のエラータイプの後に警告（存在する場合）が表示されます。また、警告はイタリック体で表示されます。

警告をダブルクリックすると、対応するメソッドが開かれます。

オプション

「コンパイラ」ウインドウには、2つの追加オプションがあります（デフォルトでは隠れています）。このオプションエリアを表示するには、展開用のボタンをクリックします。

このエリアには、次のボタンがあります。

■ **クリア**（コンパイル済みコード）：このボタンを使用すると、ストラクチャファイルのコンパイル後のコードが削除されます。ボタンをクリックすると、コンパイル中に生成されたすべてのコードが削除されます。4D Tools で圧縮を実行すると、ストラクチャファイルのサイズはそれに応じて縮小されます。

「モード」メニューの「コンパイルモードで実行」コマンドは無効になり、データベースを開くダイアログボックスの「コンパイルモードで開く」オプションはグレー表示されます。

このオプションを使用しても、生成されたコンパイラメソッドは削除されない点に注意してください。

■ **生成**（変数定義メソッド）：このボタンを使用すると、変数定義である「コンパイラメソッド」（後述の「コンパイラメソッド」の節を参照）が作成（または更新）されます。

これらのメソッドが既に存在する場合、その内容が更新されます。必要なコンパイラメソッド（つまり、データベースに既に存在する項目用のメソッド）だけが作成されます。

これらのメソッドのデフォルト名は、「環境設定」で変更することができます。情報エリアには、メソッドの作成や更新中に見つかったエラーが示されます。エラー行をダブルクリックすると、エラーの原因となるメソッドや行が「メソッド」エディタ上に表示されます。

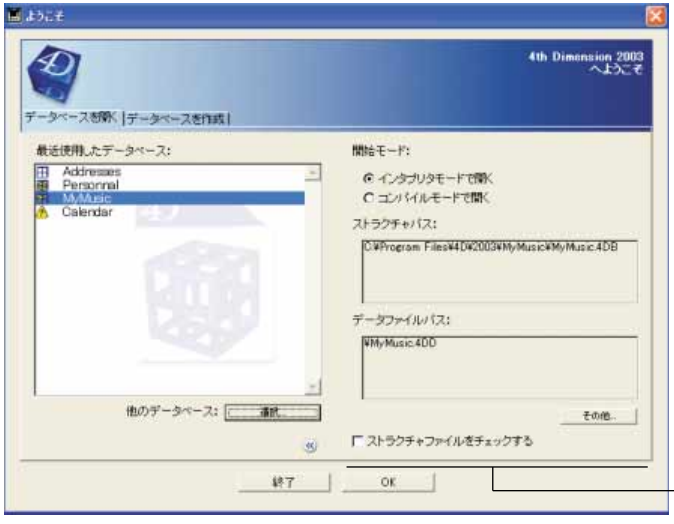
インタプリタモードとコンパイルモード間の移動

4th Dimension 2003 では、ストラクチャファイルにインタプリタ版とコンパイル版の両方のコードを納めることができます。データベースのコンパイル時に、コンパイル済コードがストラクチャファイル内に組み込まれます（別ファイルに配置する必要はありません）。

したがって、コンパイル後のデータベースはインタプリタモードでもコンパイルモードでも動作することができます。これにより、開発作業中にコンパイルモードでアプリケーションの動作状況を即座にチェックすることができます。

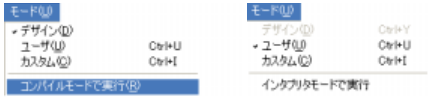
データベースにインタプリタ版とコンパイル版の両コードが含まれる場合、4th Dimensionでは起動時または使用中という2つの異なる実行モードを選択することができます。

■ **起動時**：データベースを開くダイアログボックスにおいて、オプションエリアの「インタプリタモードで開く／コンパイルモードで開く」ラジオボタンを使用し、データベースを開始するモードを選択します。



注：このダイアログボックスに関する詳細は、前述の「コンパイル版を開く」の節を参照してください。

■ **使用中**：「モード」メニューに、「コンパイルモードで実行」と「インタプリタモードで実行」（切り替えコマンド）というコマンドが追加されています。データベースが少なくとも1回はコンパイルされている場合、このコマンドがアクティブになります。



このコマンドを使用すると、いつでも実行モードを変更することができます。

注：インタプリタモードでデータベースストラクチャを変更した場合は、コンパイルモードに変更を反映させるため、再コンパイルしなくてはなりません。

モード変更とデータベースメソッド

あるモードから別のモードへ切り替える場合、4th Dimensionは現在のモードを終了して新しいモードを開きます。これはアプリケーションを終了して再オープンする作業に相当します。

この結果、モードの切り替の際に、4th Dimensionは定義されているデータベースメソッドがあれば、それを実行します。実行順序は次の通りです。

■「On Exit」データベースメソッド

■「On Startup」データベースメソッド

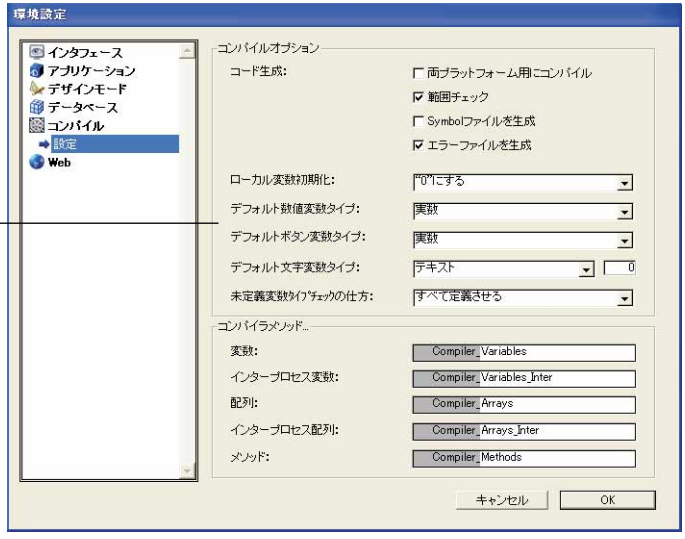
4D Server：4D Clientマシン上で、あるモードから別のモードへ切り替えても、接続している他のクライアントマシンのセッションは変わりません。

コンパイルの環境設定

全般的なコンパイルオプションは、アプリケーションの「環境設定[1]」ダイアログボックスで定義します。このダイアログボックスで指定した各オプションは、現行の4th Dimensionアプリケーションを使用して開かれたすべてのデータベースに対して適用されます。

「環境設定」ダイアログボックスを開くには、「編集」メニュー（MacOS X上ではアプリケーションメニュー）から「環境設定...」コマンドを選択します。次に、「コンパイル」テーマから「設定」ページを選びます。

1. アプリケーションの「環境設定」ダイアログボックスは、以前のバージョンの4Dの「データベースプロパティ」ダイアログボックスに替わるものです。詳細については、前述の「新しい環境設定ダイアログボックス」の節を参照してください。



全般的なコンパイルオプション

コンパイルオプションは、今まで4D Compilerで提供されていたものと概ね同じです。しかし、新しい機能もいくつか追加されています（「コンパイラメソッド...」エリア）。4D Compilerに含まれていた、陳腐化したオプションがいくつか取り除かれている点に注意してください（後述の「削除された4D Compiler 6.8.xのオプション」の節を参照）。

コンパイルの環境設定については、次の節で説明します。

コンパイルオプション

このエリアには、コンパイルプロセス中に使用される一般的なオプションが集められています。今まで、これらのオプションは4D Compilerで使用されていました。

■ **常に両プラットフォーム用にコンパイルする**：デフォルトでは、このオプションが選択されていない場合、4th Dimensionはアプリケーションが実行されるプラットフォームに対応したコンパイルコードを生成します。このオプションを選択すると、実行されたプラットフォームに関係なく、4th DimensionはPentium（Windows）およびPowerPC（MacOS）用のコンパイルコードを生成します。

■ **範囲チェック**：範囲チェックを有効または無効にするために使用します。範囲チェックとは補助的な検証を行うもので、特定の時点でのデータベースオブジェクトのステータスに応じて、その場でコードをチェックします。

■ **Symbolファイルの生成**：このオプションを使用すると、変数とそのタイプ、そのタイプを参照するメソッドの一覧を納めたASCIIタイプファイル（テキストのみ）が生成されます。また、シンボルファイルには、作成したメソッドや関数とともに、その引数や戻り値のタイプ（存在する場合）の一覧が納められます。このファイルは、データベースのストラクチャがあるフォルダ内に置かれ、次のような名前が付けられます。

■ Windows上では、データベース名.sym

■ MacOS上では、データベース名.symb

■ **エラーファイルを生成**：このオプションを使用すると、シンタックスチェックの際にエラーファイルが生成されます。このファイルには、全般的なエラーおよび特定行に関連するエラーや警告も記述されます。

コンパイラが検出したエラーはすべて、4th Dimension 2003の「メソッド」メニューから自動的にアクセスすることができます。しかし、あるマシンから別のマシンに送信できるエラーファイルがあると、特にクライアント／サーバ環境で複数の開発者が共に作業を行うフレームワークでは役立ちます。

エラーファイルは、その内容を自動的に解析しやすいように、XMLフォーマットで生成されます。また、エラー表示用に独自のインターフェースを作成することもできます。

エラーファイルには「データベース名.xml」という名前が自動的に付けられ、次の場所に保存されます。

■ **4th Dimensionの場合**：データベースのストラクチャファイルと同じ階層。

■ **4D Clientの場合**：4D Clientアプリケーションの「.exe」ファイルと同じ階層（Windows）、または4D Clientソフトウェアパッケージと同じ階層（MacOS）。

■ **ローカル変数初期化**：このオプションは、メソッドの初めでローカル変数を初期化するモードを定義するために使用します。

■ **“0”にする**：デフォルトとして、変数はゼロ（0）にリセットされます（文字タイプの場合は空の文字列、数値タイプの場合は0...）。

■ **ランダム値にする**：コンパイラは変数に対して常に同じランダム値を割り当てます（倍長整数には1919382119、文字列には“rgrg”、ブールタイプにはTrue（真）...）。このオプションにより、初期化を忘れたローカル変数を特定することができます。

■ **なし**：コンパイラは変数の初期化を行いません。初期化が正しく行われている場合、この設定によりデータベースの実行が速くなります。

■ **デフォルト数値変数タイプ**：このオプションを使用すると、明示的な方法で数値タイプの変数を実数または倍長整数タイプとして強制的に定義します。ただし、データベース内で記述されているコンパイラ命令に優先するものではありません。

■ **デフォルトボタン変数タイプ**：このオプションを使用すると、明示的な方法でボタン変数を実数または倍長整数タイプとして強制的に定義します。ただし、データベース内で記述されているコンパイラ命令に優先するものではありません。

■ **デフォルト文字変数タイプ**：このオプションを使用すると、明示的な方法で文字列変数をテキストまたは固定長文字列タイプとして強制的に定義します。ただし、データベース内で記述されているコンパイラ命令に優先するものではありません。

デフォルトの文字列タイプを固定長文字列に設定すると、入力エリアが表示されて、コンパイラに対し文字列の長さを指示することができます（必ず2から80までの値を入力する）。

■ **コンパイルパス**：このオプションを使用すると、コンパイラが実行するパスの数を定義することができ、その結果としてコンパイルの所要時間が短縮されます。

- **すべて定義させる**：コンパイルを行えるすべての段階でのパス。
- **ローカル変数のみ自動定義させる**：プロセス変数とインタープロセス変数を対象外とする変数定義のパス。
- **未定義変数タイプチェックの仕方**（自動変数定義は行わない）：ローカル変数、プロセス変数、インタープロセス変数を対象外とする変数定義のパス。

コンパイラメソッド

4th Dimension 2003では、“コンパイラ”プロジェクトメソッドを自動生成することができます。このメソッドには、すべての変数定義の宣言、プロセス配列とインタープロセス配列、およびローカル変数宣言が集められています。これらのメソッドが存在する場合には、コードのコンパイル時にコンパイラが直接これを使用します。

これらのメソッドはコンパイラウインドウを使用して生成されます（前述の「コンパイラウインドウ」を参照）。

コンパイラメソッドは5つまで生成することができます。データベースに対応する項目が含まれている場合にのみ、コンパイラメソッドが生成されます。

■ **変数**：プロセス変数の宣言を集める。

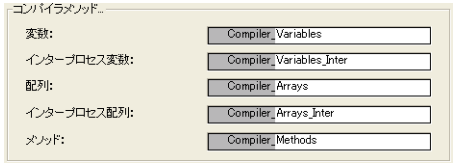
■ **インタープロセス変数**：インタープロセス変数の宣言を集める。

■ **配列**：プロセス配列の宣言を集める。

■ **インタープロセス配列**：インタープロセス配列の宣言を集める。

■ **メソッド**：メソッドの引数を示すローカル変数の宣言を集める（例：**C_INTEGER** (mymethod;$1)）。

これらのメソッド名は、対応するエリア内で変更することができます。



ただし、メソッド名は常に“Compiler_”というラベルで始まります（変更不可）。各メソッド名（接頭辞も含めて）はユニークでなければならず、また31桁を超えてはいけません。拡張ASCIIコード（アクセント文字、印刷用記号等）は許可されません。

削除された4D Compiler 6.8.xのオプション

4D Compilerで使用されていたコンパイルオプションのうち、陳腐化した、または不要となったオプションが4th Dimension 2003から削除されています。

- **スクリプトマネージャ**：生成されたコードは自動的にスクリプトマネージャとの互換性を持ちます。
- **警告**：4th Dimension 2003で提供される警告は、4D Compilerの「警告：詳細」オプションに相当します。警告は常にアクティブですが、コンパイラウインドウで表示／非表示を指定することができます。
- **プロセッサの種類**：陳腐化したオプション。
- **最適化**：コードは常に最適化されるようになりました。

アプリケーションビルダ

4th Dimension 2003には、配備目的のためのアプリケーションビルダが含まれています。この新しいビルダにより、4Dアプリケーションの完成プロセスや配備プロセスが容易になります。ビルダは各種OSに特定の機能、特にMacOS上でのソフトウェアパッケージの作成を自動的に処理します。

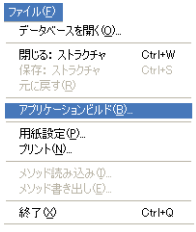
アプリケーションビルダを使用して、次の作業を行うことができます。

- インタプリタコードを除いた、コンパイル済データベースの作成（以前のバージョンの4D Compilerで提供されたコンパイルに相当）。
- ダブルクリックで起動するスタンドアロンアプリケーション、つまり4Dデータベースエンジンである4D Engineをマージしたアプリケーションの作成。

4D Server：アプリケーションビルド機能は、シングルユーザ版の4th Dimensionでのみ利用できます。

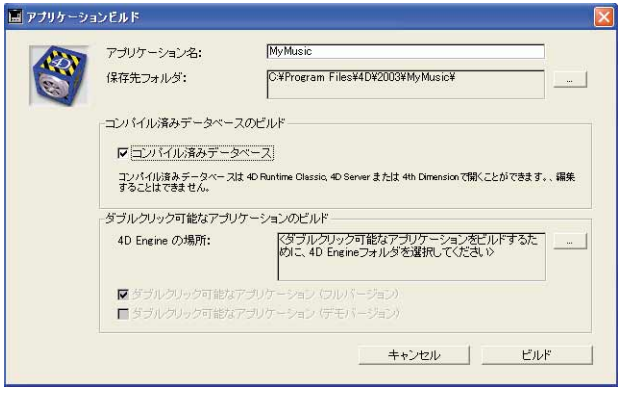
使用に関する原則

アプリケーションの構築は、「アプリケーションビルド」ウインドウを使用して実行します。このウインドウを表示するには、4Dの「ファイル」メニュー（「デザイン」モード）から「アプリケーションビルド...」コマンドを選択します。



注：データベースのコンパイル後にのみ、構築を行うことができます。データベースをあらかじめコンパイルせずにこのコマンドを選択した場合や、コンパイルコードとインタプリタコードが対応していない場合には、データベースの（再）コンパイルが必要であることを知らせる警告ダイアログボックスが表示されます。

次のようなアプリケーション構築用ウインドウが表示されます。



このウインドウを使用し、次の定義を行います。

■ 生成するファイルの名前と保存場所

■ 作成するコンパイル済データベースのタイプ

■ コンパイル済ストラクチャファイル

■ ダブルクリック可能なアプリケーション（組み込みライセンス付き、または無し）

■ ダブルクリック可能なアプリケーションの作成時に使用する4D Engineファイルの場所
構築中に、4th Dimensionは生成されたファイルを保存するための各種フォルダを自動的に作成します。

コンパイル後のデータベースに、WindowsコードとMacOSコードが共に含まれる場合[1]、コンパイル後ストラクチャファイルの対応するバージョンが生成されます。

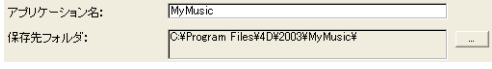
一方で、ダブルクリック可能なアプリケーションは現行のプラットフォームだけを対象とします。MacOSとWindowsの双方で利用可能なアプリケーションを構築するには、4th DimensionのMacOS版とWindows版の両バージョンを使用する必要があります。

このウインドウの各種オプションを設定したら、「ビルド」ボタンをクリックして指定した場所に目的のファイルを生成します。

「ビルド」ボタンをクリックすると、このウインドウで定義した各パラメータ（有効である場合）がデータベースのストラクチャファイルに保存されます。次にこのウインドウを使用する際には、これらのパラメータがデフォルトとして保持されます。したがって、「ビルド」ボタンをクリックするだけでファイルを再生成することができます。

アプリケーション名と保存場所の定義

アプリケーションを構築するウインドウの上部では、生成するファイルの名前と保存場所を定義することができます。



デフォルトとして「アプリケーション名」エリアには、データベースストラクチャファイルの名前が設定されています。生成されたファイルには、ここに指定した名前が使用されます（コンパイル済データベースおよびダブルクリック可能なアプリケーション）。4th Dimensionは、構築されるアプリケーションのタイプに応じて、必要な接尾辞を自動的に付加します（.4dc, .exe等）。

デフォルトの名前をそのまま使用する場合、データベースのストラクチャファイル名が変更されると、このアプリケーション名にも反映されます。

1. このオプションは、アプリケーションの環境設定で定義します（前述の「コンパイルの環境設定」の節を参照）。

アプリケーション名を変更した場合、現在のデータベースで新しく構築が行われるたびに、その新しい名前がデフォルトとして使用されます。したがって、入力する名前には拡張子を使用しないでください。さらに、OSで禁止されている文字（Windowsでは“.?!”、MacOSでは“:”等）を使用してはいけません。

「保存先フォルダ」エリアには、生成された各項目を配置する場所を指定します。デフォルトでは、データベースのストラクチャファイルを含むフォルダが選択されています。このデフォルトの場所をそのまま使用すると便利です。つまりこの場合は、保存先となるフォルダと4D Engineフォルダ（必要な場合）がストラクチャファイルとの相関関係で保存されるためです。その結果、アクセスパスを再定義しなくても、あるプラットフォームから別のプラットフォームへフォルダを再コピーすることができます。その上、ダイアログボックスの操作を中断しなくても、保存先のフォルダの移動やリネームを行うことができます。

保存先フォルダを変更するには、表示エリアの右側にある選択用のボタン「...」をクリックします。すると、フォルダを選択するダイアログボックスが表示され、新しい保存先フォルダを指定することができます。このダイアログボックスを有効にすると、フォルダのフルアクセスパスが表示されます。

次回、現在のデータベースで構築を行うたびに、この新しい場所がデフォルトとして使用されます。

構築時に、4th Dimensionは、指定された場所に1つ以上の中間フォルダを自動的に作成します（リクエストされた構築タイプに応じて、“コンパイル済みデータベース”、“ファイナルアプリケーション”、または“デモアプリケーション”という名前が付けられます）。これにより、同じ名前のファイルを誤って削除する危険を回避し、複数タイプの構築を同時に実行することができます。

コンパイル済データベースの作成

アプリケーションビルドウインドウのこのオプションを使用し、コンパイル済コードだけを含むデータベースを作成します。

この構築タイプにより、以前のバージョンの4D Compilerで提供されたコンパイルと同じ結果が得られます。

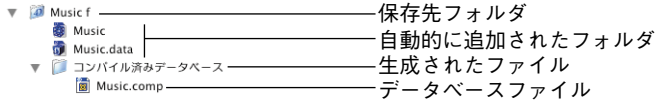
例えば、構築を実行した後、「アプリケーション名」エリアに“MyDatabase”と指定した場合、次のファイルが生成されます。

■ Windows上では2ファイル（MyDatabase.4dcおよびMyDatabase.rsr）

■ MacOS上では1ファイル（MyDatabase.comp）

「.4dc」ファイル（Windows）または「.comp」ファイル（MacOS）は、4th Dimension、4D Server、または4D Runtime Classicで開くことができます。ただし、このデータベースの「デザイン」モードにアクセスすることはできません。

構築中に、4th Dimensionはストラクチャファイルのインタプリタコードを削除します。コンパイル後のデータベースは「コンパイル済みデータベース」という名前のサブフォルダ内に置かれます。このサブフォルダは、指定された「保存先フォルダ」内に作成されます。



注：生成されたファイルの名前とアクセスパスに関する詳細は、前述の「アプリケーション名と保存場所の定義」の節を参照してください。

コンパイル済データベースを再構築する前に、4th Dimensionは「コンパイル済みデータベース」フォルダの以前の内容を消去します。したがって、保持しておきたいコンパイル版や追加項目があれば、あらかじめ必ず移動しておいてください。

ダブルクリック可能なアプリケーションの構築

4th Dimension 2003では、ダブルクリック可能なアプリケーションをデータベースから直接作成することができます。必要となるのは、4D Engine、4Dデータベースエンジン、そして適切なライセンスだけです。

その方法は、コンパイル後のストラクチャファイルと4D Engineとをマージすることです。この機能は、以前4D Compilerで提供されていましたが、別の手順が必要でした（特にMacOS上では、4D Package Makerアプリケーションを使用してソフトウェアパッケージを作成しなければなりません）。

コンパイル済4Dデータベースの実行形式版（.exe）は、「ダブルクリック可能なアプリケーションのビルド」機能を使用して4th Dimension 2003から直接作成することができます。

MacOS上では、この機能によりソフトウェアパッケージの作成も処理されます。さらに、アプリケーションの“デモ版”（つまり、ライセンス番号を含まない）を作成することもできます。

4D Engine フォルダの選択

ダブルクリック可能なアプリケーションを作成できるように、まず初めに4D Engineフォルダの場所を指定しなければなりません。対応するエリアにフォルダが指定されていない場合や、指定されたフォルダに有効な4D Engineが含まれていない場合には、ダブルクリック可能なアプリケーションを作成するオプションがグレー表示されます。

4D Engineファイルが含まれるフォルダと（Windows上では）関連するファイルも選択しなければなりません。

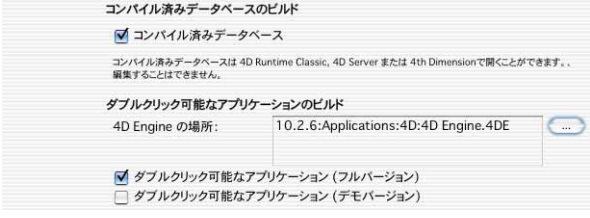
■ Windowsにおいて、このフォルダ内に4DEngine.4DE、4DEngine.RSR、ASINTPPC.DLL、ASIPORT.RSR、およびasifont.mapとASIFONT.FONファイルが含まれます。これらの項目は、選択したフォルダと同じ階層に配置されていなくてはなりません。

■ MacOSにおいて、4D Engineは各種汎用ファイルを含むソフトウェアパッケージの形式で提供されます。

4D Engineフォルダを選択するには、「4D Engineフォルダ」表示エリアの右側にある「...」ボタンをクリックします。

- Windows上では、フォルダを選択するダイアログボックスが表示され、4D Engineファイルが含まれるフォルダを指定することができます。
- MacOS上では、標準の「ファイルを開く」ダイアログボックスが表示され、4D Engineソフトウェアパッケージを選択することができます。

フォルダを選択すると、フォルダのフルアクセスパスが表示されます。4D Engineが実際に含まれている場合、ダブルクリック可能なアプリケーションを作成するオプションがアクティブになります。



ライセンス版とデモ版

4th Dimension 2003では、2種類のダブルクリック可能なアプリケーションを作成することができます。

■ **ライセンス版**：4Dの実行形式ファイルを配付するには、4th Dimensionおよびデータベースで使用するプラグインの適切な配備用ライセンスが必ず必要となります。

これらのライセンスは、4th Dimensionのライセンス番号を管理するダイアログボックス（4Dの「ヘルプ」メニューからアクセス）に入力します。

このオプションを選択すると、4th Dimensionは、そのマシン上にある配備用ライセンス番号を組み込んだダブルクリック可能なアプリケーションを作成します。この結果、アプリケーションは完全に動作可能となります。

■ **デモ版**：このオプションを選択すると、4th Dimensionは作成時にライセンス番号を組み込みません。したがって、アプリケーションは“デモ”モードでのみ動作します（テーブル数とレコード数が制限される）。

この2つのオプションを同時に選択することができます。すると、各タイプのアプリケーションが保存先フォルダの指定したサブフォルダ内に作成されます。

■ ライセンス版は、「ファイナルアプリケーション」という名前のサブフォルダに作成されます。

■ デモ版は、「デモアプリケーション」という名前のサブフォルダに作成されます。

生成されたファイルの名前、ならびにアクセスパスに関する詳細は、前述の「アプリケーション名と保存場所の定義」の節を参照してください。

「ビルド」ボタンをクリックすると、4th Dimensionは進捗サーモメータを表示し、各フェーズを実行中であることを示します。新しいパラメータが有効である場合には、それが保存されます。

注：「キャンセル」ボタンをクリック、または処理中にエラーが発生した場合には、生成されたファイルが削除され、処理中断の原因を知らせる警告ダイアログボックスが表示されます。

生成されるファイル

作成が終了すると、保存先フォルダの「ファイナルアプリケーション」サブフォルダ、または「デモアプリケーション」サブフォルダ内に次のファイルが生成されています。

■ Windows

■ 実行形式ファイルである「データベース名.EXE」

■ コンパイル済ストラクチャである「データベース名.4DC」と「データベース名.RSR」

■「ASINTPPC.DLL」、ASIPORT.RSRファイル、4D Extensionsフォルダ、「asifont.map」、ASIFONT.FONファイル

■ 4D Engineフォルダに追加された付加的な項目（後述の「4D Engineフォルダのカスタマイズ」の節を参照）

これらの項目はすべて、実行形式ファイルが動作できるように同じフォルダ内に配置しておかなければなりません。

■ MacOS

■「データベース名.app」という名前のソフトウェアパッケージ。これには、作成したアプリケーションとその操作のために必要となるすべての項目が含まれています。

注：MacOSにおいて、4Dランゲージの Application file コマンドは、「.comp」ファイル（ソフトウェアパッケージの「Contents:Resources」フォルダ内）ではなくApplicationNameファイルのアクセスパス（場所は、ソフトウェアパッケージの「Contents:MacOSClassic」または「Contents:MacOS」フォルダ内）を返すようになりました。

■ 両プラットフォームともに、必要があれば、ライセンス版に組み込まれたライセンス番号のリストが「Licenses_log.txt」ファイルに登録され、自動的に「ファイナルアプリケーション」フォルダ内に置かれます。

ダブルクリック可能なアプリケーションの再作成

ダブルクリック可能なアプリケーションを再作成する前に、4th Dimensionは「ファイナルアプリケーション」フォルダ、または「デモアプリケーション」フォルダの内容を消去します。したがって、残しておきたいバージョンがあれば必ず移動しておいてください。

4D Engine フォルダのカスタマイズ

ダブルクリック可能なアプリケーションを作成する際に、4th Dimensionは4D Engineフォルダの内容を保存先フォルダの「ファイナルアプリケーション」または「デモアプリケーション」サブフォルダにコピーします。

このため、必要に応じてオリジナルの4D Engineフォルダの内容をカスタマイズすることができます。

例えば、次のような事柄を行うことができます。

■ 4D Customizer Plusユーティリティを使用し、4D Engineの操作をカスタマイズする。

■ 特定の言語に対応する4D Engineバージョンをインストールする。

■ アプリケーションの運用に必要なプラグインを含む「Mac4DX」または「Win4DX」フォルダを追加する。

■「4D Extensions」フォルダの内容をカスタマイズする。

注：MacOSでは、4D Engineはソフトウェアパッケージの形で提供されます。これを変更するには、まず初めにパッケージ内容を表示しなければなりません（アイコンをcontrol+クリックする）。また、「Mac4DX」フォルダと「Win4DX」フォルダは必ず「Resources」フォルダと同じ階層に配置してください。

アプリケーションアイコンのカスタマイズ

4th Dimensionは、ダブルクリック可能なアプリケーションに対してデフォルトアイコンを関連付けます。しかし、アプリケーションごとにアイコンをカスタマイズすることができます。

■ MacOS 9

MacOS上でコンパイル済アプリケーションをカスタマイズするには、署名および新規作成したアプリケーションのアイコンを変更しなくてはなりません。これを行うには、ResEdit®のようなリソースエディタを使用します。

アプリケーションの署名を変更するには、2種類の操作が必要です。

■ 第一に、BNDLタイプのリソースを開きます。アプリケーション署名の4文字を新しいアプリケーション署名として選んだ4桁の文字で置き換えます。アプリケーション起動時の混乱を回避するため、この署名がユニークであることを確認してください。他のプログラムの署名を使用した場合、Finderにより別のアイコンに切り替えられる可能性があります。

■ 第二に、SIG*リソースを開き、BNDLリソースに割り当てたものと同じ4文字をコピーします。

■ お使いのリソースエディタで自動的に処理されない場合には、アプリケーションに与えた署名のリソースと同じタイプのリソースを作成します。このリソースのID番号は0となり、例えば作成したアプリケーションの名前を含みます。

次に、作成したアプリケーションのアイコンをカスタマイズします。このアイコンはicl8、icl4、ICN#、ics#、ics4およびics8リソース内に置かれています。

1 これら各リソースにおいて、ID番号128のアイコンを開く。

2 リソースエディタを使用して、このアイコンを希望通りに変更する。

この段階で処理を中断すると、アプリケーションアイコンはカスタマイズされますが、作成されるファイル（データファイル、書き出しファイル等）には標準の4th Dimensionアイコンが使用されます。128より大きなID番号を持つアイコンも、同じ手順でカスタマイズすることができます。

3 再度icl4リソースを開く。

リスト上の各アイコンは4th Dimensionにより生成されるファイルのタイプ（データ、ASCII書き出し等）に対応しています。

4 これらのアイコンを希望通りに変更する。

この変更は、カスタマイズされたコンパイル済データベースを使用して作成されたファイルに対してのみ影響します。今までのファイルはオリジナルアイコンのままです。

注：

・アプリケーションを市場に出したい場合、まず第一にAppleが定める条件に従う必要があります。これにより、署名がユニークであり、アイコンのコンフリクトを引き起こさないということを確認できます。

・変更はリアルタイムで実行されない可能性があります。この場合、表示の前にデスクトップの再構築を行わなければなりません。再構築を行うには、Commandキーと Optionキーを押しながらMacintoshを再起動します。

■ MacOS X

ダブルクリック可能なアプリケーションを作成する際に、4th Dimensionがアイコンのカスタマイズを処理します。

これを行うには、アプリケーションファイルの作成に先だって、次の操作を実行しなくてはなりません。

アイコンファイル（icnsタイプ）を作成し、インタプリタ版のストラクチャファイルと同じ階層に配置します。

注：Apple社から、.icnsアイコンファイルの作成用に特定のツールが提供されています。

作成したアイコンファイルには、インタプリタ版のストラクチャファイルと同じ名前を指定し、拡張子“.icns”を付加しなくてはなりません。

4th Dimensionは、ダブルクリック可能なアプリケーションを作成する際に自動的にこのファイルを考慮します。(.icnsファイルは「アプリケーション名.icns」とリネームされ、「リソース」フォルダに再コピーされます。“info.plist”ファイルのCFBundleFileIconエントリが更新されます)。

■ Windows

ダブルクリック可能なアプリケーションを作成する際に、4th Dimensionはアイコンのカスタマイズを処理します。

これを行うには、アプリケーションファイルの作成に先だって、次の操作を実行しなくてはなりません。

アイコンファイル（拡張子“.ico”）を作成し、インタプリタ版のストラクチャファイルと同じ階層に配置します。作成したアイコンファイルには、インタプリタ版のストラクチャファイルと同じ名前を指定し、拡張子“.ico”を付加しなくてはなりません。

4th Dimensionは、ダブルクリック可能なアプリケーションを作成する際、自動的にこのファイルを考慮します。

はじめに

4th Dimension 2003 には、お使いのデータベース内で Web サービスの公開や使用を行える新しい機能が導入されています。

Web サービスとは？

Web サービスは、一連の機能をエンティティ（実体）として集めたものであり、ネットワーク上に公開されます。Web サービスに対応し、ネットワークに接続された任意のアプリケーションから、これらの機能を呼び出したり、使用することができます。

Web サービスを使用し、運送会社の貨物配送管理や e コマース、市場価格の監視など、あらゆる種類のタスクを実行することができます。

このサービスを公開するプログラムは、“サーバ”と呼ばれます。Web サービスに対応しているアプリケーションはすべて、これらの機能のいずれかを利用することができます。これが“クライアント”プログラムです。

Web サービスの利点は、さまざまな情報システムとの相互運用が可能であるところです。つまり、システムが機能するために、サーバ側とクライアント側のプログラムが互いに互換性を持つ必要はありません。

クライアントアプリケーション側からみると、Web サービスは“ブラックボックス”のようなものです。これは、ある値を Web サービスに送信すると、処理結果である別の値が返されるためです。

サーバが提供する Web サービスは、パブリックまたはプライベートのいずれでも構いません。インターネット上にはパブリックタイプの Web サービスが多数提供されており、任意のアプリケーションが無料で要求を送信することができます。

W3C（World Wide Web Consortium、インターネットに関する管轄機関）やコンピュータ産業の主要企業により管理され、Web サービスは信頼性が高く永続的で機能向上可能なコネクティビティソリューションの代名詞となっています。

Webサービスの運用－主な定義

Webサービスは、原則的にHTTP通信プロトコルを使用して転送されます。

SOAP：Webサービスは、“オープンな”高レベルの通信プロトコルであるSOAP（Simple Object Access Protocol）を使用します。このプロトコルは、メッセージ構造（エンベロープ）レベルと交換データレベルの双方において、全面的にXML言語をベースとしています。このプロトコルの運用は、RFC（Request for Comment：インターネットに関するさまざまな局面を標準化する文書）により定義され、その広範囲にわたる互換性の高さが保証されています。

Webサービスの運用原理は、次の通りです。まず、WebサービスのクライアントがSOAPプロトコルを介し、XMLでリクエストをサーバに送信します。サーバはこのリクエストを解析し、要求された処理を実行した後、同じプロトコルと言語を使用して応答を返します。

WSDL：通常、Webサービスのサーバは、提供されるサービスに関するアクセス仕様を定義するためにWSDL（Web Service Description Language）を公開します。WSDLを用いて、Webサービスのサーバは提供するサービスの“操作説明書”を公開することができます（URL、メソッド一覧、パラメータ等）。これはXMLファイルの形式で記述され、一般的にサーバアプリケーション自体が作成します。このファイルは、必須ではありません。

UDDI：UDDI（Universal Description Discovery and Integration）は、すべてのパブリックWebサービスが登録されている世界規模のデータベースです。ただし、Webサービスをパブリックとすることは義務ではなく、また大抵の場合は、この登録が不要である点に注意してください。

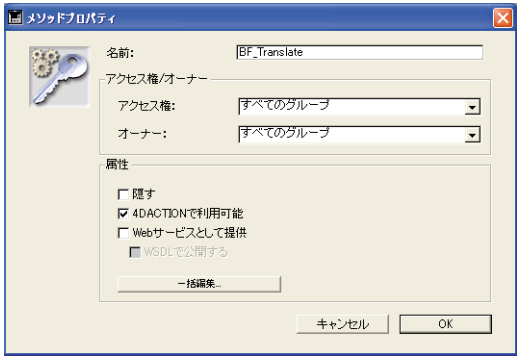
UDDIに関する詳細は、アドレスhttp://www.uddi.org/を参照してください。

Webサービスの4Dへの統合

4th Dimension 2003は、Webサービスのサーバやクライアントとして使用することができます。Webサービスと4th Dimensionとの統合は単純かつ安全です。いくつかの設定を行うことにより、公開と認可の状況を正確に監視することができます。

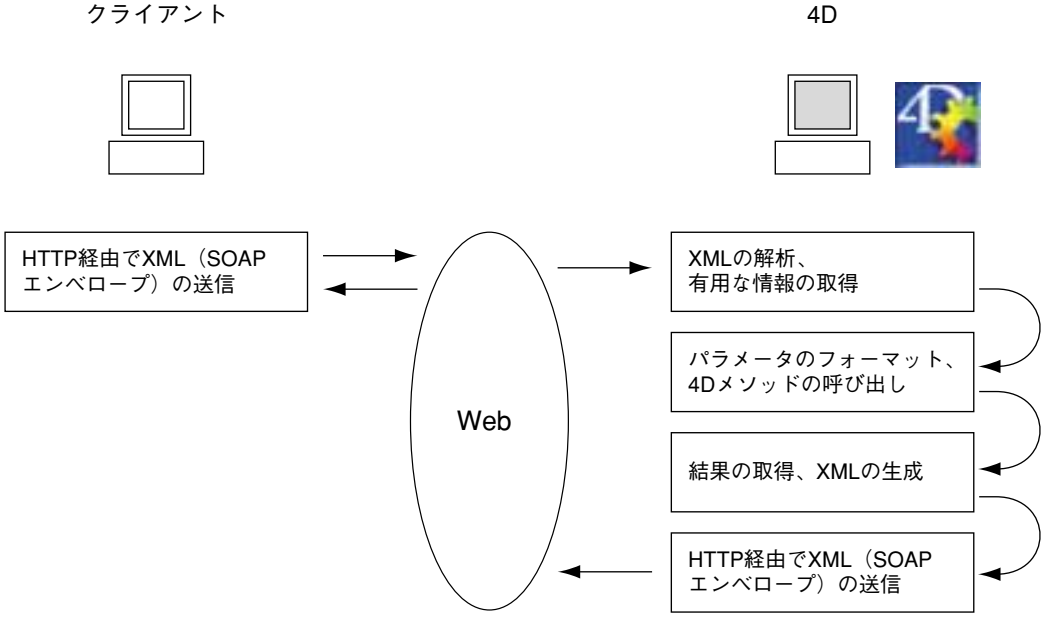
Webサービスサーバとしての4D

特に大きな変更を加えなくても、任意のプロジェクトメソッドをWebサービスとして公開することができます。公開はメソッドのプロパティで行います。



4D Webサーバは、サービスの管理、ならびにWSDLファイルの公開や保守を自動的に処理します。XMLによるリクエスト内容の解析やパラメータのフォーマット、結果の送信等は4th Dimensionが実行し、特定のプログラムは必要ありません。

4DによるSOAPリクエストの処理

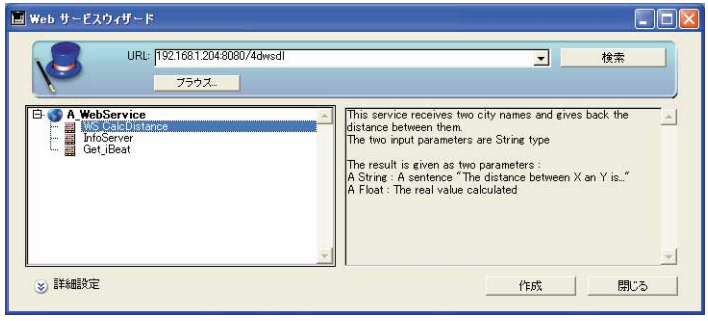


ただし、リクエスト処理をカスタマイズしたい場合に、特定の4th Dimensionランゲージコマンドを使用することができます（後述の「Webサービス（サーバ）」の節を参照）。

注：“従来型の”HTTPリクエストと同じメカニズムを使用することにより、SOAPリクエストの安全性が保証されています（後述の「Webサービスのセキュリティ」を参照）。

Web サービスクライアントとしての 4D

インターネットや使用しているネットワーク上で提供されるあらゆるタイプのWebサービスをデータベースから利用することができます。大抵の場合、クライアントの「Webサービスウィザード」を使用すると、最小のプログラム作業だけであらゆるWebサービスを即座に利用することができます。



4Dでは、ネットワーク経由でパラメータを送信し、応答を取得することによりWebサービスを使用します。これらの処理は、“プロキシ”メソッドが担当します。Webサービスを呼びだすプロキシメソッドの作成は完全に自動化され、プログラムの必要なく実行されます。つまり、作成したコードからこれらのメソッドを呼びだすだけです。

サーバ側でもこれと同じ方法で、4th Dimensionのランゲージコマンドを使用して、これらのメソッドをカスタマイズすることができます。

Web サービスのセキュリティ

4Dが公開するWebサービスは、4D Webサーバに対して設定されたセキュリティメカニズムを継承します。この結果、Webサービスリクエストは、パスワード、「On Web Authentication」と「On Web Connection」データベースメソッド、SSLプロトコルの使用など、従来のWebリクエストと同じ設定を利用することができます。

さらに、特定の設定（例えば、**Get SOAP info**コマンドや**Is SOAP request**コマンド）を用いて、Webサービス公開に関する詳細な制御を行うこともできます。

クライアント側では、Webサービスサーバへの接続はSSLを使用した暗号化モードで実行することができます。ここでも特定のコマンドを用いて、認証を必要とするサーバへの接続を行うことができます。

RPC、DOC、複合型の互換性

Webサービスのコミュニケーションレイヤ（転送、サービスのコール、交換セキュリティを保証）は、RPC（Remote Procedure Call）モードとDOC（Message/Document）モードという2種類のモードで運用することができます。この2つのモードの違いは、サーバとクライアントに対するリクエストと応答を組み立てるレベルによるものです。

これらのモードと4th Dimensionとの互換性は、次の通りです。

■ サーバ側では、4th Dimension 2003はRPCモードだけでWebサービスを公開します。

■ クライアント側では、4th Dimension 2003はRPCモードとDOCモードの両方をサポートします。

SOAPプロトコルを介して、単純型（simple types）と複合型（complex types）という異なるタイプのXMLデータのやり取りが行われます。RPCモードで公開されたWebサービスのデータは、いずれのタイプでも構いません。これとは逆に、DOCモードで公開されたWebサービスのデータは、意図的に複合型となります。

■ サーバ側において、4th Dimension 2003は、データ配列を除き、単純データタイプだけを公開します。

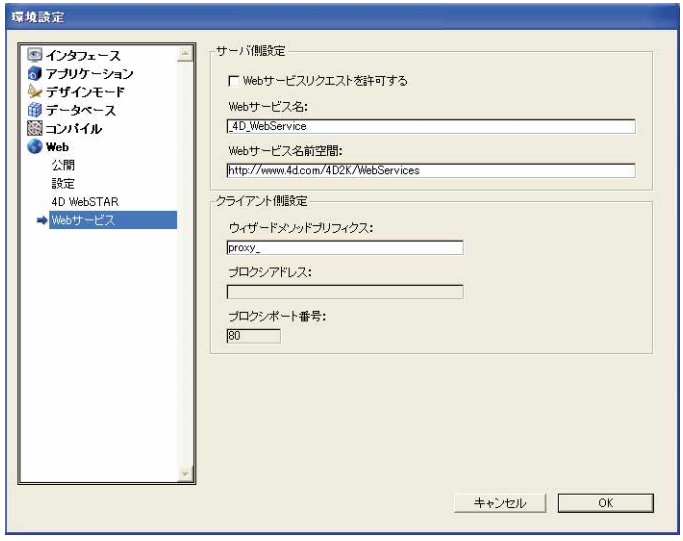
■ クライアント側において、4th Dimension 2003は、単純データタイプ（RPCモード）を用いたWebサービスと複合データタイプ（RPCまたはDOCモード）を用いたWebサービスを共にサポートします。

配列[1]を除き、4Dデータベースでは複合XMLデータタイプを直接使用することができず、特別な処理が必要になります。この追加処理は、一部クライアントのWebサービスウィザードによって対処されますが、プログラミングもまた必要になります。この件に関する詳細は、後述の「複合型の処理」の節を参照してください。

環境設定

データベースの「環境設定（「Web」テーマ）」の「Webサービス」ページを使用して、Webサービスの公開や利用に関連する全般的なパラメータを定義することができます。

1. データ配列は複合XMLデータタイプですが、4Dはそのように認識せずに自動的に処理します。



4th Dimension を使用した Web サービスの公開

4th Dimension での Web サービスの公開は、通常 3 つの段階を経て実行されます。

1 公開されるメソッドの作成

2 公開に関する設定（WSDL）

3 公開

さらに、カスタマイズを行う段階を定義することができますが、これは必須ではありません。

Web サービスメソッドの作成

Web サービスとして公開するために、あらゆるタイプのプロジェクトメソッドを作成することができます。このメソッドは、引数を受け取り、結果を返さなくてはなりません。また、これらの引数は、「コンパイラ」テーマの 4D コマンドを使用して、メソッドヘッダで必ず宣言しなくてはなりません。

デフォルトでは、Web サービスとして公開された時に、4th Dimension はメソッドの処理で必要となる引数をフォーマットします。しかし、後述する **SOAP DECLARATION** コマンドを使用して、これらの引数を変更することができます。

4th Dimension は、SOAP 経由で送受信したデータの符号化と解読を自動的に処理します。

警告：SOAP リクエストでは、メソッド名が XML タグとして使用されます。タグ名に関する XML 規格に準拠し、Web サービスとして公開されるメソッド名にはスペースや拡張文字を使用しないでください。次の Latin 文字だけを使用することができます： ([A-Za-z0-9._] ¦ '-')*

Webサービスとして公開されるメソッド開発の定義や監視を行うには、「Webサービス」テーマ内のコマンドを使用しなくてはなりません。これらのコマンドは、後述の「Webサービス（サーバ）」の節で説明しています。

メソッドの公開

データベースの1つ以上のメソッドをWebサービスとして公開するには、次の3つの条件を満たしていなくてはなりません。

注：Webサービスを公開するには、SOAPサーバ（4D Server、4D Client、またはシングルユーザの4th Dimension）として使用するマシンに4D Webサービスのライセンスが登録されている必要があります。

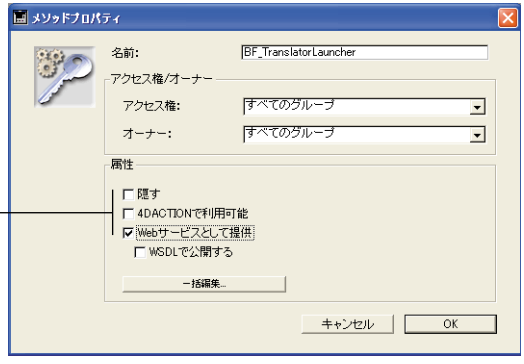
■ 4D Webサーバが起動していなければはなりません（これを調べるには、「ユーザ」モードの「Webサービス」メニューをプルダウンします。起動していれば、「Webサービスの開始」コマンドが非アクティブになっています）。

■ アプリケーションの「環境設定」ダイアログボックスの「Webサービス」ページにある「Webサービスリクエストを許可する」オプションが選択されていなければはなりません。未選択の場合、4th Dimension は SOAP リクエストを拒否し、WSDL を生成しません。

Webサービス公開オプション

このオプションが選択されていると、4th Dimension は WSDL ファイル（後述する「WSDL ファイルの生成」の節を参照）を作成します。

■ 公開される各メソッドには、「Webサービスとして提供」属性が設定されていなくてはなりません。この設定は、「メソッドプロパティ」ウインドウにあるオプションを使用して行います。



「Webサービスとして提供」オプション

注：4th Dimension 2003において、「メソッドプロパティ」ウインドウを表示する手順は次の通りです。

・「メソッド」メニュー、または「メソッド」エディタのコンテキストメニューから「メソッドプロパティ...」を選択する。

・4Dエクスプローラの「メソッド」ページのコンテキストメニューから「メソッドプロパティ...」を選択する。

メソッドに対して「Webサービスとして提供」オプションが選択されている場合、SOAPリクエストを介し、そのメソッドをWebサービスとして呼び出すことができます。

注：「WSDLで公開する」オプションも同時に選択されている場合、そのメソッドはサーバのWSDLに表示されます（次の節を参照）。

4Dエクスプローラでは、特殊なアイコンを使用してWebサービスとして提供されるメソッドを表わします（WSDLファイルで公開されるメソッドも同様）。

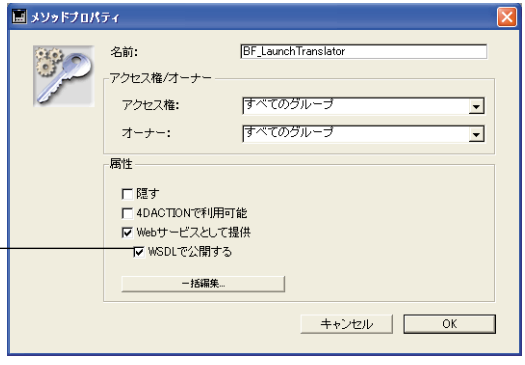
Webサービスとして提供されるメソッド

Webサービスとして提供され、WSDLファイルで公開されるメソッド

WSDL ファイルの生成

4th Dimension において、WSDL は単独の Web サービスに相当します。これはメソッドとその引数を定義し、特定の場所で参照されます。4D では、WSDL は実在上の“ファイル”ではなく（メモリ上だけに存在し、ディスク上には記録されていない）、常に「4DWSDL」という名前で Web サーバのルートに位置する URL です。例えば、使用している Web サービスのアドレスが http://www.myserver.com である場合に、その WSDL は http://www.myserver.com/4DWSDL という URL で参照することができます。

Web サービスリクエストが許可されると、「WSDL で公開する」オプションが少なくとも 1 つのメソッドに対して選択されている場合に、4th Dimension は 4D Web サーバの WSDL を動的に自動生成します。



WSDL ファイルで公開するオプション

デフォルトでは、このオプションが選択されていません。

このドキュメントには、構文ならびに 4D メソッドのコールに必要な情報（メソッド名、URL，引数等）が XML 言語で記述されています。

Web ブラウザに表示される WSDL ファイルの例

```
<?xml version="1.0" encoding="UTF-8" ?>
<!--  this WSDL file was automatically generated by 4D  -->
- <definitions name="FourD_WebService" targetNamespace="http://www.4d.com/namespace/default"
    xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
    xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:tns="http://www.4d.com/namespace/default"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
    instance">
  - <message name="EuroConverterRequest">
      <part name="currency_In" type="xsd:float" />
      <part name="Money_In" type="xsd:string" />
      <part name="Money_Out" type="xsd:string" />
    </message>
  - <message name="EuroConverterResponse">
      <part name="currency_out" type="xsd:float" />
    </message>
  - <portType name="FourD_WebServicePortType">
    - <operation name="EuroConverter">
        <input message="tns:EuroConverterRequest" />
        <output message="tns:EuroConverterResponse" />
      </operation>
    </portType>
  - <binding name="FourD_WebServiceBinding" type="tns:FourD_WebServicePortType">
```

WSDLへメソッドを追加したり、または削除する方法は、「メソッドプロパティ」ウインドウの対応するオプションを選択、または選択解除するだけです。

4th Dimensionは、即座にWSDLの内容を更新します。

注：4Dエクスプローラでは、特殊なアイコンを用いて、WSDLで公開されるメソッドを区別しています（前述の節を参照）。

Webサービス名のカスタマイズ

インターネット上に公開される各Webサービスには名前が付けられます。この名前を使用して、SOAPサーバレベル（サーバで複数のWebサービスが公開されている場合）およびWebサービスディレクトリの両方で各サービスを区別することができます。

デフォルトでは、4th Dimensionは“A_WebService”という名前を使用します。このパラメータは、データベースの「環境設定」において「Web」テーマの「Webサービス」ページで変更することができます。

警告：タグ名に関するXML規格に準拠し、文字列にはスペースや拡張文字を使用しないでください。次のLatin文字だけを使用することができます：([A-Za-z0-9._]¦'-')*

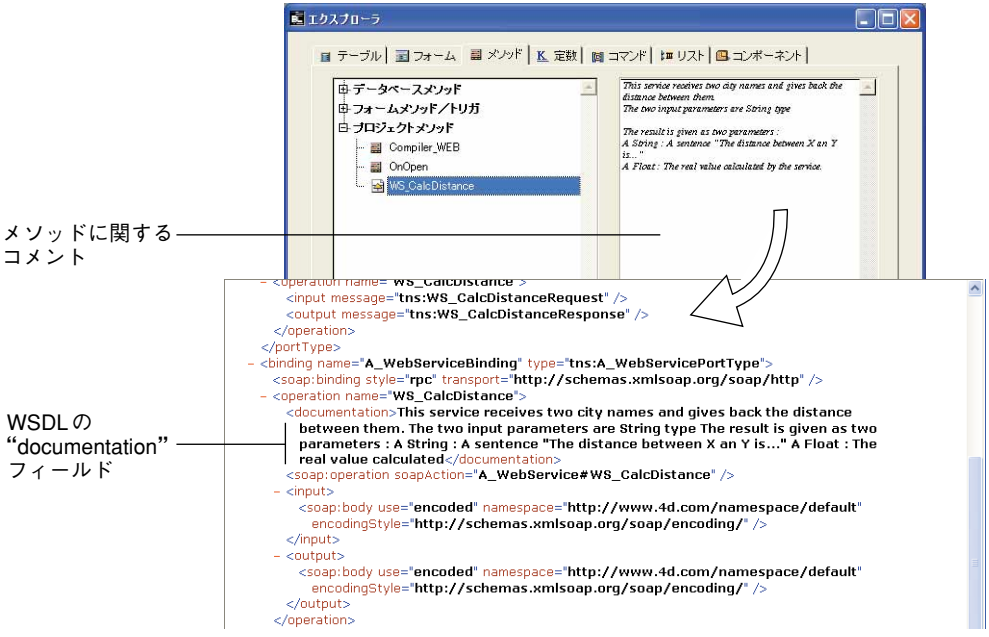
名前空間（Namespace）のカスタマイズ

インターネット上に公開されるWebサービスは、それぞれがユニークでなければなりません。Webサービス名のこの一意性は、XMLの名前空間を使用することによって保証されます。名前空間は任意の文字列であり、独自の方法で一連のXMLタグを識別するために使用されます。一般的に、名前空間はその会社のURLで始まります（http://mycompany.com/mynamespace）。この場合、特別なURLを指定しなければならないという訳ではなく、使用する文字列が一意であることが重要になります。

デフォルトとして、4th Dimensionは http://www.4d.com/namespace/default という名前空間を使用します。このパラメータは、データベースの「環境設定」において「Web」テーマの「Webサービス」ページで変更することができます。

公開メソッドへのコメントの追加

Webサービスとして提供され、WSDLで公開されるメソッドに関連付けたコメントは、“documentation”フィールドとして自動的にWSDLファイル内に現われます。



このメカニズムは、公開されるメソッドの説明や証明を行うために使用されています。このフィールドの解釈や処理方法は、クライアントのWebサービスの実装によって決まります。

4D により公開された Web サービスへのアクセス

4DによりWebサービスが公開されると、Webサービスをサポートする任意のクライアントアプリケーションからWebサービスに接続することができます。アクセスモードならびにWebサービスのサーバを使用して交換する情報の処理方法は、その処理に使用するクライアントアプリケーション側で決定されます。

4D Webサービスを使用する上で必要となる情報はすべて（サービスのURL、使用するパラメータ等）、4DのWSDLで公開されます。原則として、Webサービスを使用する際は、この情報を取得するために、まずSOAPサーバのWSDLを読み込むことから始めてください。

4Dでは、WSDLのURLは常にhttp://サーバアドレス/4DWSDLです。

ただし、このステップは必須ではありません。SOAPサーバへの接続はダイレクトに行うことができます。

SOAPリクエストの作成、ならびにそのメソッド定義に必要となる値の一覧を次に示します。

■ 4Dによって公開されたWebサービスにアクセスするURL

http://サーバアドレス/4DSOAP/（カスタマイズ不可）

■ Webサービス名

デフォルト：A_WebService

カスタマイズ可能な値（前述の「Webサービス名のカスタマイズ」の節を参照）

■ 公開メソッド名

開発者が定義した4Dプロジェクトメソッド名（前述の「Webサービスメソッドの作成」の節を参照）

■ メソッドの引数

メソッドには引数が宣言されていなくてはならない（開発者が定義）。

デフォルトのSOAP名：FourD_arg0, FourD_arg1... FourD_argn

カスタマイズ可能な値（後述の**SOAP DECLARATION**コマンドを参照）

■ 名前空間

デフォルト：http://www.4d.com/namespace/default

カスタマイズ可能な値（前述の「名前空間のカスタマイズ」の節を参照）

■「SOAP Action」フィールドの内容

サービス名#メソッド名 （カスタマイズ不可）

4th Dimension における Web サービスへのサブスクライブ

4th Dimension 2003 を使用して、Web サービスへのサブスクライブ、つまり独自の 4D データベース内で Web サービスを呼び出し、使用することができます。

インターネット上で利用可能な Web サービスを使用すると、株価情報へのアクセスや貨物配送の追跡、複雑な計算の実行など、あらゆるタイプの機能を容易にデータベースへ追加することができます。インターネット上で利用できる数多くの Web サービスが、ほとんどすべての要求を満たしてくれます。

また、自分で公開した Web サービスに別のデータベースからサブスクライブすることも可能であり、この方法によりさまざまな 4D データベースが各々の間で通信できるようになります。

方法

あらゆる 4D 2003 データベースは、インターネットに接続するだけで Web サービスにサブスクライブすることができます。

通常、Web サービスを呼び出すためには、次に説明する手順に従ってください。

1 サブスクライブしようとする Web サービスの URL を取得する。

URL を取得するには、インターネット上に公開された Web サービスを登録している Web サイトや（例えば、www.xmethods.net）、UDDI のようなディレクトリを利用することができます。大抵の場合、Web サービスに関する WSDL ファイルの URL を取得しなければなりません。

注：4th Dimension は、RPC モードや DOC モードで公開された Web サービスを使用することができます（前述の「RPC、DOC、複合型の互換性」の節を参照）。

2「Web サービスウィザード」を使用して、利用する Web サービスの WSDL の内容を解析し、対応する「プロキシメソッド」を生成する。

プロキシメソッドはローカルなプロジェクトメソッドで、Web サービスへの問い合せや戻り値の取得を行います。この手順については、後述の「Web サービスウィザードの使用」の節を参照してください。

注：

・WSDL ファイルを使用しなくても、「Web サービスウィザード」でプロキシメソッドを作成することができます（使用するパラメータを手動で入力するだけです）。

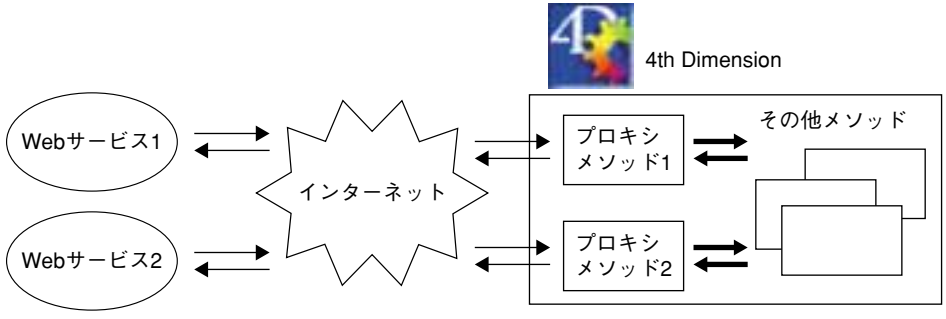
・また、「Web サービスウィザード」を使用しなくても、「メソッド」エディタでプロキシメソッドを作成することもできます（上級ユーザ向け）。

3 4Dデータベースのコードにおいて、必要になるたびに適切な引数を渡してプロキシメソッドをコールする。

この手順については、後述の「プロキシメソッドの呼び出し」の節で説明します。

プロキシメソッドはWebサービスへの接続を処理します。

クライアントWebサービスとしての4Dの処理に関する原理



Webサービスウィザードの使用

4th Dimension 2003アプリケーションからのWebサービスへのサブスクライブは、「Webサービスウィザード」がすべて処理します。このウィザードは以下の処理を自動的に実行します。

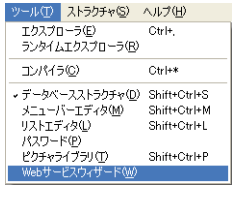
■ 利用するWebサービスのWSDLファイルの解析

■ 作成するプロキシメソッドのパラメータ定義

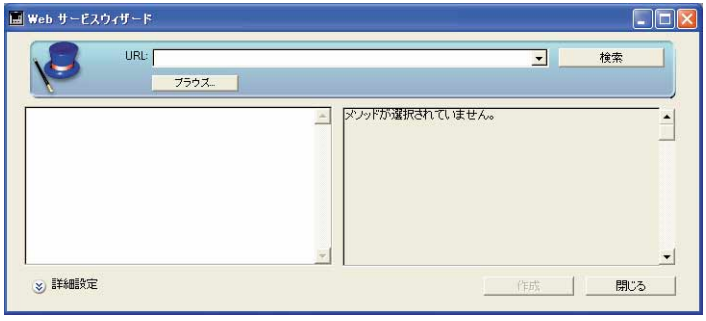
■ プロキシメソッドの作成

ウィザードウインドウ

「Webサービスウィザード」のウインドウを開くには、4th Dimensionの「ツール」メニューから「Webサービスウィザード」を選択します（「デザイン」モード）。



すると、ウィザードウインドウが表示されます。



このウインドウには3つのエリアがあります。

■「URL」エリアでは、選択したWebサービス用のWSDLファイルのURLを入力したり、あるいは選択することができます。このエリアはコンボボックスで、以前に入力された値がドロップダウンリスト形式で保存されています。

■ 中央のエリアには、サービス名や公開メソッドなど、WSDLファイルの内容を解析した結果が表示されます。

■ 下側のエリア（“上級”パラメータ、デフォルトでは非表示）には、中央のエリアで選択したメソッドのパラメータが表示されます。

「検索」ボタンにより、指定したWSDLファイルの解析が開始され、情報エリアに結果が表示されます。

「ブラウズ...」ボタンにより、標準のファイルオープンダイアログボックスが表示され、ローカル上に保存されたWSDLファイルを選択することができます。すると、「URL」エリアには“file://”で始まるファイルのアクセスパスが表示されます（このエリアにアクセスパスを手動で入力することもできます）。

「作成」ボタンを使用すると、選択したWebサービスに対応するプロキシメソッドが生成されます。

「閉じる」ボタンを使用すると、「Webサービスウィザード」ダイアログボックスが再度閉じられます。

WSDLの解析とプロキシメソッドの作成（標準モード）

Webサービスウィザードの主な使用目的は、WSDLファイルを解析し、対応するプロキシメソッドを生成することです。この標準的な処理は完全に自動化されているため、ユーザ側のプログラミングや特別な知識は必要ありません。

▼ WSDLファイルを解析し、プロキシメソッドを生成するには、次の手順に従ってください。

1「URL」エリアに、利用しようとするWebサービスのWSDLファイルのURLを入力するか、ペーストする。

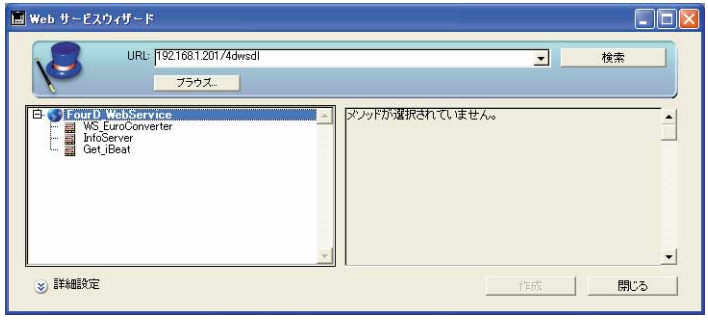
例えばこのURLは、Webサービスの“ディレクトリ”から取得したものや、Webサービスサーバで直接通信したものなどです。

また、ローカルなURL、つまりハードディスク上に保存されたWSDLファイルのアドレスを指定することもできます。これを行うには、「ブラウズ...」ボタンをクリックしてローカル上のWSDLファイルを選択するか、または直接「URL:」エリアにファイルのアクセスパスを入力します。ローカルファイルのアクセスパスは“file://”で始まり、その後に標準のシステムフォルダの区切り文字を使用します。必ず絶対パスを渡さなければなりません。

注：この例題では、4th Dimensionにより公開されたWebサービスを使用しますが、任意のタイプのWebサービスを選択しても構いません。

2「検索」ボタンをクリックして、4th DimensionにWSDLファイルの内容解析を行わせる。

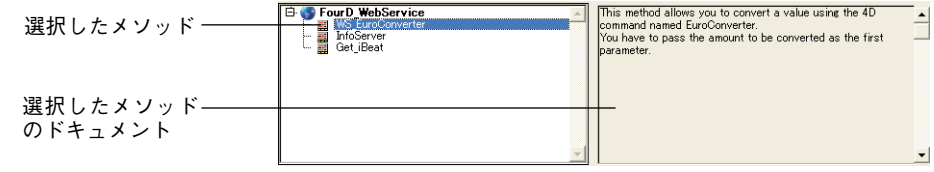
しばらくすると、中央のエリアにファイルの解析結果が表示されます。階層リスト形式で、Webサービスの名前ならびに公開されたメソッドが示されます。



注：Shiftキーを押しながら「検索」ボタンをクリックすると、WSDLファイルのXMLソースコードをデフォルトのWebブラウザに直接表示することができます。

Webサービス名をクリックすると、ウインドウの右側にドキュメント（存在する場合）が表示されます。ドキュメントがない場合には、“ドキュメントがありません”という表示が現われます。

同様に、メソッド名を選択すると、各メソッドのドキュメント（存在する場合）が表示されます。



注：WSDL ファイルの解析の結果、複合型のパラメータが存在した場合には、ウィザードは問題となるメソッドの横に黄色の旗を表示します。この場合、Web サービスをデータベースに組み込むには、ユーザ側のプログラミングと XML 言語の知識が必要になります。詳細については、後述の「複合型の処理」の節を参照してください。

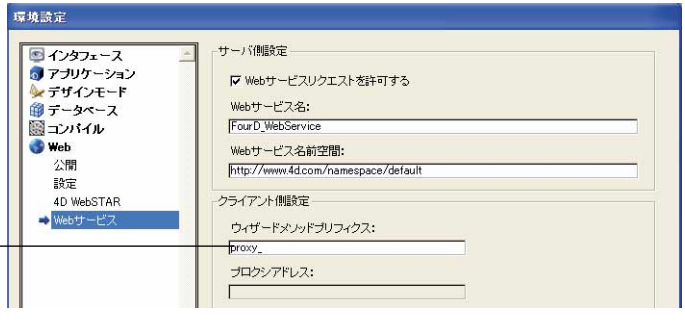
PieChart

ただし、複合型が存在しても、ウィザードは対応するプロキシメソッドを生成します。

3 利用したい Web サービスメソッドを選択し、「作成」ボタンをクリックする。

すると、即座に 4th Dimension は対応するプロキシメソッドを生成し、4D の「メソッド」エディタのウインドウ上に表示します。

プロキシメソッドには、デフォルトの接頭辞“proxy_”と Web サービスメソッド名を連結した名前が指定されます。デフォルトの接頭辞は、データベースの「環境設定」の「Web サービス」ページで変更することができます。



また、作成後にプロキシメソッドの名前を変更することもできます。名前を変更しても、メソッドの動作には影響を与えません。

注：外部のWebサービスの使用を管理する各種4Dランゲージコマンドに関しては、後述の「ランゲージ」の章で説明します。

上級パラメータの使用

WSDLファイルの解析に基づき「Webサービスウィザード」で生成されたプロキシメソッドは、即座に操作可能であり、そのままの状態で使用できます（標準モード）。

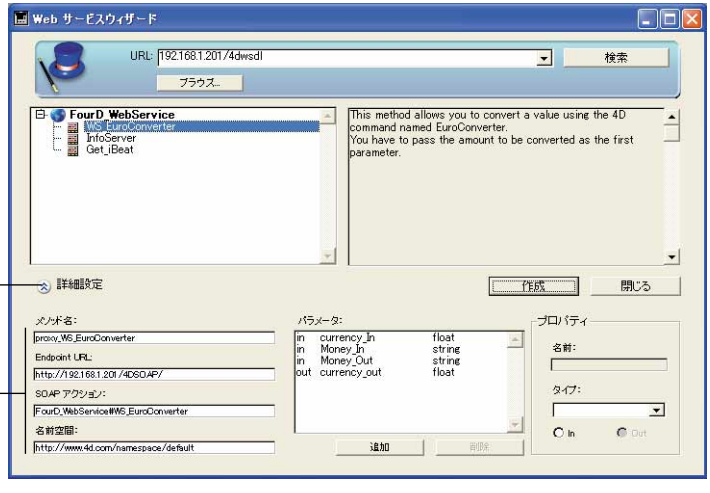
しかし、WSDL解析の結果生じたパラメータを変更したい場合があります。例えば、プロキシメソッド名を変更することができます。

また、「Webサービスウィザード」を使用して、自分で入力したパラメータ用のプロキシメソッドを作成することもできます。この場合は、WSDLパーサーを使用しないでください。

メソッドを作成するために、すべてのパラメータを入力する必要はありません。

プロキシメソッドの“テンプレート”を作成するためには、パラメータをひとつも入力しなくても構いません。後で4th Dimensionのプログラムを使用して、このテンプレートにパラメータを設定することができます。

これらの非標準モードにおいては、「Webサービスウィザード」の上級パラメータを使用しなければなりません。これらのパラメータを表示するには、ウィザードウインドウの左下にある展開用のボタンをクリックします。クリックすると、上級パラメータフィールドが現われます。メソッドを選択している場合には、これらのフィールドには現在のパラメータが表示されます。



展開用のボタン

上級パラメータ

これらパラメータはすべて変更可能です。ただし、WSDL解析で生成されたパラメータを変更する場合には（パラメータ名を除く）、その結果としてWebサービスの動作が変わる可能性があるため、注意深く行わねばなりません。

次に上級パラメータについて説明します。

■ **メソッド名：**作成されるプロキシメソッドに対してウィザードが割り当てる名前。デフォルトでは、接頭辞“proxy_”（4Dアプリケーションの「環境設定」で変更可）の後に選択したメソッド名を連結した名前になります。この名前は自由に変更することができ（例えば、データベースに同じ名前が既に存在する場合など）、Webサービスの動作には影響を及ぼしません。

■ **エンドポイントURL：**プロキシメソッドがSOAPリクエストを送信するURL。

■ **SOAPアクション：**「SOAPAction」フィールドの内容。通常、このフィールドには“サービス名#メソッド名”という値が納められます。

■ **名前空間：**Webサービスの名前空間（詳細については、前述の「名前空間のカスタマイズ」を参照）。

■ **パラメータテーブル：**この表には、公開メソッドのパラメータ一覧が表示されます。

表の各行にパラメータが記述されます。

- 最初のカラムは、そのパラメータが入力タイプ（“in”）か出力タイプ（“out”）かを示します。この特性は、公開メソッド側ではなくプロキシメソッド側の視点から判断されます。
- 2番目のカラムは、パラメータ名を表わします。
- 3番目のカラムは、パラメータのSOAPタイプを示します。4Dが受け入れる各種SOAPタイプについては、「プロパティ」エリアの「タイプ」メニューで表示することができます。プロキシメソッド内において、SOAPタイプを対応する4Dタイプに関連付ける処理は、4th Dimensionの「Webサービスウィザード」が担当します。

次の表は、受け入れられるSOAPタイプの値と、それに対応する4Dタイプを示しています。

SOAP タイプ	対応する 4D タイプ
boolean	ブール
int	倍長整数
time	時間
float	実数
double	実数
date	日付
string	テキスト
base64Binary	BLOB
ArrayOfBoolean	ブール配列
ArrayOfInt	倍長整数配列
ArrayOfTime	倍長整数配列
ArrayOfFloat	実数配列
ArrayOfDate	日付配列
ArrayOfString	テキスト配列
AsXML [1]	BLOB

「プロパティ」エリアには、表より選択したパラメータの特性が表示されます。4D の「Web サービスウィザード」を使用すると、例えば、指定した WSDL ファイルが最新ではないなどの場合に、既存のパラメータの変更や新規パラメータの追加を行うことができます。

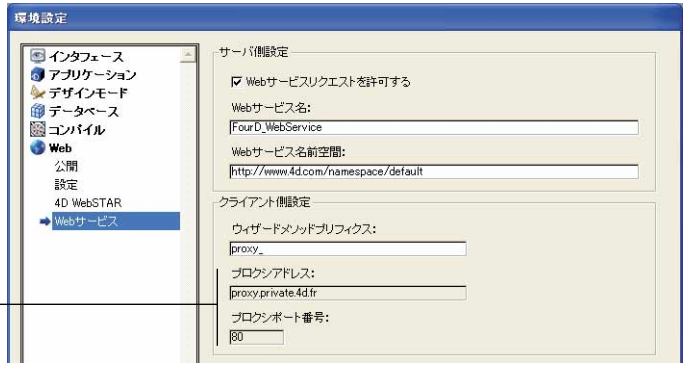
- パラメータを変更するには、パラメータの選択後、「プロパティ」エリアで変更を行います。
- パラメータを追加するには、「追加」ボタンをクリックし、「プロパティ」エリアでその特性を定義します。
- パラメータを削除するには、リストからパラメータを選択し、「削除」ボタンをクリックします。

注：「作成」ボタンをクリックしてプロキシメソッドが実際に作成された場合にのみ、上級パラメータの変更が反映されます。

1. 厳密に言うと、AsXML タイプは SOAP タイプではなく、複合型の XML をサポートするために使用されます（後述の「複合型の処理」の節を参照）。

接続パラメータの表示

Webサービスへサブスクライブする際、特にプロキシサーバの場合に、4th Dimensionはそのマシンに設定された現在のインターネット接続用パラメータを使用します。これらのパラメータは、データベースの「環境設定」において「Web」テーマの「Webサービス」ページで調べることができます。

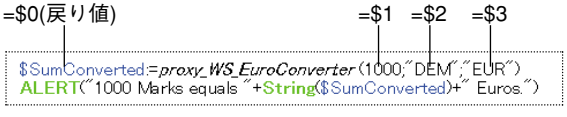


4Dだけがこれらの値を読み込みます。これらの値を変更したい場合、そのマシンのインターネット用パラメータを使用して行わなければなりません。

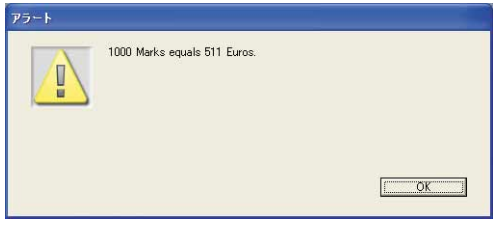
プロキシメソッドの呼び出し

コード内でプロキシメソッドを呼び出すには、プロキシメソッド名を記述して必要な引数を渡します。これらの引数は、「Webサービスウィザード」によりプロキシメソッドのヘッダエリアに宣言されています。4Dのメソッド間でのパラメータ受け渡しに関する標準的なシンタックスに従い、引数には$0、$1、$2などの名前が付けられます。これら引数は公開メソッドに関する上級パラメータの説明エリアに表示され（前述の「上級パラメータの使用」の節を参照）、またこのマニュアルの各所でも説明しています。

例えば、前述の例で使用したメソッド（WS_EuroConverter）を再度見てみると、プロキシメソッドは次の方法で呼びだすことができます。



メソッドを実行すると、次の警告が表示されます。



複合型の処理

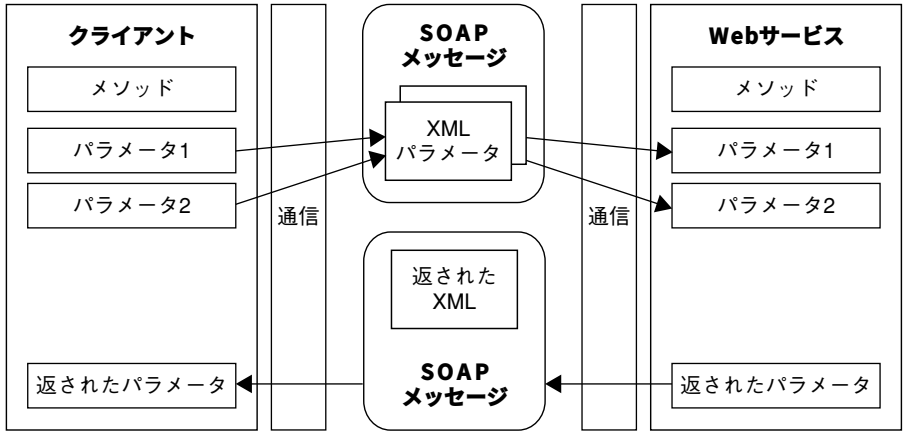
4th Dimensionでは、RPCモードやDOCモード、ならびに複合型で公開されたWebサービスを使用することができます（前述の「RPC、DOC、複合型の互換性」の節を参照）。

注：実際には複合XMLタイプであるにも関わらず、4Dはデータ配列を単純型として処理します。

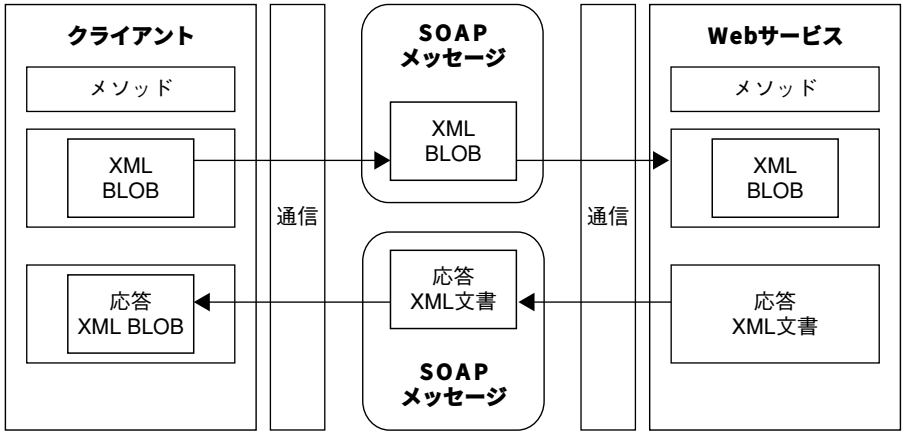
「Webサービスウィザード」によって生成された、複合型を含むプロキシメソッド（つまり複合型を伴うRPCモードで公開、またはDOCモードで公開されたメソッド）は、標準的なプロキシメソッドと似ています。しかし、これらのWebサービスを使用する場合、**CALL WEB SERVICE**コマンドでは引数として“manual”という語句を含む定数が使われていることがわかります

実際には、このようなWebサービスを使用する際には、4D開発者による追加処理が必要になります。この主な理由は、複合型のやり取りが文書やXML要素の形で行われるためです。つまり、これらのSOAPパラメータから情報の取得や組み込みを行うには、事前にXMLを解析する必要があります。これに対して単純型の場合は、パラメータの値を直接読み取ることができます。

単独タイプ（RPC）の使用



複合型（DOC）の使用



4th Dimensionでは、複合型のパラメータ（配列を除く）はBLOB形式で処理されます。これを取り扱うためには、4Dランゲージを使用しなければなりません。

複合入力タイプと出力タイプ

RPCモードで公開されたWebサービスは、単純／複合型と入力／出力タイプのパラメータを組み合わせたさまざまな設定で利用することができます。例えば、あるWebサービスが、単純型のパラメータを含むリクエストを受け入れ、その結果を複合型の形でクライアント側に返すことができます。

これとは逆に、DOCモードで公開されたWebサービスは、複合型のパラメータのみを処理することができます。

複合型の処理は、SOAPパラメータが入力タイプか出力タイプかによって異なります。

■ 入力パラメータの場合（4th Dimensionによって受信される）、XML構造の解析と有用な情報の取り出しは、4DのXMLコマンドを使用して実行されます（後述の「XML」の節を参照）。

■ 出力パラメータの場合（4th Dimensionから送信される）、サーバから要求された情報に従って、Webサービスへ送信するXML構造を“手動で”作成するかどうかの判断は4D開発者に委ねられます。

この操作に関する詳細（ならびに入力パラメータの解析）に関しては、4th Dimensionから提供されるサンプルデータベースのコードを参照されることをお勧めします。複合型のパラメータの送信／受信に関する詳細は、後述の**CALL WEB SERVICE**コマンドの説明を参照してください。

4Dバージョン2003のランゲージコマンドには、さまざまな変更が加えられています。

■ 新しいテーマ「Webサービス（サーバ）」と「Webサービス（クライアント）」には、4th DimensionにおけるWebサービスの管理と監視のためのコマンドが集められています。

■ 新しいテーマ「XML」には、XMLフォーマットのデータ解析用コマンドが集められています。

■「印刷」テーマには新しいコマンド（後述の「新しい印刷コマンド」の節を参照）が追加され、機能が強化されました。また、いくつかの既存コマンドの機能が拡張されました。これらは、**SET PRINT MARKER**コマンドに関する変更、ならびに**PAGE BREAK**コマンドと**CANCEL**コマンド（「入力制御」テーマ）を**Print form**関数と一緒に使用する場合に関する変更です。

■ その他にも、さまざまな新しいコマンドが追加されています：**MULTI SORT ARRAY**（「配列」テーマ）、**BEST OBJECT SIZE**（「オブジェクトプロパティ」テーマ）、**Is data file locked**（「4D環境」テーマ）。

■ 既存の各種コマンドと定数も変更されています：**MENU BAR**（「メニュー」テーマ）、**SET DATABASE PARAMETER**（「ストラクチャアクセス」テーマ）。

■ 最後に、4D Chartのグラフィックエディタに関する「CT Area Control」テーマに2つの新しいコマンドが追加されました。

注：新しいテーマ「クイックレポート」には、新しくなった「クイックレポート」エディタを管理するコマンドが集められています。この新しいエディタに関しては、別のドキュメントで説明しています。

Webサービス（サーバ）

4th Dimension 2003はWebサービスをサポートします。つまり、プログラムを使用して、作成したデータベースから直接Webサービスの公開（サーバ側）や利用（クライアント側）を行うことができます。

この節では、4th DimensionにおいてWebサービス（サーバ側）の公開を行うために使用するコマンドについて説明します。Webサービスへのサブスクライブを行うために使用するコマンドの説明は、後述の「Webサービス（クライアント）」の節を参照してください。

4th Dimensionを使用したWebサービスの公開は、メソッドプロパティのオプションを利用して簡単に実行することができます（4th Dimension 2003におけるWebサービスのセットアップに関する詳細は、前述の「Webサービス」の章を参照してください）。大抵の場合、この操作だけでWebサービスの公開を行うことができます。ただし、データ配列を使用する場合など、特定の仕組みをカスタマイズしたい場合には、4th Dimension 2003のサーバSOAPコマンドを使用しなければなりません。

注：規約により、サーバ側とクライアント側のコマンド（および定数）名を区別するため、“SOAP”と“Webサービス”という用語がそれぞれ使用されています。それ以外の点では、これら2つの概念は同じ技術について触れています。

SEND SOAP FAULT

SEND SOAP FAULT (faultタイプ; 説明)

引数	タイプ		説明
faultタイプ	倍長整数	→	1= クライアント側のfault 2= サーバ側のfault
説明	文字列	→	SOAPクライアントに送信されるエラーの説明

このコマンドは、SOAPクライアントに対してエラーを返し、誤り（fault）の原因（クライアントまたはサーバ）を伝えるために使用します。

このコマンドを使用すると、結果を返さなくてもクライアントにエラーを知らせることができます。

例えば、Webサービス“Square_root”を公開し、クライアントが負数を用いてリクエストを送信した場合、クライアント側の誤りが検出されます。その場合、このコマンドを使用して、正数が必要であることをクライアントに知らせることができます。

例えば、サーバ側の誤りとしては、メソッド実行時に起きるメモリ不足があります。

＜faultタイプ＞には、エラーの原因を渡します。「Webサービス（サーバ）」テーマ内の次の定義済定数を使用することができます。

定数	タイプ	値
SOAP Client Fault	倍長整数	1
SOAP Server Fault	倍長整数	2

＜説明＞には、エラーの説明を渡します。クライアントの実行がこれに適合する場合、エラーが処理されます。

▼ コマンドの説明で取り上げた、Webサービス“Square_root”の例題に戻ると、負数を用いたリクエストの処理には、次のコマンドを使用することができます。

```
SEND SOAP FAULT(Client Fault; "正数が必要です。")
```

SOAP DECLARATION

SOAP DECLARATION (変数; タイプ; 入出力{; エイリアス})

引数	タイプ		説明
変数	4D変数	→	SOAP入力、または出力引数を参照する変数
タイプ	倍長整数	→	引数が示す4Dタイプ
入出力	倍長整数	→	1= SOAP入力 2= SOAP出力
エイリアス	文字列	→	SOAP交換の際にこの引数用に公開された名前

このコマンドを使用して、Webサービスとして公開された4Dメソッドで使用する引数のSOAPタイプを明示的に宣言します。

メソッドをWebサービスとして公開する場合、標準的な引数である$0、$1～$nを使用して、外部に対し（特にWSDLファイル内で）Webサービスのパラメータを表現します。SOAPプロトコルでは引数に明示的に名前が付けられている必要があります。デフォルトとして、4th Dimensionは“FourD_arg0、FourD_arg1～ FourD_argn”という名前を使用します。

しかし、次の理由からデフォルトとしてのこの運用は問題点が多いことが判明しています。

■ $0や$1は配列として宣言することができません。したがって、ポインタを使う必要がありますが、その場合にはWSDLファイルの生成時に値のタイプが分かっていなければなりません。

■ 次に、パラメータ名（入力と出力）のカスタマイズが有効であったり、必要となる場合があります。

■ 最後に、この運用では、RPCコール毎に複数の値を返すことが不可能になります（$0内に）。

SOAP DECLARATIONコマンドを使用すると、これらの制約から解放されます。このコマンドをそれぞれの入力および出力引数に対して実行し、引数の名前やタイプを割り当てることができます。

注：SOAP DECLARATIONコマンドを使用した場合でも、常に「コンパイラ」テーマのコマンドを使用して4D変数を宣言する必要があります。

＜変数＞には、RPCコールで参照される4D変数を渡します。

警告：参照できるのは、プロセス変数または4Dメソッドの引数（$0～ $n）だけです。ローカル変数とインタープロセス変数は使用することができません。

＜タイプ＞には、対応する4Dタイプを渡します。大半の4D変数や配列を使用することができます。「Field and Variable Types」テーマにある次の定義済定数を使用できます。

定数	タイプ	値
Is BLOB	倍長整数	30
Is Boolean	倍長整数	6
Is Integer	倍長整数	8
Is LongInt	倍長整数	9
Is Real	倍長整数	1
Boolean array	倍長整数	22
String array	倍長整数	21
Date array	倍長整数	17
Integer array	倍長整数	15
LongInt array	倍長整数	16
Real array	倍長整数	14
Text array	倍長整数	18
Is Text	倍長整数	2
Is Date	倍長整数	4
Is Time	倍長整数	11
Is String Var	倍長整数	24

注：SOAPメソッドで使用されない定数は次の通りです：Is Alpha Field、Is Pointer、Array 2D、Picture array、Pointer array、Is Picture、Is Subtable、Is Undefined

＜入出力＞には、処理される引数が“入力”タイプ（つまり、メソッドが受け取る値に相当する）か、“出力”タイプ（つまり、メソッドから返される値に相当する）かを示す値を渡します。「Web Services(Server)」テーマにある次の定義済定数を使用することができます。

定数	タイプ	値
SOAP Input	倍長整数	1
SOAP Output	倍長整数	2

注：4D変数（4Dメソッドの引数ではなく）を使用して参照されるSOAP入力の引数は、COMPILER_WEBプロジェクトメソッド内で最初に宣言しなければなりません。詳細は、後述の「COMPILER_WEBメソッドの使用」の節を参照してください。

＜エイリアス＞には、WSDLやSOAP交換で現われるものと同じ引数名を渡します。

警告：RPCコールにおいて、この名前は一意でなくてはなりません（入力引数と出力引数は一緒とみなされる）。そうでない場合、4Dは最後に行われた宣言だけを考慮します。

注：マーカー名に関するXML規格に準拠し、引数名には空白や拡張文字を決して使用しないでください。次のLatin文字だけを使用することができます：([A-Za-z0-9._] ¦ '-')*

引数＜エイリアス＞を省略すると、デフォルトとして4th Dimensionは4Dメソッドの引数（$0 ～ $n）に対して、変数名やFourD_argNという名前を使用します。

SOAP DECLARATIONコマンドは、Webサービスとして公開されたメソッド内に含まれていなくてはなりません。他のメソッドからこのコマンドを呼び出すことはできません。

▼ この例題は引数名を指定します。

```
` WSDLファイルの生成およびSOAPコールの際に、
` fourD_arg1の代わりにzipcodeという文言を使用します。
C_LONGINT($1)
SOAP DECLARATION($1;Is LongInt;SOAP Input;"zipcode")
```

▼ この例題を使用して、倍長整数タイプで郵便番号の配列を取得します。

```
ARRAY LONGINT(codes;0)
SOAP DECLARATION(codes;LongInt array;SOAP Input;"in_codes")
```

▼ この例題を使用して、引数名を指定せずに2つの戻り値を参照します。

```
SOAP DECLARATION(ret1;Is LongInt;SOAP Output)
SOAP DECLARATION(ret2;Is LongInt;SOAP Output)
```

COMPILER_WEB メソッドの使用

SOAP リクエストが受け付けられるたびに、***COMPILER_WEB*** メソッドが呼びだされます（存在する場合）。Web サービスとして公開されるすべてのメソッドに対し、COMPILER_WEB メソッドを使用して、SOAP 入力引数に関連付けた 4D 変数をすべて宣言しなければなりません。実際には、Web サービスメソッドでプロセス変数を使用する場合、メソッド呼び出しの前にそれらの変数が宣言されている必要があります。

デフォルトでは、***COMPILER_WEB*** メソッドが存在していません。したがって、特別に作成しなくてはなりません。

注：4D Web サーバは、POST タイプの“通常の”Web リクエストを受信する際にも、COMPILER_WEB メソッドを呼び出します。

Is SOAP request

Is SOAP request → ブール

引数	タイプ		説明
このコマンドに引数はありません。			
戻り値	ブール	←	SOAP リクエストの場合、True（真） それ以外は False（偽）

このコマンドは、実行されるコードが SOAP リクエストの一部である場合に True を返します。

セキュリティ上の理由から、受信したリクエストの種類を判断するために「On Web Authentication」データベースメソッドでこのコマンドを使用することができます。

Get SOAP info

Get SOAP info (情報番号) → 文字列

引数	タイプ		説明
情報番号	倍長整数	→	取得するSOAP情報 のタイプ番号
戻り値	文字列	←	SOAP情報

このコマンドを使用し、SOAPリクエストに関する各種情報を文字列の形で取得することができます。

SOAPリクエストを処理する際、RPCパラメータの値以外にリクエストに関する追加情報を取得すると便利な場合があります。例えば、セキュリティ上の理由から、このコマンドを「On Web Authentication」データベースメソッドで使用し、要求されたWebサービスメソッドの名前を調べることができます。

＜情報番号＞には、調べようとするSOAP情報タイプの番号を渡します。「Web Services(Server)」テーマにある次の定義済定数を使用することができます。

定数	タイプ	値
SOAP Method Name	倍長整数	1
SOAP Service Name	倍長整数	2

SOAP Method Name：実行しようとしているWebサービスメソッドの名前。

SOAP Service Name：メソッドが属しているWebサービスの名前。

注：4Dへ送信するWebサービスリクエストに対して、最大サイズを設定することができます。この設定は、SET DATABASE PARAMETERコマンド（「ストラクチャアクセス」テーマ）を使用して行います。このコマンドについては後述しています。

Webサービス（クライアント）

この節では、4th Dimensionから外部のWebサービスへサブスクライブする（クライアント側）際に使用するコマンドについて説明します。Webサービスの公開に使用するコマンドの説明は、前述の「Webサービス（サーバ）」の節を参照してください。

4th DimensionをWebサービスへのサブスクライブは、「Webサービスウィザード」を用いて簡単に行うことができます（4th Dimension 2003におけるWebサービスのセットアップに関する詳細は、前述の「Webサービス」の章を参照してください）。大抵の場合、このウィザードだけでWebサービスを利用することができます。しかし、特定の仕組みをカスタマイズしたい場合には、4th Dimension 2003のクライアント SOAPコマンドを使用しなければなりません。

注：規約により、サーバ側とクライアント側のコマンド（および定数）名を区別するため、“SOAP”と“Webサービス”という用語がそれぞれ使用されています。それ以外の点では、これら2つの概念は同じ技術について触れています。

SET WEB SERVICE PARAMETER

SET WEB SERVICE PARAMETER (名前; 値{; soapタイプ})

引数	タイプ		説明
名前	文字列	→	SOAPリクエストに含めるパラメータ名
値	変数	→	パラメータの値を納めた4D変数
soapタイプ	文字列	→	パラメータのSOAPタイプ

このコマンドにより、クライアントSOAPリクエストで使用するパラメータを定義することができます。リクエスト内の各パラメータに対して、このコマンドを呼び出してください（このコマンドを呼び出す回数は、パラメータの数によって決まります）。

＜名前＞には、SOAPリクエスト中に出現するものと同じパラメータ名を渡します。

＜値＞には、パラメータの値を含む4D変数を渡します。プロキシメソッドの場合、通常この変数は、$1、$2、$3等であり、コールされた時にプロキシメソッドへ渡された4D変数に対応します。ただし、中間変数を使用することができます。

注：使用する各4D変数や配列は、「コンパイラ」および「配列」テーマのコマンドを使用して、最初に定義しておかなくてはなりません。

デフォルトとして、4th Dimensionは＜値＞の内容に応じ、引数＜名前＞に対して最も適したSOAPタイプを自動的に決定します。このタイプ指定はリクエスト中に組み込まれます。しかし、パラメータのSOAPタイプを“強制的に”定義したい場合があります。この場合には、以下の文字列のいずれかを使用して、任意の引数である＜soapタイプ＞を渡すことができます（プライマリーデータタイプ）。

SOAPタイプ	説明
string	文字列
int	倍長整数
boolean	ブール
float	32ビットの実数
decimal	小数点付き実数
double	64ビットの実数
duration	年、月、日、時、分、秒単位での期間。 例：P1Y2M3DT10H30M
datetime	ISO8601フォーマットの日付と時間。 例：2003-05-31T13:20:00
time	時間。例：13:20:00
date	日付。例：2003-05-31
gyearmonth	年と月（太陽暦）。例：2003-05
gyear	年（太陽暦）。例：2003
gmonthday	月と日（太陽暦）。例：--05-31
gday	日（太陽暦）。例：---31
gmonth	月（太陽暦）。例：--10--
hexbinary	16進数で表わした値
base64binary	BLOB
anyuri	Uniform Resource Identifier(URI)。 例：http://www.company.com/page
qname	正規のXML名（名前空間とローカルパート）
notation	Notation属性

注：XMLデータタイプに関する詳細は、次のURLを参照してください。

http://www.w3.org/TR/xmlschema-2/

▼ この例題は、2つのパラメータを定義します。

```
C_TEXT($1)
C_TEXT($2)
SET WEB SERVICE PARAMETER("city";$1)
SET WEB SERVICE PARAMETER("country";$2)
```

参照

GET WEB SERVICE RESULT

CALL WEB SERVICE

CALL WEB SERVICE (アクセスURL; soapAction; メソッド名; 名前空間{; 複合型})

引数	タイプ		説明
アクセスURL	文字列	→	Webサービスへのアクセス URL
soapAction	文字列	→	SOAPActionフィールドの内容
メソッド名	文字列	→	メソッドの名前
名前空間	文字列	→	名前空間
複合型	倍長整数	→	複合型の設定（省略すると、単純型）

CALL WEB SERVICEコマンドは、HTTPリクエストを送信してWebサービスを呼び出すために使用します。このリクエストには、**SET WEB SERVICE PARAMETER**コマンドを使用して事前に作成されたSOAPメッセージが納められます。

次に**SET WEB SERVICE PARAMETER**コマンドを呼び出すと、新しくリクエストが作成されます。**CALL WEB SERVICE**コマンドを実行した場合も、以前に呼び出したWebサービスの結果が消去され、新しい結果と置き換えられます。

＜アクセスURL＞には、Webサービスへのアクセスが可能である、完全なURLを渡します（このURLとWSDLファイルのURLを混同しないでください。WSDLファイルはWebサービスを説明するものです）。

暗号化モードでのアクセス（SSL）：SSLを使用した暗号化モードでWebサービスを利用したい場合は、URLの初めに“http://”ではなく“https://”を渡します。この設定により、自動的に暗号化モードでの接続が可能になります。

＜soapAction＞には、リクエストのSOAPActionフィールドの内容を渡します。通常、このフィールドには、“サービス名#メソッド名”という値を納めます。

＜メソッド名＞には、実行しようとするリモートメソッド（Webサービスに属する）の名前を渡します。

＜名前空間＞には、SOAPリクエストで使用するXML名前空間を渡します。XML名前空間に関する詳細は、前述の「名前空間のカスタマイズ」の節を参照してください。

任意の引数である＜複合型＞は、送受信するWebサービスのパラメータの設定（**SET WEB SERVICE PARAMETER**コマンドと**GET WEB SERVICE RESULT**コマンドを使用して定義する）を指定します。引数＜複合型＞の値は、Webサービスの公開モード（DOCまたはRPC、前述の「複合型の処理」の節を参照）と独自のパラメータによって決定します。

＜複合型＞には、「Web Services(Client)」テーマにある以下の定数のうちいずれかを渡します。

定数	タイプ	値
Web Service Dynamic	倍長整数	0（デフォルト）
Web Service Manual In	倍長整数	1
Web Service Manual Out	倍長整数	2
Web Service Manual	倍長整数	3

定数はそれぞれ、Webサービスの“設定”に対応しています。この設定は、公開モード（RPC／DOC）と実装するパラメータタイプ（入力／出力、単純／複合）の組み合わせを表わします。

注：パラメータの“入力”や“出力”特性はプロキシメソッドやWebサービスの観点から判断される点に留意してください。

・“入力”パラメータは、プロキシメソッドへ渡される値であり、したがってWebサービスへ渡される値です。

・“出力”パラメータは、Webサービスから返される値であり、したがってプロキシメソッドから返される値です（通常は $0 を使用）。

次の表は、考えられる設定とそれに対応する定数を示しています。

	入力パラメータ	
出力パラメータ	単純	複合
単純	Web Service Dynamic（RPCモード）	Web Service Manual In（RPCモード）
複合	Web Service Manual Out（RPCモード）	Web Service Manual（RPCまたは DOCモード）

したがって、以下に説明する5つの設定を実装することができます。すべてのケースにおいて、4th DimensionはWebサービスへ送信されるSOAPリクエストのフォーマットとそのエンベロープを処理します。使用した設定に従ってリクエストの内容をフォーマットするかどうかは、ユーザ次第です。

注：データ配列に関しては、複合型のXMLであるにも関わらず、4Dは単純型として処理します。

RPCモード、単純入力と単純出力

これは最も利用しやすい設定です。この場合、引数＜複合型＞には**Web Service Dynamic**定数を指定するか、または省略します。送信されたパラメータと受信した応答は、事前処理なしに直接処理されます。

後述する**GET WEB SERVICE RESULT**コマンドの例題を参照してください。

RPC モード、複合入力と単純出力

この場合、引数＜複合型＞には**Web Service Manual In**定数を指定します。この設定を使用すると、**SET WEB SERVICE PARAMETER**コマンドを使用し、“手動で”各XMLソース要素をBLOB形式でWebサービスに渡さなければなりません。

最初のBLOBを妥当なXML要素としてフォーマットするかどうかはユーザ次第です。その一番目の要素として、このBLOBには最終的なリクエストの<Body>要素の最初の明示的な“子”要素を必ず納めなければなりません。

▼ 例題：

```
C_BLOB($1)
C_BOOLEAN($0)
SET WEB SERVICE PARAMETER("MyXMLBlob";$1)
CALL WEB SERVICE("http://my.domain.com/my_service";
                 "MySoapAction";"http://my.namespace.com/"; Web Service Manual In)
GET WEB SERVICE RESULT($0;"MyOutputVar";*)
```

RPC モード、単純入力と複合出力

この場合、引数＜複合型＞には**Web Service Manual Out**定数を指定します。各出力パラメータは、WebサービスからXML要素の形式でBLOB内に納めて返されます。**GET WEB SERVICE RESULT**コマンドを使用して、このパラメータを取得します。この後、4DのXMLコマンドを使用して、取得したBLOBの内容を解析することができます。

▼ 例題：

```
C_BLOB($0)
C_BOOLEAN($1)
SET WEB SERVICE PARAMETER("MyInputVar";$1)
CALL WEB SERVICE("http://my.domain.com/my_service";
                 "MySoapAction";"http://my.namespace.com/"; Web Service Manual Out)
GET WEB SERVICE RESULT($0;"MyXMLOutput";*)
```

RPC モード、複合入力と複合出力

この場合、引数＜複合型＞には**Web Service Manual**定数を指定します。前述した2種類の設定と同様に、それぞれの入力および出力パラメータはBLOB内にXML要素の形式で納めなければなりません。

▼ 例題：

```
C_BLOB($0)
C_BLOB($1)
SET WEB SERVICE PARAMETER("MyXMLInputBlob";$1)
CALL WEB SERVICE("http://my.domain.com/my_service";
                 "MySoapAction";"http://my.namespace.com/"; Web Service Manual)
GET WEB SERVICE RESULT($0;"MyXMLOutput";*)
```

DOC モード

DOC Webサービスを呼び出すためのプロキシメソッドは、複合型の入力および出力パラメータを用いてRPC Webサービスを呼び出すプロキシメソッドとよく似ています。

これら2つの設定における唯一の相違点は、送受信を行うBLOBパラメータのXML内容のレベルにあります。4th Dimensionの観点から見ると、SOAPリクエストの作成と送信は全く同じです。

▼ 例題：

```
C_BLOB($0)
C_BLOB($1)
SET WEB SERVICE PARAMETER("MyXMLInput";$1)
CALL WEB SERVICE("http://my.domain.com/my_service";
                 "MySoapAction";"http://my.namespace.com/"; Web Service Manual)
GET WEB SERVICE RESULT($0;"MyXMLOutput";*)
```

注：DOC Webサービスの場合、パラメータとして渡した文字列の値（上の例では“MyXMLInput”および“MyXMLOutput”）は重要ではなく、空の文字列を渡すことさえ可能です。実際のところ、これらの値はXML文書を含むSOAPリクエストでは使用されません。それでも、これらのパラメータは必ず渡さなくてはなりません。

DOCモード（または複合型を用いたRPCモード）で公開されたWebサービスを使用する際には、次のような処理を行うことをお勧めします。

■ クライアントWebサービスウィザードを用いてプロキシメソッドを生成します。プロキシメソッドを次のように呼び出します。

```
$XMLresultBlob:=$DOCproxy_Method($XMLparamBlob)
```

■ オンラインテスト（例えば、http://soapclient.com/soaptest.html）を使用して、Webサービスへ送信するSOAPリクエストの内容を把握しておいてください。この種類のツールは、WebサービスのWSDLを基にしたHTMLテストフォームを生成する際に使用されます。

■ <body>の最初の子要素をもとに生成されたXMLコンテンツをコピーします。

■ 実際のパラメータ値をXMLコードに配置するメソッドを作成します。この後、このコードはBLOBである$XMLparamBlobに配置しなければなりません。

■ 応答を解析するため、オンラインテストを使用したり、または帰される要素を定義しているWSDLを利用することもできます。

リクエストが正しく送り出され、Webサービスがそれを受け付けた場合、システム変数OKには1が代入されます。それ以外の場合には、0が代入され、エラーが返されます。

参照：

SET WEB SERVICE PARAMETER、GET WEB SERVICE RESULT

GET WEB SERVICE RESULT

GET WEB SERVICE RESULT (戻り値{; パラメータ名}{; *})

引数	タイプ		説明
戻り値	変数	←	Webサービスから返される値
パラメータ名	文字列	→	取り出されるパラメータの名前
*		→	メモリの解放

GET WEB SERVICE RESULTコマンドは、実行された処理の結果としてWebサービスから返される値を取得するために使用します。

注：このコマンドは、CALL WEB SERVICEコマンドを実行した後に使用しなければなりません。

引数<戻り値>には、Webサービスから返される値を受け取ります。この引数には4D変数を渡してください。通常、この変数は$0であり、プロキシメソッドから返される値に相当します。ただし、中間変数を使用することができます（プロセス変数のみ使用可能）。

注：使用される4D変数や配列はそれぞれ、「コンパイラ」および「配列」テーマのコマンドを使用して事前に定義しなければなりません。

任意の引数<パラメータ名>を使用して、取り出されるパラメータの名前を指定することができます。しかし、大半のWebサービスは単一の値だけを返すため、通常このパラメータは必要ありません。

任意の引数である * は、リクエストの処理に占有されていたメモリを解放するようプログラムに指示します。Webサービスから送信された最後の値を取得した後は、この引数を渡さなくてはなりません。

▼ Webサービスにより、世界中の任意の都市における現在時刻が返される場合を想定してみましょう。Webサービスへ送信するパラメータは、都市の名前とその国のコードです。引き換えに、Webサービスからは対応する時刻が送信されます。Webサービスを呼び出すプロキシメソッドは、次の通りです。

```
C_TEXT($1)
C_TEXT($2)
C_TIME($0)
SET WEB SERVICE PARAMETER("city";$1)
SET WEB SERVICE PARAMETER("country_code";$2)
CALL WEB SERVICE("http://www.citiesoftheworld.com/WS"; "WSTime#City_time";
                 "City_time"; "http://www.citiesoftheworld.com/namespace/default")
If(OK=1)
    GET WEB SERVICE RESULT($0;"return";*)
End if
```

参照：

SET WEB SERVICE PARAMETER

Get Web Service error info

Get Web Service error info (情報タイプ) → 文字列

引数	タイプ		説明
情報タイプ	倍長整数	→	取得する情報
戻り値	文字列	←	前回のSOAPエラーに関する情報

このコマンドは、リモートWebサービスへ送信したSOAPリクエストの実行中に発生した最後のエラーに関する情報を返します。

引数<情報タイプ>を使用して、取得しようとする情報のタイプを指定することができます。「Web Services(Client)」テーマ内にある次の定数のいずれかを渡さなくてはなりません。

定数	タイプ	値
Web Service Error Code	倍長整数	0
Web Service Detailed Message	倍長整数	1
Web Service HTTP Error code	倍長整数	2
Web Service Fault Actor	倍長整数	3

これらの定数は、次の値を取得するために使用します。

■ Web Service Error Code：メインエラーコード（4Dにより定義される）。このコードは、システム変数Errorへも返されます。

返されるコードの一覧は次の通りです。

■ 9910：Soap fault（Web Service Fault Actorも参照）

■ 9911：パーサーfault

■ 9912：HTTP fault（Web Service HTTPエラーコードも参照）

■ 9913：ネットワークfault

■ 9914：インターネットfault

■ Web Service Detailed Message：エラーを解説する詳細なメッセージ。メッセージタイプは、メインエラータイプによって異なります。

■ メインエラーコード＝9910（Soap fault）の場合：

SOAP faultの原因が返されます（例：“リモートメソッドが存在しない”）。

■ メインエラーコード＝9911（パーサー fault）の場合：

XML文書内のエラー箇所が返されます。

■ メインエラーコード＝9912（HTTP fault）の場合：

- HTTPエラーが300～400の間である場合（リクエストされた文書の場所に関連した問題）、リクエストされたURLの新しい場所が返されます。

- その他任意のHTTPエラーコードに対しては、<body>が返されます。

■ メインエラーコード＝9913（ネットワークfault）の場合：

ネットワークfaultの原因が返されます（例：“ServerAddress: DNS検索失敗”）。

■ メインエラーコード＝9914（インターネットfault）の場合：

インターネットfaultの原因が返されます。

■ Web Service HTTP Error code：HTTPエラーコード（メインエラーコード9912の場合に使用される）。

■ Web Service Fault Actor：エラーの原因（SOAPプロトコルから返され、メインエラーコード9910の場合に使用される）。

次のような原因が返されます。

■ Version Mismatch

■ Must Understand（サーバは必須属性として定義されたパラメータを理解できない）

■ Client Fault

■ Server Fault

■ Encoding Unknown

情報が取得できない場合には、空の文字列が返されます。

AUTHENTICATE WEB SERVICE

AUTHENTICATE WEB SERVICE (名前; パスワード)

引数	タイプ		説明
名前	文字列	→	ユーザ名
パスワード	文字列	→	ユーザパスワード

AUTHENTICATE WEB SERVICEコマンドにより、クライアントアプリケーションの認証を必要とするWebサービスを使用できるようになります（シンプル認証）。

引数＜名前＞と＜パスワード＞には、要求される識別情報（ユーザ名とパスワード）を渡します。この情報は暗号化され、**CALL WEB SERVICE**コマンドを使用してWebサービスへ送信されるHTTPリクエストに追加されます。したがって、**AUTHENTICATE WEB SERVICE**コマンドは、**CALL WEB SERVICE**コマンドを呼び出す前にコールする必要があります。

認証情報は、各リクエストの後で0へリセットされるため、各**CALL WEB SERVICE**コマンドの前に、**AUTHENTICATE WEB SERVICE**コマンドを使用しなくてはなりません。

認証に失敗すると、SOAPサーバからエラーが返され、**Get Web Service error info**コマンドを用いてこのエラーを識別することができます。

参照：

CALL WEB SERVICE

XML

4th Dimension 2003には、XML（eXtensible Markup Language）データを含むオブジェクトの解析に使用する一連の新しいコマンドが導入されています。XMLはタグの使用を基本とする言語で、やり取りされるデータやその構造に関して正確に記述することができます。XMLに関する詳細については、http://xmlfr.org等のサイトを参照してください。

注：この章で使用するXMLという用語についての簡単な説明は、後述の付録 A「XML用語集」を参照してください。

テキスト、URL、文書、BLOBタイプのオブジェクトは、これらのコマンドを使用して解析することができます。XMLを解析するため、4th DimensionはApache Software Foundationが開発した「Xerces.dll」という名前のライブラリを使用します。

4th Dimensionは、XMLバージョン1.0をサポートします。

4th Dimensionにおいて、XMLオブジェクト解析に使用するメソッドの構造は次の通りです。

■ ソースのオープンと解析：**Parse XML source**、**Parse XML variable**

■ 要素の識別と読み込み：**Count XML elements**、**Get First XML element**、**Get Next XML element**、**Get XML element**、**GET XML ELEMENT NAME**、**GET XML ELEMENT VALUE**

■ 属性の識別：**Count XML attributes**、**GET XML ATTRIBUTE BY INDEX**、**GET XML ATTRIBUTE BY NAME**

■ エラーおよび情報の取得：**GET XML ERROR**、**Parse XML information**

■ ソースのクローズ：**CLOSE XML**

注：また、4th Dimension 2003では、読み込み／書き出しエディタを使用して、XML文書の読み込みや書き出しを行うこともできます。この件に関する詳細は、前述の「XMLフォーマットでの読み込み／書き出し」の節を参照してください。

Parse XML source

Parse XML source (文書{; 妥当性{; dtd}}) → 要素参照

引数	タイプ		説明
文書	文字列	→	文書のアクセスパス
妥当性	ブール	→	True= DTDによる妥当正検証
			False= 妥当性検証なし
dtd	変数	→	DTDの場所
要素参照	文字列	←	XML要素の参照（16桁）

Parse XML source関数は、XML構造を含む文書を解析し、この文書への参照を返します。また、このコマンドは文書の妥当性を検証することができます（または、検証を行いません）。

文書の場所は、ディスク上であっても、インターネット／イントラネット上でも構いません。

引数＜文書＞には、次のいずれかを渡すことができます。

■ 標準的なフルアクセスパス（Windowsでは、C:¥Folder¥File¥...、MacOS上では、MacintoshHD:Folder:File）

■ Unixパス（http://www.site.com/File または file://Myfile）

＜文書＞にファイル名だけを渡した場合、コマンドはデータベースのストラクチャファイルと同じ階層内を検索します。MacOSのソフトウェアパッケージの場合には、コマンドはソフトウェアパッケージと同じ階層内のファイルを検索します。

ブールタイプの引数＜妥当性＞を使用し、DTDを用いて構造の妥当性検証を行うかどうかを指定します。

■ ＜妥当性＞がTrue（真）の場合、XML構造の妥当性が検証されます。この場合、パーサーは、文書内で定義または参照されるDTD、あるいは引数＜dtd＞で指定されたDTDに基づいて、文書のXML構造の妥当性検証を試みます。

■ ＜妥当性＞がFalse（偽）の場合、XML構造の妥当性検証は行われません。

3番目の引数＜dtd＞を使用し、文書解析に使用する特定のDTDを指定します。この引数を使用する場合、コマンドはXML文書内で参照されるDTDを無視します。

DTDを指定する方法には2通りあります。

■ **参照として指定**：これを行うには、引数＜dtd＞に新しいDTDのフルアクセスパスを渡します。指定された文書に有効なDTDが含まれていない場合、引数＜dtd＞は無視され、エラーが生成されます。

■ BLOBまたはテキスト内で指定：この場合、引数の内容が“<xml!”で開始していれば、4Dはそれを DTDであるものとみなします。それ以外の場合、4Dはそれをアクセスパスであるものとみなします。

妥当性検証が行われない場合（DTDなし、DTDへの不正なURL等）、エラーが生成されます。システム変数Errorにはエラー番号が代入されます。**ON ERR CALL**コマンドでインストールされたメソッドを使用して、このエラーを阻止することができます。

要素参照：このコマンドは、メモリ上での文書の仮想構造への参照を表わす16桁の文字列を返します。参照である＜要素参照＞は、他のXML解析用コマンドと一緒に使用しなければなりません。

コマンドが正常に終了すると、システム変数OKには1が代入されます。それ以外の場合には0が代入されます。

例題

▼ ディスク上にあるXML文書を開き、妥当性検証は行わない。

```
$xml_Struct_Ref:=Parse XML source("C:\import.xml")
```

▼ データベースのストラクチャファイルと同じ階層上にあるXML文書を開き、妥当性検証は行わない。

```
$xml_Struct_Ref:=Parse XML source("import.xml")
```

▼ ディスク上にあるXML文書を開き、ディスク上のDTDを使用して妥当性を検証する。

```
$xml_Struct_Ref:=Parse XML source("C:\import.xml";True; "C:\import_dtd.xml")
```

▼ 指定されたURLにあるXML文書を開き、妥当性検証は行わない。

```
$xml_Struct_Ref:=Parse XML source("http://www.4D.com/xml/import.xml")
```

参照：

Parse XML variable

Parse XML variable

Parse XML variable (変数{; 妥当性{; dtd}}) → 要素参照

引数	タイプ		説明
変数	BLOB／テキスト	→	変数の名前
妥当性	ブール	→	True= DTDによる妥当正検証 False= 妥当性検証なし
dtd	変数	→	DTDの場所
要素参照	文字列	-	XML要素の参照（16桁）

Parse XML variable関数は、XML構造を含むBLOBやテキストタイプの変数を解析し、この変数への参照を返します。このコマンドは文書の妥当性を検証することができます（または、検証を行いません）。

引数＜変数＞には、XMLオブジェクトを含むBLOBやテキストタイプの変数の名前を渡します。

ブールタイプの引数＜妥当性＞を使用し、DTDを用いて構造の妥当性検証を行うかどうかを指定します。

■ ＜妥当性＞がTrue（真）の場合、XML構造の妥当性が検証されます。この場合、パーサーは、文書内で定義または参照されるDTD、あるいは引数＜dtd＞で指定されたDTDに基づいて、文書のXML構造の妥当性検証を試みます。

■ ＜妥当性＞がFalse（偽）の場合、XML構造の妥当性検証は行われません。

3番目の引数＜dtd＞を使用し、文書解析に使用する特定のDTDを指定します。この引数を使用する場合、コマンドはXML変数内で参照されるDTDを無視します。

DTDを指定する方法には2通りあります。

■ **参照として指定**：これを行うには、引数＜dtd＞に新しいDTDのフルアクセスパスを渡します。指定された文書に有効なDTDが含まれていない場合、引数＜dtd＞は無視され、エラーが生成されます。

■ **BLOBまたはテキスト内で指定**：この場合、引数の内容が“<xml!”で開始していれば、4DはそれをDTDであるものとみなします。それ以外の場合、4Dはそれをアクセスパスであるものとみなします。

妥当性検証が行われない場合（DTDなし、DTDへの不正なURL等）、エラーが生成されます。システム変数Errorにはエラー番号が代入されます。

ON ERR CALLコマンドでインストールされたメソッドを使用して、このエラーを阻止することができます。

要素参照：このコマンドは、メモリ上での文書の仮想構造への参照を表わす16桁の文字列を返します。参照である＜要素参照＞は、他のXML解析用コマンドと一緒に使用しなければなりません。

コマンドが正しく実行されると、システム変数OKには1が代入されます。それ以外の場合には0が代入されます。

例題

▼4Dのテキスト変数内にあるXMLオブジェクトを開き、妥当性検証は行わない。

```
C_TEXT(myTextVar)
C_TIME(vDoc)
C_STRING(16;$xml_Struct_Ref)

vDoc:=Open document("Document.xml")
If(OK=1)
    RECEIVE PACKET(vDoc;myTextVar;32000)
    CLOSE DOCUMENT(vDoc)
    $xml_Struct_Ref:=Parse XML variable(myTextVar)
End if
```

▼4DのBLOB内にあるXML文書を開き、妥当性検証は行わない。

```
C_BLOB(myBlobVar)
C_STRING(16;$xml_Struct_Ref)

DOCUMENT TO BLOB(“c:\import.xml” ;myBlobVar)
$xml_Struct_Ref:=Parse XML variable(myBlobVar)
```

参照：

Parse XML source

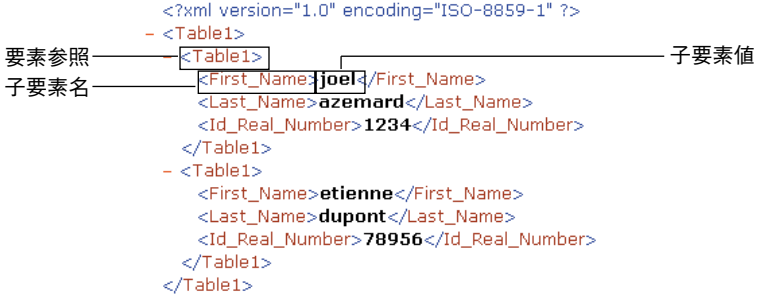
Get First XML element

Get First XML element (要素参照{; 子要素名{; 子要素値}}) → 子要素参照

引数	タイプ		説明
要素参照	文字列	→	XML要素参照
子要素名	文字列	←	選択されたフィールドの名前
子要素値	文字列	←	選択されたフィールドの値
子要素参照	文字列	←	XML参照（16桁）

このコマンドは、参照として渡したXML要素の最初の“子”への参照を返します。この参照は他のXML解析用コマンドで使用することができます。

引数<子要素名>と<子要素値>を指定すると、子要素の名前と値をそれぞれ受け取ります。



コマンドが正常に終了すると、システム変数OKには1が代入されます。それ以外の場合には0が代入されます。

例題

▼親ルートの最初のXML要素への参照を検索します。まず初めに、XML構造（C:¥import.xml）がBLOBにロードされます。

```
C_BLOB(myBlobVar)
C_STRING(16;$xml_Parent_Ref;$xml_Child_Ref)

DOCUMENT TO BLOB("c:\import.xml";myBlobVar)
$xml_Parent_Ref:=Parse XML variable(myBlobVar)
$xml_Child_Ref:=Get First XML element($xml_Parent_Ref)
```

▼親ルートの最初のXML要素の参照と名前、値を検索します。XML構造（C:¥import.xml）がBLOBにロードされます。

```
C_BLOB(myBlobVar)
C_STRING(16;$xml_Parent_Ref;$xml_Child_Ref)
C_TEXT($childName;$childValue)
DOCUMENT TO BLOB("c:\import.xml";myBlobVar)
$xml_Parent_Ref:=Parse XML variable(myBlobVar)
$xml_Child_Ref:=Get First XML element($xml_Parent_Ref;
                                      $childName;$childValue)
```

参照：

Get Next XML element

Get Next XML element

Get Next XML element (要素参照{; 子要素名{; 子要素値}}) → 子要素参照

引数	タイプ		説明
要素参照	文字列	→	XML要素参照
子要素名	文字列	←	選択されたフィールドの名前
子要素値	文字列	←	選択されたフィールドの値
子要素参照	文字列	←	XML参照（16桁）

このコマンドは、参照として渡したXML要素の次の“子”への参照を返します。この参照は他のXML解析用コマンドで使用することができます。

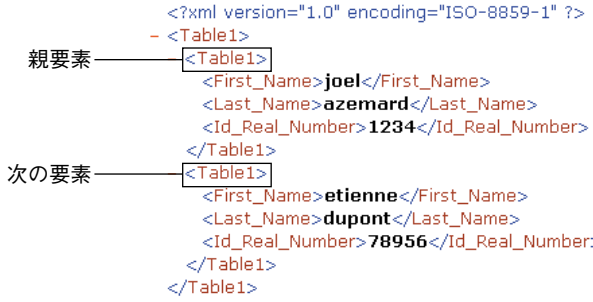
引数<子要素名>と<子要素値>を指定すると、子要素の名前と値をそれぞれ受け取ります。

このコマンドは、引数として渡されたXML要素のすべての“子”を連続して解析するために使用します。最後の“子”の解析後、システム変数OKには0が代入されます。

コマンドが正常に終了し、解析された要素が参照要素の最後の“子”でなければ、システム変数OKに1が代入されます。エラーが発生するか、解析された要素が参照要素の最後の“子”である場合、システム変数OKには0が代入されます。

▼引数として渡した要素に続く、次のXML要素への参照を検索します。

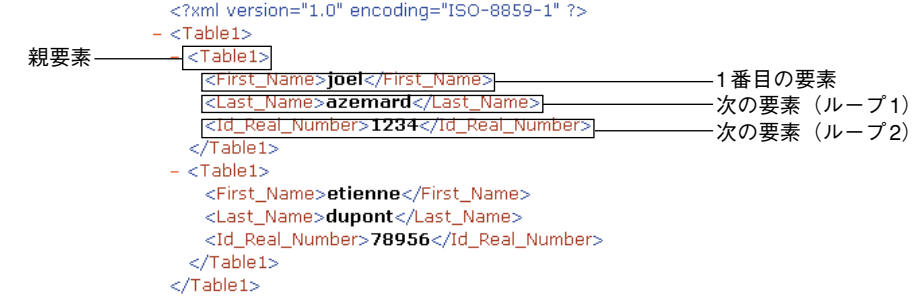
```
C_STRING(16;$xml_Parent_Ref;$next_XML_Ref)
$next_XML_Ref:=Get Next XML element($xml_Parent_Ref)
```




```
<?xml version="1.0" encoding="ISO-8859-1" ?>
- <Table1>
    <Table1>
      <First_Name>joel</First_Name>
      <Last_Name>azemard</Last_Name>
      <Id_Real_Number>1234</Id_Real_Number>
    </Table1>
    <Table1>
      <First_Name>etienne</First_Name>
      <Last_Name>dupont</Last_Name>
      <Id_Real_Number>78956</Id_Real_Number>
    </Table1>
</Table1>
```

▼引数として渡した親要素に続く、XML要素の参照ループを検索します。

```
C_STRING(16;$xml_Parent_Ref;$first_XML_Ref;$next_XML_Ref)
$first_XML_Ref:=Get First XML element($xml_Parent_Ref)
$next_XML_Ref:=$first_XML_Ref
While(OK=1)
    $next_XML_Ref:=Get Next XML element($next_XML_Ref)
End while
```




```
<?xml version="1.0" encoding="ISO-8859-1" ?>
- <Table1>
    <Table1>
      <First_Name>joel</First_Name>
      <Last_Name>azemard</Last_Name>
      <Id_Real_Number>1234</Id_Real_Number>
    </Table1>
  - <Table1>
      <First_Name>etienne</First_Name>
      <Last_Name>dupont</Last_Name>
      <Id_Real_Number>78956</Id_Real_Number>
    </Table1>
</Table1>
```

参照：

Get First XML element

GET XML ATTRIBUTE BY NAME

GET XML ATTRIBUTE BY NAME (要素参照; 属性名; 属性値)

引数	タイプ		説明
要素参照	文字列	→	XML要素参照
属性名	文字列	→	属性の名前
属性値	変数	←	属性の値

このコマンドを使用し、名前により指定された属性の値を取得することができます。

＜要素参照＞にはXML要素の参照を渡し、＜属性名＞には値を知りたい属性の名前を渡します。引数＜属性値＞には、属性の値が返されます。4th Dimensionは、取得した値を引数として渡された変数と同じタイプに変換します。

XML要素内に＜属性名＞で指定した属性が存在しない場合、エラーが返されます。

指定されたものと同じ名前を持つXML要素の属性がいくつか存在する場合、最初に発見された属性の値だけが返されます。

コマンドが正常に終了すると、システム変数OKには1が代入されます。それ以外の場合には0が代入されます。

▼ このメソッドを使用し、名前を用いてXML属性の値を取得します。

```
C_BLOB(myBlobVar)
C_STRING(16;$xml_Parent_Ref;$xml_Child_Ref)
C_LONGINT($LineNum)
$xml_Parent_Ref:=Parse XML variable(myBlobVar)
$xml_Child_Ref:=Get First XML element($xml_Parent_Ref)
GET XML ATTRIBUTE BY NAME($xml_Child_Ref;"N";$LineNum)
```

このメソッドを下記の例に対して実行すると、$LineNumには値1が納められます。

```
<?xml version="1.0" ?>
- <STANZA>
    <LINE N="1">I heard a thousand blended notes,</LINE>
    <LINE N="2">While in grove I sate reclined,</LINE>
    <LINE N="3">In that sweet mood when pleasant thoughts</LINE>
    <LINE N="4">Bring sad thoughts to the mind.</LINE>
  </STANZA>
```

参照：

GET XML ATTRIBUTE BY INDEX

GET XML ATTRIBUTE BY INDEX

GET XML ATTRIBUTE BY INDEX (要素参照; 属性インデックス; 属性名{; 属性値})

引数	タイプ		説明
要素参照	文字列	→	XML要素参照
属性インデックス	文字列	→	属性のインデックス番号
属性名	変数	←	属性の名前
属性値	変数	←	属性の値

このコマンドを使用し、インデックス番号で指定した属性の名前、ならびにオプションとしてその値を調べることができます。

＜要素参照＞にはXML要素の参照を渡し、＜属性インデックス＞には名前を知りたい属性のインデックス番号を渡します。引数＜属性名＞には、属性名が返され、引数＜属性値＞を渡した場合には、この引数に属性の値が返されます。この場合、4th Dimensionは、取得した値を引数として渡された変数と同じタイプに変換します。

注：テキストフォーム上に表示されたXMLファイル内の属性の位置とインデックス番号は対応していません。XMLにおいて、属性のインデックスは、属性が（属性名に基づいて）アルファベット順に並べられた際の位置を表わします。実例に関しては、後述するCount XML attributes関数の例題を参照してください。

＜属性インデックス＞に渡された値が、XML要素内にある属性の数よりも大きい場合、エラーが返されます。

コマンドが正常に終了すると、システム変数OKには1が代入されます。エラーが発生した場合には0が代入されます。

▼ 後述する**Count XML attributes**関数の例題を参照してください。

参照：

GET XML ATTRIBUTE BY NAME

Count XML attributes

Count XML attributes (要素参照) → 倍長整数

引数	タイプ		説明
要素参照	文字列	→	XML要素参照
戻り値	倍長整数	←	属性の数

このコマンドは、＜要素参照＞で指定されたXML要素内にあるXML属性の数を返します。

コマンドが正常に終了すると、システム変数OKには1が代入されます。エラーが発生した場合には0が代入されます。

▼配列内の要素の値を取得する前に、次のXML要素内の属性の数を知りたいものとします。

```
<?xml version="1.0" ?>
- <LINES>
    <LINE N="1" Font="verdana" size="10">this is a line</LINE>
    <LINE N="2" Font="charcoal" size="12">and another line</LINE>
    <LINE N="3" Font="verdana" size="18">and we stop here</LINE>
  </LINES>
```

```
C_BLOB(myBlobVar)
C_STRING(16;$xml_Parent_Ref;$xml_Child_Ref)
C_TEXT(myResult)
C_LONGINT($numAttributes)
$xml_Parent_Ref:=Parse XML variable(myBlobVar)
$xml_Child_Ref:=Get First XML element($xml_Parent_Ref)
$numAttributes:=Count XML attributes($xml_Child_Ref)
ARRAY TEXT(tAttrib;$numAttributes)
Loop($i;1;$numAttributes)
        GET XML ATTRIBUTE BY INDEX($xml_Child_Ref;$i;tAttrib{$i})
End of loop
```

上記の例題では、$numAttributesは3であり、tAttrib{1}には“Font”、tAttrib{2}には“N”、tAttrib{3}には“size”が代入されます。

注：テキストフォーム上に表示されたXMLファイル内の属性の位置とインデックス番号は対応していません。XMLにおいて、属性のインデックスは、属性が（属性名に基づいて）アルファベット順に並べられた際の位置を表わします。

参照：

Count XML elements

GET XML ELEMENT NAME

GET XML ELEMENT NAME (要素参照; 要素名)

引数	タイプ		説明
要素参照	文字列	→	XML要素参照
要素名	変数	←	要素の名前

このコマンドは、＜要素参照＞で指定されたXML要素の名前を引数＜要素名＞に代入して返します。

コマンドが正常に終了すると、システム変数OKには1が代入されます。エラーが発生した場合には0が代入されます。

▼ このメソッドは、要素$xml_Element_Refの名前を返します。

```
C_STRING(16;$xml_Element_Ref)
C_TEXT($name)
GET XML ELEMENT NAME($xml_Element_Ref;$name)
```

参照：

GET XML ELEMENT VALUE、Get XML element

GET XML ELEMENT VALUE

GET XML ELEMENT VALUE (要素参照; 要素値)

引数	タイプ		説明
要素参照	文字列	→	XML要素参照
要素値	変数	←	要素の値

このコマンドは、＜要素参照＞で指定されたXML要素の値を引数＜要素値＞に代入して返します。4th Dimensionは、取得した値を引数として渡された変数と同じタイプに変換します。

コマンドが正常に終了すると、システム変数OKには1が代入されます。エラーが発生した場合には0が代入されます。

▼ このメソッドは、要素$xml_Element_Refの値を返します。

```
C_STRING(16;$xml_Element_Ref)
C_REAL($value)
GET XML ELEMENT NAME($xml_Element_Ref;$value)
```

参照：

GET XML ELEMENT NAME、Get XML element

Get XML element

Get XML element (要素参照; 要素名; インデックス; 要素値) → 子要素参照

引数	タイプ		説明
要素参照	文字列	→	XML要素参照
要素名	文字列	→	取得する要素名
インデックス	倍長整数	→	取得する要素のインデックス番号
要素値	変数	←	要素の値
子要素参照	文字列	←	XML参照（16桁）

このコマンドは、引数＜要素名＞と＜インデックス＞に対応する“子”要素への参照を返します。

また、要素の値が引数＜要素値＞に返されます。

コマンドが正常に終了すると、システム変数OKには1が代入されます。エラーが発生した場合には0が代入されます。

参照：

GET XML ELEMENT NAME、GET XML ELEMENT VALUE

Count XML elements

Count XML elements (要素参照; 要素名) → 倍長整数

引数	タイプ		説明
要素参照	文字列	→	XML要素参照
要素名	文字列	→	カウントするXML要素の名前
戻り値	倍長整数	←	要素の数

このコマンドは、引数＜要素参照＞に指定した親要素と＜要素名＞の名前に対応する“子”要素の数を返します。

コマンドが正常に終了すると、システム変数OKには1が代入されます。エラーが発生した場合には0が代入されます。

CLOSE XML

CLOSE XML (要素参照)

引数	タイプ		説明
要素参照	文字列	→	XMLルート要素の参照

このコマンドは、＜要素参照＞で指定したXMLオブジェクトが占有していたメモリを解放します。

＜要素参照＞がXMLのルートオブジェクトではない場合、エラーが生成されます。

コマンドが正常に終了すると、システム変数OKには1が代入されます。エラーが発生した場合には0が代入されます。

GET XML ERROR

GET XML ERROR (要素参照; エラーテキスト{; 行{; 列}})

引数	タイプ		説明
要素参照	文字列	→	XML要素参照
エラーテキスト	変数	←	エラーテキスト
行	変数	←	行番号
列	変数	←	列番号

このコマンドは、引数＜要素参照＞で指定されたXML要素を処理する際に発生したエラーの説明を引数＜エラーテキスト＞に代入して返します。返される情報は、Xerces.DLLライブラリから提供されます。

任意の引数＜行＞と＜列＞は、エラー箇所を示します。指定した変数には行番号、およびエラーの原因となる表現式の最初の桁位置がそれぞれ返されます。

コマンドが正常に終了すると、システム変数OKには1が代入されます。エラーが発生した場合には0が代入されます。

Parse XML information

Parse XML information (要素参照; xml情報) → 文字列

引数	タイプ		説明
要素参照	文字列	→	XMLルート要素の参照
xml情報	倍長整数	→	取得する情報のタイプ
戻り値	文字列	-	XML情報の値

このコマンドを使用して、＜要素参照＞で指定したXML要素に関する各種情報を取得します。

＜xml情報＞には、取得する情報のタイプを示すコードを渡します。「XML」テーマ内にある次の定義済定数を使用することができます。

定数	タイプ	値
PUBLIC ID	倍長整数	1
SYSTEM ID	倍長整数	2
DOCTYPE Name	倍長整数	3
Encoding	倍長整数	4
Version	倍長整数	5
Document URI	倍長整数	6

これらの定数は、次の情報を表わします。

■ PUBLIC ID：文書が準拠するDTDの公開識別子（FPI）（DOCTYPE xxx PUBLICタグがある場合）。

■ SYSTEM ID：システム識別子

■ DOCTYPE Name：DOCTYPEマーカーで定義されたルート要素の名前

■ Encoding：使用される符号化方式（UTF-8、ISO...）

■ Version：受け入れられるXMLバージョン

■ Document URI：DTDのURL

参照：

GET XML ERROR

新しい印刷コマンド

「印刷」テーマに新しいコマンドが追加されて機能強化されました。これらのコマンドにより、プリンタおよびプリンタが提供するオプションをより適確に管理することができます。

MacOS上の注意：MacOS 9では、システムアーキテクチャに関する理由により、「印刷」テーマの新規コマンドとオプションのなかには使用できないものがあります。本テキスト内では、これらの制約について示されています。

「印刷」テーマのいくつかの既存コマンドが機能拡張されている点に注意してください。この件については、後述の「印刷の最適化」の節で説明しています。

SET CURRENT PRINTER

SET CURRENT PRINTER (プリンタ名)

引数	タイプ		説明
プリンタ名	文字列	→	使用するプリンタの名前

注：MacOS 9上では、このコマンドは動作しません。

SET CURRENT PRINTERコマンドを使用し、現行の4Dアプリケーションでの印刷に使用するプリンタを指定することができます。

引数<プリンタ名>には、選択するプリンタの名前を渡します。使用できるプリンタ一覧を取得するには、このコマンドの前に新規コマンドである**PRINTERS LIST**コマンドを使用します。

<プリンタ名>に空の文字列を渡した場合、システムで定義された現行のプリンタが使用されます。

新しいプリンタを選択すると、現在のプリントオプションがリセットされます。したがって、**SET CURRENT PRINTER**コマンドは、必ず**PAGE SETUP**や**SET PRINT OPTION**コマンドを使用する前に呼び出してください。そうしないと、前に設定されたパラメータが失われます。

このコマンドは、**PRINT SELECTION**、**PRINT LABEL**、**PRINT RECORD**、**Print form**、および**QR REPORT**コマンドと一緒に使用することができ、「デザイン」モードを含め、4th Dimensionにおけるすべての印刷に対して適用されます。

プリンタの選択が正しく実行されると、システム変数OKには1が代入されます。選択を正常に行えない場合（例えば、指定したプリンタが見つからない場合）は、システム変数OKに0が代入され、プリンタは現行のまま変わりません。

参照：

Get current printer、PRINTERS LIST

Get current printer

Get current printer→ 文字列

引数	タイプ		説明
このコマンドに引数はありません。			
戻り値	文字列	←	現在のプリンタの名前

注：MacOS 9上では、このコマンドは動作しません。

Get current printer関数は、4Dアプリケーションで定義された現在のプリンタ名を返します。デフォルトとして、4Dの起動時には、システムで定義されたプリンタが現行プリンタとなります。

プリントサーバ（スプーラ）を使用して、現行プリンタを管理している場合、フルアクセスパス（Windows）またはスプーラの名前（MacOS）が返されます。

使用できるプリンタ一覧および追加情報を取得するには、**PRINTERS LIST**コマンドを使用します。現在のプリンタを変更するには、**SET CURRENT PRINTER**を使用してください。

プリンタがインストールされていない場合、システム変数OKには0が代入されます。それ以外の場合には1が代入されます。

参照：

SET CURRENT PRINTER、PRINTERS LIST

PRINTERS LIST

PRINTERS LIST (名前配列{; 場所配列{; モデル配列}})

引数	タイプ		説明
名前配列	テキスト配列	←	プリンタ名
場所配列	テキスト配列	←	プリンタの場所
モデル配列	テキスト配列	←	プリンタのモデル

注：MacOS 9上では、このコマンドは動作しません。

PRINTERS LISTコマンドは、引数として渡された各配列にそのマシンで使用できるプリンタの名前、ならびにオプションとしてプリンタの場所とモデルを代入します。

注：プリントサーバ（スプーラ）を使用して、プリンタを管理している場合、フルアクセスパス（Windows）またはスプーラの名前（MacOS）が返されます。

引数＜名前配列＞には、テキスト配列の名前を渡します。コマンドの実行後、この配列には使用できるプリンタの名前が代入されます。任意の引数である＜場所配列＞と＜モデル配列＞を渡した場合、各プリンタのネットワーク上の位置（またはローカルポート）とモデルが返されます。

注：引数＜場所配列＞と＜モデル配列＞は、Windows上でのみ使用できます。

4Dで選択されたプリンタの変更や取得を行うには、**SET CURRENT PRINTER**コマンドと**Get current printer**関数を使用してください。

Windows上では、プリンタ名はOSレベルで手動にて変更することができます。一方、プリンタの場所とモデルタイプは、その物理的特性に関連しています。したがって、任意の配列の値を使用して、選択したプリンタの特性を調べることができます。通常は、クライアントマシンがすべて同じプリンタを使用していることをチェックします。

MacOS上では、プリンタ名（プリンタサーバの名前）を使用して、このチェックを行います。このプリンタ名は、接続している各マシンに対して同じ名前になります。

コマンドが正常に終了すると、システム変数OKには1が代入されます。それ以外の場合には0が代入され、配列には何も返されません。

参照：

SET CURRENT PRINTER、Get current printer

SET PRINT OPTION

SET PRINT OPTION (オプション; 値1{; 値2})

引数	タイプ		説明
オプション	倍長整数	→	オプション番号
値1	倍長整数¦文字列	→	オプションの値1
値2	倍長整数	→	オプションの値2

SET PRINT OPTIONコマンドを使用し、プログラムからプリントオプションの値を変更することができます。プリントパラメータを変更する他のコマンド（**PRINT SETTINGS**、引数“>”を使用しない**PRINT SELECTION**）が呼び出されない限り、このコマンドを使用して定義された各オプションは、4Dセッションの間データベース全体に対して適用されます。

引数＜オプション＞を使用し、変更するオプションを指定することができます。値を渡すか、または「Print options」テーマ内にある次の定義済定数のいずれかを渡すことができます。

定数	タイプ	値
Paper option	倍長整数	1
Orientation option	倍長整数	2
Scale option	倍長整数	3
Number of copies option	倍長整数	4
Paper source option	倍長整数	5
Color option	倍長整数	8
Destination option	倍長整数	9
Double sided option	倍長整数	11
Spooler document name option	倍長整数	12

指定した＜オプション＞の新しい値は、引数＜値1＞と（任意の）＜値2＞に渡します。渡す値の数値と種類は、指定したオプションのタイプによって異なります。オプションと考えられる値に関しての詳細は、次の表を参照してください

プロパティ	オプション	定数	値1	値2
用紙	1	**Paper option**	名前 高さ	- 幅
方向	2	**Orientation option**	1=縦方向、2=横方向	-
倍率	3	**Scale option**	数値（%）	-
部数	4	**Number of copies option**	数値	-
用紙ソース	5	**Paper source option**	Windowsのみ インデックス（数値）	-
カラー - N/B	8	**Color option**	Windowsのみ 1=N/B、2=カラー	-
出力先	9	**Destination option**	1=プリンタ、 2=ファイル(PC)／PS(Mac)、 3=PDF（Mac)、 4=EPS（Mac）	- アクセスパス アクセスパス アクセスパス
両面印刷	11	**Double sided option**	Windowsのみ 0=片面（標準） 1=両面	- とじしろ位置：0=左（デフォルト)、1=上
スプーラ文書名	12	**Spooler document name option**	印刷する文書名	-

■ Paper option(1)：**PRINT OPTION VALUES** コマンドを使用して取得した、使用可能なすべての用紙タイプの名前一覧。

＜値1＞（この場合、＜値2＞は省略）に用紙の名前を渡すか、または＜値1＞に用紙の高さ、＜値2＞に用紙の幅を渡します。幅と高さは、画面のピクセル単位で表わさなくてはなりません。

■ Orientation option(2)：1（縦方向）または2（横方向）のいずれかを＜値1＞に渡します。

■ Scale option(3)：倍率を＜値1＞に渡します。プリンタのなかには倍率を変更できないものがあるので注意が必要です。無効な値を渡すと、印刷時にこのプロパティが100%にリセットされます。

■ Number of copies option(4)：印刷する部数を＜値1＞に渡します。

■ Paper source option(5)：使用する用紙トレイのインデックスに対応する数値を、**PRINT OPTION VALUES** コマンドから返されるトレイの配列に渡します。

注：このオプションは、Windows でのみ使用可能です。

■ Color option(8)：＜値1＞に、カラー処理用のモードを示すコードを渡します。1=黒白（モノクロ）、2=カラー。

注：このオプションは、Windows でのみ使用可能です。

■ Destination option(9)：＜値1＞に、印刷先のタイプを示すコードを渡します：1=プリンタ、2=ファイル（PC）／PS（MacOS）、3=PDF ファイル（MacOS のみ）、4=EPS ファイル（MacOS のみ）。

＜値1＞が1以外の場合、出力結果となる文書へのアクセスパスを＜値2＞に渡します。別のパスが指定されるまで、セッションの間このパスが使用されます。出力先に同じ名前のファイルが既に存在する場合は、新しいファイルと置き換えられます。＜値2＞に空の文字列を渡した場合や、この引数を省略した場合（Windows）、印刷時にファイル保存ダイアログボックスが表示されます。

注：MacOS 9 上では、このコマンドは動作しません。

■ Double sided option(11)：0（片面または標準）か1（両面）のいずれかを＜値1＞に渡します。

＜値1＞が1の場合、適用するとじしろを定義することができます。＜値2＞を使用して、0=左とじ（デフォルト）または1=上とじを設定します。

注：このオプションは、Windows でのみ使用可能です。

■ Spooler document name option(12)：＜値1＞に、プリントサーバ文書の一覧に表示しなければならない印刷文書の名前を渡します。標準操作（メソッドの場合はメソッド名、レコードの場合はテーブル名を使用等）を使用、または復帰するには、＜値1＞に空の文字列を渡します。

警告：新しい名前か空の文字列が渡されるまで、この命令文で指定した名前がセッションの印刷文書全体に対して使用されます。

このコマンドを使用して設定を行うと、4Dアプリケーション全体のセッションの間中、そのプリントオプションが保持されます。**PRINT SELECTION**、**PRINT LABEL**、**PRINT RECORD**、**Print form**、**QR REPORT**コマンドおよび「デザイン」モードを含めた4th Dimensionの印刷全般に対して、この設定が使用されます。

注：SET PRINT OPTIONコマンドを用いて設定したプリントオプションがリセットされないように、PRINT SELECTION、PRINT LABEL、PRINT RECORDおよび PAGE BREAKコマンドでは、任意の引数“>”を必ず使用してください。

通常、複数の印刷設定用コマンドを使用する場合の優先順位は次のようになります。

1. SET CURRENT PRINTER（リセット）

2. 印刷用のパラメータをすべて変更するPAGE SETUPやその他のコマンド

3. SET PRINT OPTION（印刷オプションの個別変更）

＜オプション＞に渡した値が無効であるか、そのプリンタで＜オプション＞が利用できない場合、コマンドはエラーを返し（**ON ERR CALL**コマンドでインストールされたエラー管理メソッドを用いて、このエラーを阻止できます）、オプションの現在の値がそのまま保持されます。コマンドが正常に終了すると、システム変数OKには1が代入され、それ以外の場合には0が代入されます。

参照：

GET PRINT OPTION、PRINT OPTION VALUES

GET PRINT OPTION

GET PRINT OPTION (オプション; 値1{; 値2})

引数	タイプ		説明
オプション	倍長整数	→	オプション番号
値1	倍長整数¦文字列	←	オプションの値1
値2	倍長整数	←	オプションの値2

GET PRINT OPTIONコマンドは、プリント＜オプション＞の現在の値を返します。

引数＜オプション＞により、取得するオプションを指定することができます。値を渡すか、または「Print options」テーマ内にある次の定義済定数のいずれかを渡すことができます。

定数	タイプ	値
Paper option	倍長整数	1
Orientation option	倍長整数	2
Scale option	倍長整数	3
Number of copies option	倍長整数	4
Paper source option	倍長整数	5
Color option	倍長整数	8
Destination option	倍長整数	9
Double sided option	倍長整数	11
Spooler document name option	倍長整数	12

このコマンドは、指定した＜オプション＞の現在の値を引数＜値1＞と（任意の）＜値2＞に返します。オプションおよび考えられる値に関する詳細は、**SET PRINT OPTION**コマンドの説明を参照してください。次に示す**GET PRINT OPTION**コマンド特定の機能に注意してください。

■ option=1（paper option）：＜値2＞を省略した場合、＜値1＞には現在の用紙名が返されます。＜値2＞を渡した場合、このコマンドは用紙の幅と高さをそれぞれ＜値1＞と＜値2＞に返します。プリンタが提供するすべての用紙フォーマットの名前、高さ、幅を取得するには、**PRINT OPTION VALUES**コマンドを使用してください。

■ option=2（orientation option）：1（縦方向）または2（横方向）が返されます。異なる方向オプションが使用されている場合、＜値1＞には0が代入されます。

■ option=5（paper source option）：＜値1＞には使用する用紙トレイの（**PRINT OPTION VALUES**コマンドから返されたトレイ配列内の）インデックスが返されます（＜値2＞は必ず省略してください）。

注：このオプションはWindows上でのみ使用できます。

■ option=8（color option）：＜値1＞にはカラー処理用のモードを表わすコードが返されます（1= 黒白（モノクロ）、2= カラー）。

注：このオプションは、Windows上でのみ使用できます。

■ option=9（destination option）：定義済リスト上に現在の値が存在しない場合、＜値1＞には-1が代入され、システム変数OKには1がセットされます。エラーが発生した場合、＜値1＞とシステム変数OKには0が代入されます。＜値1＞に1以外の定義済の値が代入されている場合、＜値2＞には印刷されたファイルのアクセスパスが納められます。

■ option=11（double sided option）：＜値1＞に、0（デフォルト値である標準か片面印刷）または1（両面）が返されます。

＜値1＞1の場合、＜値2＞には0=左とじ（デフォルト）または1= 上とじのいずれかの値が返されます。

注：このオプションは、Windows上でのみ使用できます。

■ option=12（spooler document name option）：事前に定義されている場合、＜値1＞には現在の印刷文書の名前が返されます。事前定義されていない場合には、空の文字列が返されます。

コマンドが正常に終了すると、システム変数OKには1が代入され、それ以外の場合には0が代入されます。

PRINT OPTION VALUES

PRINT OPTION VALUES (オプション; 名前配列{; 情報1配列; 情報2配列})

引数	タイプ		説明
オプション	倍長整数	→	オプション番号
名前配列	テキスト配列	←	値の名称
情報1配列	倍長整数配列	←	オプションの値（1）
情報2配列	倍長整数配列	←	オプションの値（2）

PRINT OPTION VALUESコマンドは、指定したプリント＜オプション＞に対して使用できる値の名称リストを引数＜名前配列＞に返します。オプションとして、＜情報1配列＞と＜情報2配列＞内に各値に関する情報を取得することができます。

引数＜オプション＞を使用して、取得するオプションを指定することができます。必ず、「Print options」テーマ内にある次の定数のいずれかを渡してください（値名の一覧を返すことができるオプション）。

定数	タイプ	値
Paper	倍長整数	1
Paper source	倍長整数	5

コマンドの実行後、配列＜名前配列＞ならびに配列＜情報1配列＞と＜情報2配列＞（適用可能な場合）には、使用できる値の名称と情報が代入されます。

引数＜オプション＞に値1（paper option）を渡した場合、コマンドから次の情報が返されます。

■ ＜名前配列＞には、使用できる用紙フォーマットの名前。

■ ＜情報1配列＞には、各用紙フォーマットの高さ。

■ ＜情報2配列＞には、各用紙フォーマットの幅。

注：この情報を取得するには、プリンタドライバがそのプリンタの有効なPPD（PostScript Printer Description）へアクセスできなくてはなりません。

SET PRINT OPTIONコマンドを使用して特定の用紙フォーマットを適用するために、＜名前配列＞のいずれかの値を渡すか、または＜情報1配列＞と＜情報2配列＞の対応する値を渡すことができます。

引数＜オプション＞に値5（paper source option）を渡した場合、このコマンドからは利用可能な別のトレイ名が＜名前配列＞に返され、＜情報1配列＞には内部的なWindowsのID番号が返されます（＜情報2配列＞は空のままです）。

配列内の値の順序は、プリンタドライバにより決まります。**SET PRINT OPTION**コマンドを使用してトレイを指定するには、目的の要素のインデックスを＜名前配列＞または＜情報1配列＞に設定された通りに渡さなくてはなりません。

注：このオプションは、Windows上でのみ使用できます。

各種プリントオプションに関する詳細は、**SET PRINT OPTION**コマンドと**GET PRINT OPTION**コマンドの説明を参照してください。

これらのコマンドから返される情報はすべて、OSにより提供されます。特定のオプションについての詳細は、お使いのシステムのドキュメントを参照してください。

その他新規コマンド

MULTI SORT ARRAY（「配列」テーマ）

MULTI SORT ARRAY (ポインタ配列名; ソート配列名)

引数	タイプ		説明
ポインタ配列名	ポインタ配列	→	配列ポインタの配列
ソート配列名	倍長整数配列	→	並び替え順の配列（1= 昇順並び替え、-1= 降順並び替え、0= 前の並び替えとの同期化）

MULTI SORT ARRAYコマンドにより、一連の配列に対してマルチレベルソートを実行することができます。この機能は、フォーム内のグループ化したスクロールエリアのコンテキストにおいて特に役立ちます。また、汎用的な開発では非常に有効です（例えば、あらゆるタイプの配列を並び替える汎用メソッドを作成したり、汎用的な**SORT ARRAY**コマンドに相当するものを作成する場合）。

引数＜ポインタ配列名＞には、配列ポインタの配列名を指定します。この配列の各要素は、並び替える配列を示すポインタです。＜ポインタ配列名＞に指定した配列ポインタの順に、並び替えが実行されます。

注：＜ポインタ配列名＞には、ローカル（$ptrArrayName）、プロセス（ptrArrayName）、インタープロセス（<>ptrArrayName）タイプのポインタの配列を指定することができます。これとは逆に、この配列の要素が指す対象は、プロセス配列またはインタープロセス配列でなくてはなりません。

引数＜ソート配列名＞には配列名を渡し、この配列の各要素は対応するポインタ配列要素の並び替え順を示します。

- 1=降順並び替え
- 0=配列は並び替え条件として使用されませんが、他の並び替えに応じて並び替えられます。
- 1=昇順並び替え

注：ポインタタイプやピクチャタイプの配列を並び替えることはできません。二次元配列（つまり、a2DArray{$vlThisElement}）の要素を並び替えることができますが、二次元配列そのもの（つまり、a2DArray）を並び替えることはできません。

配列＜ポインタ配列名＞の各要素に対して、対応する配列＜ソート配列名＞の要素が存在していなければなりません。したがって、必ずこの2つの配列の要素数は全く同じになります。

▼次の例題は、4つの配列を作成し、都市（昇順）と会社（降順）で並び替えます。最後の2つの配列、names_Array と telNum_Array は、前の並び替え条件に応じて同期化されます。

```
SELECT ALL([Employees])
SELECTION TO ARRAY([Employees]City;cities;[Employees]Company;companies;
                   [Employees]Name;names;[Employees]TelNum;telNums)
ARRAY POINTER (pointers_Array;4)
ARRAY LONGINT(sorts_Array;4)
pointers_Array{1}:=->cities
sorts_Array{1}:=1
pointers_Array{2}:=->companies
sorts_Array{2}:=-1
pointers_Array{3}:=->names
sorts_Array{3}:=0
pointers_Array{4}:=->telNums
sorts_Array{4}:=0
MULTI SORT ARRAY(pointers_Array;sorts_Array)
```

3番目の並び替え条件としてnames配列を使用したい場合には、sorts_Array{3} 要素に値“1”を割り当てる必要があります。または、都市だけを条件として配列を並び替えたい場合は、sorts_Array{2}、sorts_Array{3}、sorts_Array{4}の要素に値“0”を割り当てます。この方法で、**SORT ARRAY**(cities; companies;names;telNums;>) と同じ結果を得ることができます。

参照：

SELECTION TO ARRAY、ORDER BY、SORT ARRAY

BEST OBJECT SIZE（「オブジェクトプロパティ」テーマ）

BEST OBJECT SIZE ({*;}オブジェクト; 最適幅; 最適高さ{; 最大幅})

引数	タイプ		説明
*	*	→	指定した場合、オブジェクトは オブジェクトの名前（文字列） 省略した場合は、オブジェクトは変数
オブジェクト	オブジェクト	→	オブジェクト名（*を指定した場合） または、フィールドまたは変数 (*を省略した場合)
最適幅	倍長整数	←	オブジェクトの最適な幅
最適高さ	倍長整数	←	オブジェクトの最適な高さ
最大幅	倍長整数	→	オブジェクトの最大幅

BEST OBJECT SIZEコマンドは、引数＜*＞と＜オブジェクト＞で指定されたフォームオブジェクトの“最適な”幅と高さを、引数＜最適幅＞と＜最適高さ＞に納めて返します。これらの値は、ピクセルで表わされます。

現在のコンテンツが制限内にすべて含まれるように、返される最適値はオブジェクトの最小サイズを表わします。もちろん、これらの値はテキストを含むオブジェクトに関してのみ意味を持ちます。この計算には、フォント、フォントサイズ、フォントスタイルおよびオブジェクト内容が考慮されます。さらに、ハイフンや改行も考慮されます。指定された＜オブジェクト＞が空の場合、＜最適幅＞には0が返されます。

返されるサイズは、オブジェクトの周囲に貼り付けられたグラフィックフレームやスクロールバーを計算に入れていません。画面上のオブジェクトの実際のサイズを取得するには、これらの要素の幅を加算する必要があります。

任意の引数＜最大幅＞により、オブジェクトに最大幅を割り当てることができます。オブジェクトの最適な幅がこの値よりも大きい場合、**BEST OBJECT SIZE**コマンドは＜最大幅＞を＜最適幅＞に代入して返し、この結果として最適な高さを大きくします。

任意の引数＜*＞を渡すと、引数＜オブジェクト＞がオブジェクト名（文字列）であることを表わします。引数＜*＞を渡さない場合、＜オブジェクト＞はフィールドまたは変数となります。この場合には、文字列ではなくフィールドまたは変数への参照（オブジェクトタイプのみ）を渡してください。

このコマンドは、次のオブジェクトを処理することができます。

■ スタティックテキストエリア

■ 参照するフォームに挿入されたテキスト

■ 次のタイプのフィールドや変数：文字、テキスト、実数、整数、倍長整数、日付、時間、ブール（チェックボックスとラジオボタン）

■ ボタン

この他のオブジェクトタイプ（グループエリア、タブ、矩形、直線、円／楕円、プラグインエリア等）に対して、**BEST OBJECT SIZE**コマンドは現在のオブジェクトサイズ（「フォーム」エディタや**MOVE OBJECT**コマンドで指定）を返します。

▼ 後述の**SET PRINT MARKER**コマンドの例題を参照してください。

Is data file locked（「4D 環境」テーマ）

Is data file locked→ブール

引数	タイプ		説明
このコマンドに引数はありません。			
戻り値	ブール	←	True= ファイル／セグメントがロックされている False= ファイル／セグメントはロックされていない

Is data file locked関数は、開かれているデータベースのデータファイル、または少なくとも1つのセグメントがロックされている（つまり、書き込み禁止）場合にTrue（真）を返します。

例えば、「On Startup」データベースメソッドにおいてこのコマンドを使用すると、ロックされたデータファイルを誤ってオープンする危険性を回避することができます。ロックされたデータファイルの処理に関する詳細は、前述の「ロックされたデータファイルの検出」の節を参照してください。

▼ 例題

```
If(Is data file locked)
    ALERT("データファイルはロックされています。データベースを開くことができません。")
    QUIT 4D
End if
```

変更されたコマンド

変更された構文要素はイタリック体で表わされています。

MENU BAR（「メニュー」テーマ）

MENU BAR (メニューバー番号 | メニューバー名 {; プロセス}{; *})

引数	タイプ		説明
メニューバー番号 ¦	数値 ¦	→	メニューバーの番号または名前
メニューバー名	文字列		
プロセス	数値	→	プロセス参照番号
*	*	←	既存のメニューバーの状態保存

バージョン2003の4th Dimensionより、「カスタム」メニューのメニューバーに独自の名前を割り当てられるようになります（前述の「メニューバーの命名」を参照）。

この結果として、**MENU BAR**コマンドは1番目の引数としてメニューバー番号または名前のいずれかを受け入れるようになりました。

したがって、次のシンタックスのいずれかを使用することができます。

```
MENU BAR(3)
```

または、

```
MENU BAR("formulaBar")
```

注：メニューバー名には、31桁までのユニークな名前を指定することができます。

SET DATABASE PARAMETER（「ストラクチャアクセス」テーマ）

SET DATABASE PARAMETER ({テーブル; }セレクタ; 値)

SET DATABASE PARAMETERコマンドが受け入れるセレクタが増え、4D ClientのWebサーバの操作や新しいデータ書き込みモード、プログラムからのWebサービスの設定を行えるようになります。

新しいセレクタ定数

新しいセレクタは、次の定数を使用して指定することができます。

定数	タイプ	値
Client Minimum Web Process	倍長整数	19
Client Maximum Web Process	倍長整数	20
Client Max Web requests size	倍長整数	21
Client Port ID	倍長整数	22
Client IP Address to listen	倍長整数	23
Client Character set	倍長整数	24
Client Max Concurrent Web Proc	倍長整数	25
Cache writing mode	倍長整数	26
Maximum Web requests size	倍長整数	27

■ セレクタ = 19 からセレクタ = 25

■ 値：対応する 4D のセレクタと同じ（4th Dimension の『ランゲージリファレンス』マニュアルを参照）

■ 説明：これらのセレクタを使用して、Web サービスとして使用する 4D Client マシンの操作用パラメータを指定することができます。

これらのセレクタを用いて指定された値は、Web サービスとして使用するすべての 4D Client マシンに対して適用されます。特定の 4D Client マシンに対してのみ値を指定したい場合には、4D Client の「環境設定」ダイアログボックスを使用してください（詳細については、後述の「Web サービスとしての 4D Client」の節を参照）。

■ セレクタ = 26（Cache writing mode）

■ 値：0 または 1（0= 無効、1= 有効）

■ 説明：新しいキャッシュ書き込みモードを有効、または無効に設定します。デフォルトでは、この値は 1 になります（有効モード）。

バージョン 2003 の 4th Dimension より、キャッシュ書き込みモードが追加され、特に MacOS において 4D アプリケーションの処理速度が著しく向上します（詳細については、後述の「キャッシュ書き込み」の節を参照）。これはデフォルトとして有効に設定されています。

■ セレクタ = 27（Maximum Web requests size）

■ 値：500 000 から 2 147 483 648

■ 説明：Web サービスが処理を許可された受信 HTTP リクエストの最大数（バイト単位）。デフォルトでは、この値は 2,000,000、つまりほぼ 2MB となります。

最大値（2,147,483,648）を渡すと、実際上は制限がなくなります。

この制限を使用し、受信するリクエストが多すぎるためにWebサービスが限界に達してしまう危険性を回避します。リクエストがこの制限に達すると、4D Webサービスはリクエストを拒否します。

REPORT（「Printing」テーマ）

4th Dimension 2003の新しいクイックレポートエディタの導入にともない、今まで「Printing」テーマ内にあった**REPORT**コマンドは、**QR REPORT**という名前に変更され、「Quick Report」テーマ内に配置されるようになりました。また、シンタックスも変更されています（次の項を参照）。

QR REPORT（「Quick Report」テーマ）

QR REPORT ({テーブル; } ドキュメント{; *} {;ウィザード{; クエリ}})

引数	タイプ		説明
テーブル	テーブル	→	印刷するテーブル 省略した場合、デフォルトテーブル
ドキュメント	文字列	→	クイックレポートドキュメント
*	*	→	プリントダイアログボックスの表示 取り消し
ウィザード	ブール	→	True= ウィザードボタンの表示 Falseまたは省略 = 表示しない
クエリ	ブール	→	True= クエリボタンの表示 Falseまたは省略 = 表示しない

新しい2つの引数＜ウィザード＞と＜クエリ＞を用いて、このコマンドを使用して「クイックレポート」エディタを表示した際に、それぞれ対応するボタンをエディタ上に表示するかどうかを指定することができます。

ボタンを表示するにはTrueを渡し、ボタンを隠す場合にはFalseを渡します。デフォルトでは（これらの引数を省略した場合）、「ウィザードを開く」ボタンと「新規クエリ」ボタンが表示されません。

4D Engineを組み込んだファイルのデフォルト位置（MacOS）

MacOS上で、4D Engineをマージしたアプリケーションにより処理されるファイルは、デフォルトとしてソフトウェアパッケージと同じ階層で検索され、保存されるようになりました。このアプリケーションにより処理されるすべてのファイル（BLOB、読み込み／書き出し、ピクチャ等）はこの場所に置かれます。

ただし、**Structure file**関数は、4D Engineアプリケーションから実行された場合には特別な処理を行います。つまり、このコマンドはソフトウェアパッケージ内にあるストラクチャファイルのアクセスパスを返します。

4D Engineを使用し、ソフトウェアパッケージ自体のアクセスパスを取得したい場合には、**Application file**関数を利用することをお勧めします。方法としては、**Application type**関数を使用してアプリケーションタイプを調べた後、その結果に応じて**Structure file**関数または**Application type**関数を実行します。

「Open form window」テーマの定数

他の定数との混同を避けるため、「Open form window」テーマ内の4つの定数の名称が変更されました。

以前の4D定数	4D 2003定数
Modal dialog box	Modal form dialog box
Movable dialog box	Movable form dialog box
Palette window	Palette form window
Plain window	Standard form window

印刷の最適化

バージョン2003の4th Dimensionでは、印刷に関連する各種コマンドの機能が拡張され、カスタマイズしたレポートの作成を速やかに行えるようになりました。特に、**SET PRINT MARKER**、**PAGE BREAK**、**CANCEL**の各コマンドは、**Print form**関数により開始された印刷中でも動作するようになります。さらに、これらのコマンドは**BEST OBJECT SIZE**（「Object Properties」テーマ）コマンドと一緒に使用することができます。印刷に関する詳しい例題は、この後に記述されています。

注：「Printing」テーマにも新しいコマンドが追加され、機能強化されています（前述の「新しい印刷コマンド」の節を参照）。

SET PRINT MARKER

SET PRINT MARKER (マーカー番号; 位置{; * })

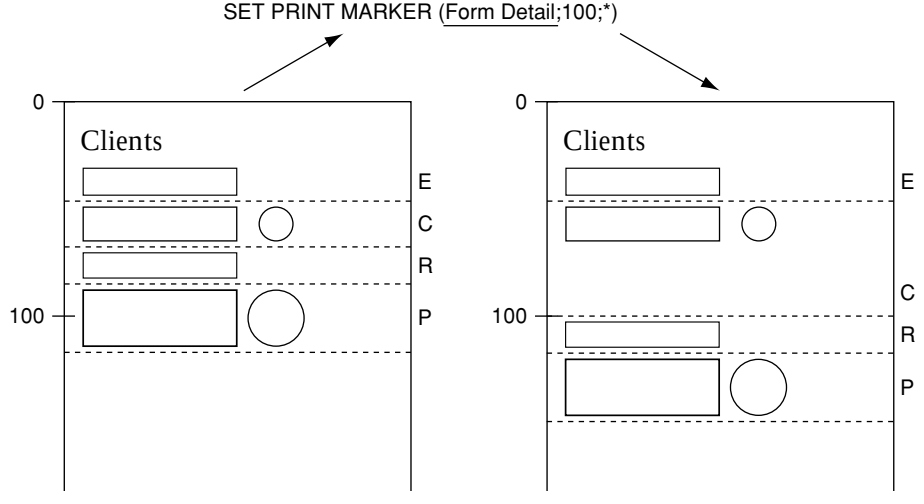
引数	タイプ		説明
マーカー番号	数値	→	マーカーの番号
位置	数値	→	マーカーの新しい位置
*	*	→	渡した場合、次のマーカーを移動する 省略した場合、次のマーカーを移動しない

■ **SET PRINT MARKER** コマンドは、「On Printing Detail」フォームイベント中に **Print form** 関数を使用した場合に呼び出せるようになりました。この操作により、カスタマイズしたレポートの印刷がスムーズに行えます（例題を参照）。

■ さらに、**SET PRINT MARKER** コマンドは、3番目の引数として * 記号を受け入れます。

この引数を渡すと、このコマンドの実行時に、＜マーカー番号＞で指定したマーカーより下側に位置するすべてのマーカーが、指定したマーカーと同じピクセル数だけ、同じ方向へ移動します。このマーカーより下側にあるエリア内のオブジェクトもすべて移動します。

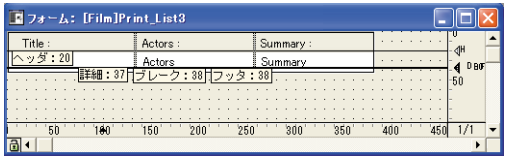
引数＜*＞を使用すると、後続の各マーカーの最初の位置より下側に＜マーカー番号＞で指定したマーカーを位置付けることができます。これら後続のマーカーも同時に移動します。



注：引数＜*＞を使用しない場合、移動するマーカを後続のマーカーよりも下側に位置付けることはできません。

▼ この例題を使用して、3つのカラムがある印刷を実行することができます。各行の高さは、フィールド内容に応じて実行中に計算されます。

印刷に使用する出力フォームは次の通りです。



注：このフォームに対して、「On Printing Detail」フォームイベントが選択されています（印刷するエリアに関係なく、Print form関数はこのタイプのフォームイベントだけを生成する点に留意してください）。

レコードごとに、“Actors”または“Summary”カラムの内容に応じて行の高さを調整しなくてはなりません。目的とする結果を次に示します。

Title	Actors	Summary
Braveheart	Mel Gibson, Sophie Marceau, Patrick McGoohan, Catherine McCormack	Mel Gibson stars on both sides of the as lead role plus directing and producing this detailed saga of fierce combat, tender lo all that's precious: freedom. In an emoti performance, Gibson is William Wallace, used the steel of his blade and the fire i his countrymen to liberation. Filled with s spectacle, Braveheart is a tumultuous ti some alive, OOthe most sumptuous and epic since Lawrence Of Arabia.OO --Rol Magazine
Terminator 2	Arnold Schwarzenegger, Linda Hamilton, Edward Furlong, Robert Patrick	Arnold Schwarzenegger returns as the Te explosive adventure spectacle. His mis Connor, the boy destined to lead the fre future. His opponent: the T-1000, the mo ever created, sent back through time to k ally: Sarah Connor, John's mother, a wo seemingly go unheeded by a world cement holocaust she knows is inevitable. T2 is stunts and astounding special effects,bl and emotional human story. Includes a booklet and limited edition aluminum sta
Scary Movies	Carmen Electra, Marlon Wayans, John Abrahams,Shannon Elizabeth, Shawn Wayans, Cheri Oteri	The hilarious, must-see comedy smash Electra (TVOs Baywatch), Marlon Wayan Abrahams (Boiler Room, The Faculty) an hottest young stars in a wickedly funny most popular horror movies! A familiar-l teenagers find themselves being stalked vaguely recognizable masked killer! As pile up and the laughs pile on, none of y movies escape the razor-sharp satire of t funny parody! With Shannon Elizabeth, Cheri Oteri adding sidesplitting performa to fear in this scary movie!Useless you laugh much!
Leon: The Professional (International Version)	Jean Reno, Gary Oldman, Natalie Portman, Danny Aiello	OOJean Reno, Gary Oldman, Natalie Po Aiello star in Leon, a go-for-broke thriller assassin whose work becomes dangerou himself a stranger, the mysterious Leon i man. When his next-door neighbors are becomes the unwilling guardian of the fa 12-year-old Mathilda. But Mathilda doesn she wants revenge. Training her in the de trade, Leon helps her track the psychotic her family. From the electrifying opening Leon is a nonstop crescendo of action, suspense. This uncut version contains 2 footage deemed too explicit for the Ame
The 13th Warrior	Antonio Banderas, Neil Marlin, Asli gm 'Bear' Fisi,Clive Russell, Daniel Southern, Oliver Reinhall, Sven Walter, Diana Vesses, Dennie Storhji, Vladimir Kulich, Omar Sharif, Andrew T. Anderssen, Richard Brammer, Tony Curran, Nicoka Hauserman	Antonio Banderas (OOThe Mask of Zoro enormous talent to the film from the firs Hand(0) and the OOThomas Crown Affai action and adventure with suspense! An Ambassador, Ahmed (Banderas) crosses hardened warriors that have come to the attacked by ferocious creatures. As old the fighters that they will surely fail eith warrior. Ahmed has no other choice but t them destroy the mysterious enemy. S enjoy this movie!

印刷用のプロジェクトメソッドは次の通りです。

```
C_LONGINT(vLprint_height;$vLheight;vLprinted_height)
C_STRING(31;vSprint_area)
PAGE SETUP([Film];"Print_List3")
GET PRINTABLE AREA(vLprint_height)
vLprinted_height:=0
SELECT ALL([Film])
```

```
vSprint_area:="Header" `ヘッダエリアの印刷
$vLheight:=Print form([Film];"Print_List3";Form Header)
$vLheight:=21`Fixed height
vLprinted_height:=vLprinted_height+$vLheight

While(Not(End selection([Film])))
    vSprint_area:="Detail" `詳細エリアの印刷
    $vLheight:=Print form([Film];"Print_List3";Form Detail)
    `フォームメソッドにおいて詳細行の計算が実行される
    vLprinted_height:=vLprinted_height+$vLheight
    If(OK=0) `フォームメソッドでCANCELが実行された
        PAGE BREAK
        vLprinted_height:=0
        vSprint_area:="Header" `ヘッダエリアの再印刷
        $vLheight:=Print form([Film];"Print_List3";Form Header)
        $vLheight:=21
        vLprinted_height:=vLprinted_height+$vLheight
        vSprint_area:="Detail"
        $vLheight:=Print form([Film];"Print_List3";Form Detail)
        vLprinted_height:=vLprinted_height+$vLheight
    End if
    NEXT RECORD([Film])
End while
PAGE BREAK `最後のページが印刷されたことを確認
```

フォームメソッドPrint_List3は次の通りです。

```
C_LONGINT($l;$t;$r;$b;$fixed_wdth;$exact_hght;$l1;$t1;$r1;$b1)
C_LONGINT($final_pos;$i)
C_LONGINT($detail_pos;$header_pos;$hght_to_print;$hght_remaining)
Case of
    \ (vSprint_area="Detail") `詳細行の印刷が進行中
        GET OBJECT RECT([Film]Actors;$l;$t;$r;$b)
        $fixed_wdth:=$r-$l`Actorsテキストフィールドサイズの計算
        $exact_hght:=$b-$t
        BEST OBJECT SIZE([Film]Actors;$wdth;$hght;$fixed_wdth)
            `内容に応じたフィールドの最適サイズ
        $movement:=$hght-$exact_hght
        GET OBJECT RECT([Film]Summary;$l1;$t1;$r1;$b1)
        $fixed_wdth1:=$r1-$l1`Summaryテキストフィールドサイズの計算
        $exact_hght1:=$b1-$t1
        BEST OBJECT SIZE([Film]Summary;$wdth1;$hght1;$fixed_wdth1)
            `内容に応じたフィールドの最適サイズ
```

```
$movement1:=$hght1-$exact_hght1
If($movement1>$movement)
        `最も高さがあるフィールドを決定する
        $movement:=$movement1
End if

If($movement>0)
        $position:=Get print marker(Form Detail)
        $final_pos:=$position+$movement
                `Detailマーカーと、これに続くマーカーを移動
        SET PRINT MARKER(Form Detail;$final_pos;*)
                `テキストエリアのサイズ変更
        MOVE OBJECT([Film]Actors;$l;$t;$r;$hght+$t;*)
        MOVE OBJECT([Film]Summary;$l1;$t1;$r1;$hght1+$t1;*)
                `分割ラインのサイズ変更
        GET OBJECT RECT(*;"H1Line";$l;$t;$r;$b)
        MOVE OBJECT(*;"H1Line";$l;$final_pos-1;$r;$final_pos;*)
        Loop($i;1;4;1)
                GET OBJECT RECT(*;"VLine"+文字列($i);$l;$t;$r;$b)
                MOVE OBJECT(*;"VLine"+文字列($i);$l;$t;$r;
                                                        $final_pos;*)
        End of loop
End if

        `利用できるスペースの計算
$detail_pos:=Get print marker(Form Detail)
$header_pos:=Get print marker(Form Header)
$hght_to_print:=$detail_pos-$header_pos
$hght_remaining:=printing_height-vLprinted_height
If($hght_remaining<$hght_to_print) `不十分な高さ
        CANCEL `フォームを次ページに移動
End if
End case
```

PAGE BREAK

PAGE BREAK {(* | >)}

引数	タイプ		説明
*¦>		→	*：Print formで開始した印刷ジョブをキャンセルする、または >：印刷ジョブを強制実行する

PAGE BREAKコマンドは、**Print form**関数の使用に伴って「On Printing Detail」フォームイベント中に呼び出せるようになります。

Print form関数を使用してフォームを印刷する前に、ページブレークを挿入しなくては成りません。

▼ 前述の**SET PRINT MARKER**コマンドの例題を参照してください。

CANCEL（「入力制御」テーマ）

CANCEL

引数	タイプ	説明

このコマンドに引数はありません。

バージョン2003の4Dより、「On Printing Detail」フォームイベント中に**Print form**関数を使用する際、**CANCEL**コマンドを使用できるようになります。

この状況において、**CANCEL**コマンドは実行直前の印刷を一時中断し、次のページで印刷を再開します。この仕組みを利用して、スペースが不足している場合やページブレークが必要な場合のフォーム印刷を管理することができます。

注：この操作は、PAGE BREAK(*)コマンドとは異なります。PAGE BREAK(*)コマンドは、印刷待ちであるすべてのフォームをキャンセルします。

この場合、**CANCEL**コマンドを呼び出すと、システム変数OKには0が代入されます。

▼ 前述の**SET PRINT MARKER**コマンドの例題を参照してください。

4D Chartの新しいコマンド

4D Chartの「CT Area Control」テーマに新しく2つのコマンドが追加されました。これらのコマンドを使用して、4D Serverの使用時に、4D Chart文書のテンプレートの保存や読み込みを行う場所を管理することができます。

注：以前のバージョンの4Dでは、これらの機能は4D Customizer Plusユーティリティによって処理されていました。

CT SET AREA PROPERTY

CT SET AREA PROPERTY (エリア; プロパティ; 値)

引数	タイプ		説明
エリア	倍長整数	→	4D Chartエリア
プロパティ	整数	→	プロパティの番号
値	整数	→	プロパティの値

CT SET AREA PROPERTYコマンドを使用し、カレントセッションに関して、4D Chart＜エリア＞の＜プロパティ＞の＜値＞を変更することができます。

＜エリア＞に-1を渡すと、**CT SET AREA PROPERTY**コマンドは、セッション中に続けてロードされたすべての4D Chartエリアに対して適用されます。この場合、「On Startup」データベースメソッド内でこのコマンドを使用することをお勧めします。

次のプロパティを変更することができます。

番号	プロパティ	値	意味
0	クライアント／サーバモードでテンプレートを保存	0	クライアント上
		1	サーバ上
1	クライアント／サーバモードでテンプレートをロード	0	クライアントよりロード
		1	サーバよりロード

デフォルトとして、テンプレートはサーバマシン上で保存され、読み込まれます。

参照：

CT GET AREA PROPERTY

CT GET AREA PROPERTY

CT GET AREA PROPERTY (エリア; プロパティ; 値)

引数	タイプ		説明
エリア	倍長整数	→	4D Chart エリア
プロパティ	整数	→	プロパティの番号
値	整数	→	プロパティの値

CT GET AREA PROPERTY コマンドを使用し、4D Chart ＜エリア＞の＜プロパティ＞の現在の＜値＞を取得することができます。

次のプロパティを使用することができます。

番号	プロパティ	値	意味
0	クライアント／サーバモードでテンプレートを保存	0	クライアント上
		1	サーバ上
1	クライアント／サーバモードでテンプレートをロード	0	クライアントよりロード
		1	サーバよりロード

デフォルトとして、テンプレートはサーバマシン上で保存され、読み込まれます。

参照：

CT SET AREA PROPERTY

4th Dimension 2003に統合されたWebサーバにより、この章で説明する次のような新しい機能が提供されます。

- **Webサーバとしての4D Client**：4D Server 2003を使用すると、各4D Clientマシンは Webサーバとして動作することができます。
- **新しい設定**：「4DACTIONで利用可能」オプションを使用すると、4D Webサーバのセキュリティを向上することができます。

  さらに、新規データベースに対する4D Webサーバのデフォルトパラメータが変更され、非コンテキストモードでの起動やWebフォルダ内へのデフォルトホームページの作成が可能になります。

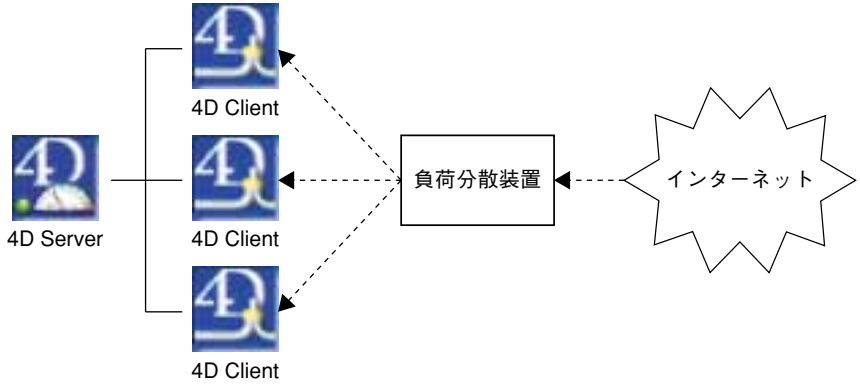
Webサーバとしての4D Client

4D Server 2003を使用すると、任意の4D ClientマシンをWebサーバに変えることができます。この新しい機能により、より優れたパフォーマンスとセキュリティを備えるWeb情報システムを構築することができます。

この新しいアーキテクチャにより、大規模な開発が可能になり、特に次の事柄が実現します。

- 例えば、リクエストの発信元に応じて、同一データからさまざまなビューを作成できます。社内ネットワークにおいて、プロテクトされた4D Client Webサーバがイントラネット上のリクエストを処理し、ファイヤウォール上にある別の4D Client Webサーバがインターネット上のリクエストを処理することができます。
- 障害対策済のWebサービスの構築：4D Webサイトは、2台以上の4D Clientマシン上にミラー処理されます。ある4D Client Webサーバがダウンすると、もう一方が処理を引き継ぎます。
- 4D Webサーバのパフォーマンスを最適化するためのロードバランシング（負荷分散）システムの構築：各4D Client Webサーバ上にインストールされるWebサイトミラーを使用し、現時点のそれぞれの負荷に基づいて負荷分散装置（ロードバランサー：ハードウェアまたはソフトウェア）が各クライアントマシンにリクエストを送信します。

負荷分散装置の設定例



■ 異なる4D Client Webサーバ間でのタスク分配：ある4D Client WebサーバがSOAPリクエストを処理し、別のサーバが標準的なリクエストを処理する、など。

一般原則

通常、4D Client Webサーバの運用方法は、非コンテキストモードでの4th Dimensionや4D ServerのWebサーバと同じです。ただし、4D Client Webサーバでは、コンテキストモードがサポートされません。

各4D Clientマシンは、スタティックあるいはセミダイナミックな独自のWebページを保持するスタンドアロンのWebサーバのように動作します。しかしながら、4D Serverデータベースに接続する4D Client Webサーバはすべて、同じデータ（サーバ上に保存された）を共有します。

4D Webサーバの基本的な仕組みが4D Clientでも同じような方法で使用されています。つまり、Webサービスの開始や終了が随時可能、「On Web Authentication」および「On Web Connection」メソッドの実行、4Dタグおよび URLの管理、***COMPILER_WEB***プロジェクトメソッドの呼び出し、SSL接続の処理などといった仕組みです。それでもやはり、4D Client Webサーバの管理には、ある程度の適応化や追加オプションが必要になります。この事柄については、次の節で説明します。

同様に、「Webサーバ」テーマのランゲージコマンドの動作は、4th Dimension、4D Server、4D Clientのいずれでコマンドが実行されても通常は同じです。重要なのは、コマンドはそれが実行されるマシンのWebサイトに対して適用されるという点です。

したがって、**Execute on server**や**EXECUTE ON CLIENT**コマンドを使用して、これを管理しなくてはなりません。しかし、4D Client Webサーバを使用する場合は、一部のコマンドの動作が変わります（後述の「ランゲージコマンド」の節を参照してください）。

クライアントWebライセンス

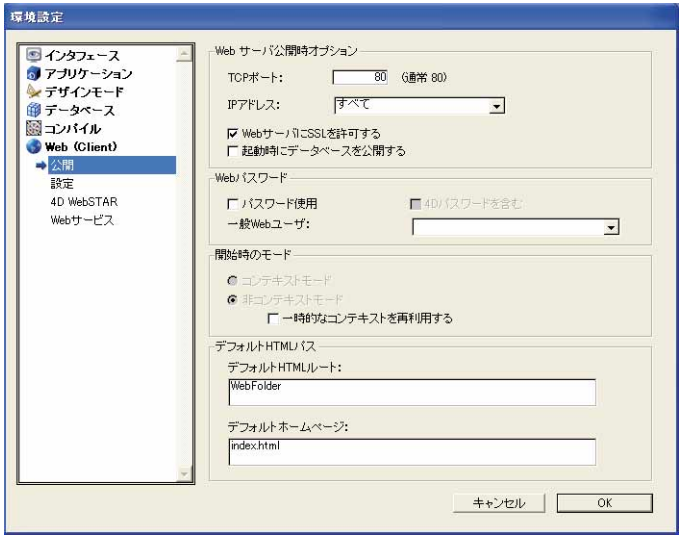
4D ClientマシンをWebサーバとして使用するには、クライアントWebライセンスと呼ばれる特別なライセンスが必要です。このライセンスは対応する4D Serverで登録しなくてはなりません。お使いのクライアントWebライセンスに組み込まれた接続数は、Webサーバとして同時に公開できる4D Clientマシンの数を表わします。

適切なライセンスが登録されていない場合、4D Client Webサーバはデモモードで動作します（1時間の使用が可能）。

4D Client Webサーバの設定

この節では、4D Clientにより公開されたWebサーバの設定に関する特定の機能について説明します。特に、アプリケーションの「環境設定」ダイアログボックスの使用について詳しく述べています（第2章の「データベースのオープンと作成」の節を参照）。

4D Clientの「環境設定」ダイアログボックス



通常、このダイアログボックスで定義されたパラメータは、各4D Clientマシンに対して個別に適用されます。**SET DATABASE PARAMETER**コマンドを使用して同じパラメータを定義した場合には、4D Serverマシン上に公開されたサイトに対してのみ適用される点に注意してください（後述の「SET DATABASE PARAMETERとGet database parameter」の節を参照）

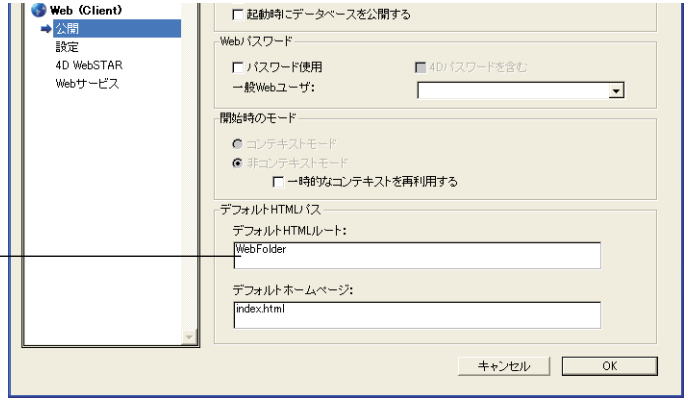
Web フォルダ

4D Client Webサイトに必要なファイルはすべて、クライアントマシンのディスク上にあるHTMLルートフォルダ内に必ず配置してください。

このフォルダには、4D Clientが公開するWebサイトに対応するスタティックページやダイナミックページ、ピクチャ等を手動で配置しなくてはなりません。4D Client自体が、このフォルダ内容の変更や管理を行うことはありません。

デフォルトとして、4D Clientはこのフォルダが存在しなければフォルダを作成します。フォルダには「WebFolder」という名前が付けられ、4D Clientの「.exe」ファイル（Windows）またはソフトウェアパッケージ（MacOS）と同じ階層に自動的に作成されます。

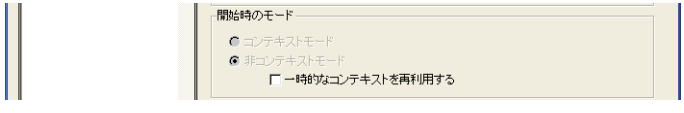
このフォルダの移動やリネームを行ったり、別のフォルダを使用したい場合には、4D Clientアプリケーションの「環境設定」において新しいフォルダを指定しなくてはなりません（「Web」テーマ、「公開」ページ）。



HTMLルートフォルダの場所

開始モード

4D Client Webサーバは、コンテキストモードでのWeb公開をサポートしていません。したがって、「環境設定」ダイアログボックスでWebサーバの開始モードを指定することはできません。



一時的なコンテキストを再利用する

この新しいオプション（4D Clientの「環境設定」固有のオプション）は、前回のWebリクエストの処理で作成されたWebプロセスを再利用することにより、4D ClientのWebサーバ操作を最適化するために使用します。

実際のところ、4D ClientのWebサーバでは、各Webリクエスト処理のために特定のWebプロセスが必要になります。つまり、必要に応じてこのプロセスが4D Serverマシンに接続し、データとデータベースエンジンにアクセスします。この後、独自の変数やセレクション等を使用して一時的なコンテキストが生成されます。リクエストの処理が終わると、このプロセスは終了します。

「一時的なコンテキストを再利用する」オプションを選択すると、4Dは4D Client上で作成された特定のWebプロセスを保持しておき、次回のリクエストでこのプロセスを再利用します。プロセス作成段階が省かれることにより、Webサーバのパフォーマンスが向上します。

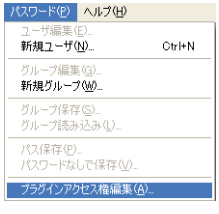
その代わりこの場合は、誤った結果を取得しないように、4Dメソッドを使用して意識的に各変数を初期化しなくてはなりません。同様に、前回のリクエスト中に定義されたカレントセレクションやレコードを消去する必要があります。

公開の認可

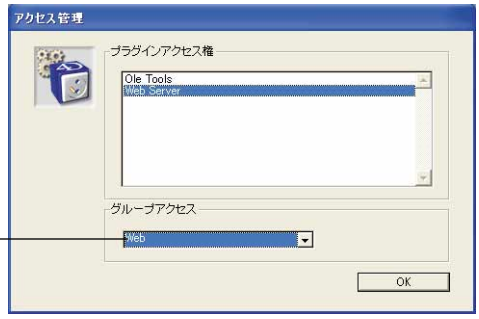
デフォルトとして、すべての4D Client 2003マシンは、接続しているデータベースをWeb上に公開することができます。ただし、4Dのパスワードシステムを使用すると、各4D ClientのWeb公開を制御することができます。

クライアントのWebライセンスは、4D Serverからはプラグインランセンスとみなされます。したがって、プラグインと同じ方法で、Webサーバライセンスの使用権を特定グループのユーザに限定することができます。

これを行うには、4D Clientのパスワードウインドウを表示します（これらのパラメータを変更するための適切なアクセス権が必要です）。パスワードウインドウが前面に表示されたら、「パスワード」メニューの「プラグインアクセス権...」コマンドを選択します。



プラグインのアクセスを管理するダイアログボックスにおいて、「Webサーバ」の選択後、「グループアクセス」ポップアップメニューを用いてユーザグループを関連付けます。



“Web”グループに属するユーザだけが4D Client を Web サーバとして公開できる

接続管理

4D Client Webサーバにおいて、接続およびリクエストの管理は「On Web Authentication」と「On Web Connection」データベースメソッドを使用して行われます。

引数$1から$4に返される情報は、各4D Client Webサーバに特定のものです。特に、引数$4に返されるアドレスは、4D ClientマシンのIPアドレスに相当します。

引数$5と$6には、必要に応じてユーザが入力したユーザ名とパスワードが返されます。

4Dのパスワードテーブルを用いてサイトのユーザアカウントを管理している場合、4D Clientマシンにより公開されたすべてのサイトは同じユーザテーブルを共有するという点に注意してください。実際には、4D Serverアプリケーションがユーザやパスワードの検証を行います。

ランゲージコマンド

前述の「一般原則」の節で説明したように、4D Serverまたは4D Clientのいずれの Webサーバを使用しても、大半の4Dランゲージ Webコマンドを使用することができます。コマンドの実行場所（サーバまたはクライアント）によって、そのコマンドの有効範囲と適用されるサイトが決定します。

しかし、コマンドのなかにはコンテキストモード（4D Client Webサーバではサポートされない）でしか動作しないものがあります。したがって、これらのコマンドをクライアントマシンで使用することは禁止されています。

次の節では、4D Client Webサーバで4Dランゲージコマンドを使用した場合の特殊状況について説明します。

START WEB SERVERとSTOP WEB SERVER

これらのコマンドは、コマンドが実行されるマシンのWebサーバに対して適用されます。

SET HTML ROOT

このコマンドはコンテキストモードでのみ動作するため、クライアントマシン上では使用しないでください。

SET HOME PAGE

このコマンドは、コマンドが実行されるマシンのWebサイトのホームページを定義するために使用します。

Web Context

このコマンドは、4D Client上で実行されると常に“False（偽）”を返します。

SET DATABASE PARAMETERとGet database parameter

これらのコマンドは、4D Client Webサーバ特定の処理用パラメータの定義や読み込みに用いる新しいセレクタを受け付けます。データベースの「環境設定」ダイアログボックスを用いて定義するパラメータとは異なり、**SET DATABASE PARAMETER**コマンドを使用して定義する“クライアント”パラメータは、すべての4D Client Webサーバに対して一様に適用されるという点に注意してください。詳細については、前述の**SET DATABASE PARAMETER**コマンド（「ストラクチャアクセス」テーマ）の説明を参照してください。

新しいパラメータ

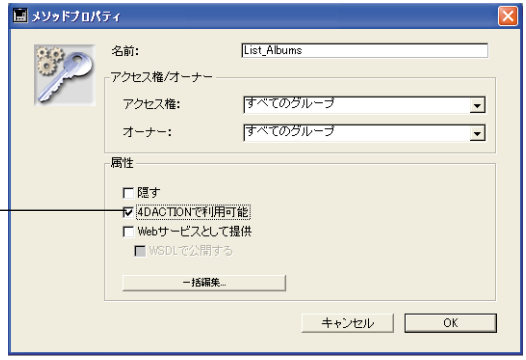
4D 2003のWebサーバには、次のパラメータが追加されています。

■「4DACTIONで利用可能」オプション。Web経由でのメソッドアクセスをロックするために使用します。

■ Webサイト用の新しいデフォルトパラメータ

4DACTIONで利用可能

プロジェクトメソッドの「メソッドプロパティ」ダイアログボックスに、新しいオプションである「4DACTIONで利用可能」が表示されます。



新しいセキュリティオプション

このオプションを使用して、4D Webサーバのセキュリティを強化することができます。プロジェクトメソッドに対してこのオプションが選択されていない場合、4th Dimensionメソッドの呼び出しに使用される特別なURLを含んだHTTPリクエスト（4DACTIONおよび4DMETHOD）を用いて、そのプロジェクトメソッドを実行することができません。

4th Dimension 2003におけるWebサービスの導入にともない、この新しいオプションが特に不可欠なものとなりました。実際、Webサービスが公開されると特定のメソッド名も公開されます。したがって、実行中にこれらのメソッドへのアクセスを防ぐために、このオプションが役立ちます。Webサービスに関する詳細は、前述の「Webサービス」の章を参照してください。

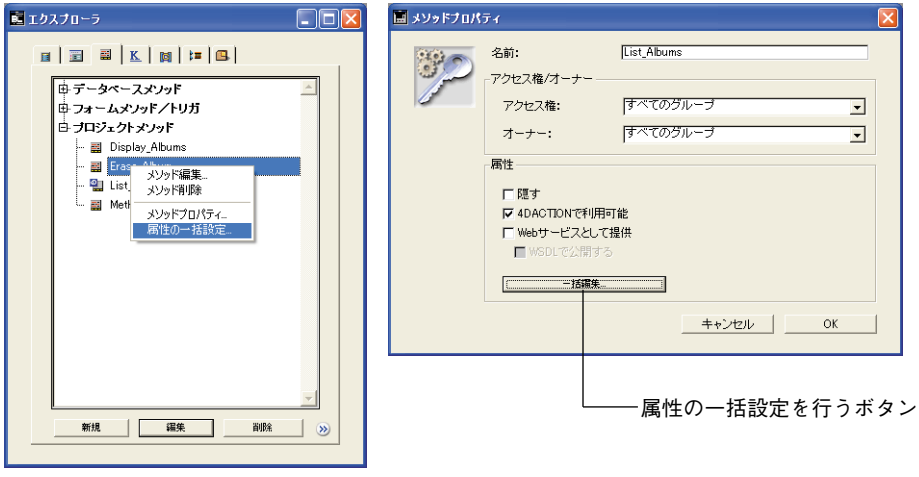
注：この新しいオプションを使用すると、「On Web Authentication」データベースメソッドの内容を簡略化することができます。特に、 $1 を用いてメソッド名に関するコールをフィルタリングする必要がなくなります。

4DACTIONを用いて利用できるプロジェクトメソッドには、特定のアイコンが使用されます。

4th Dimension 2003で作成されたデータベースに対し、デフォルトではこのオプションが選択されていません。4DACTIONや4DMETHOD Web URLを使用して実行されるメソッドに対しては、このオプションを個別に指定しなくてはなりません。

これとは逆に、互換性上の理由から、既存のデータベース（2003より以前のバージョンの4Dで作成）に対しては、このオプションが選択されています。デフォルトでは、特別な4DのURLを使用して、すべてのプロジェクトメソッドにアクセスすることができます。

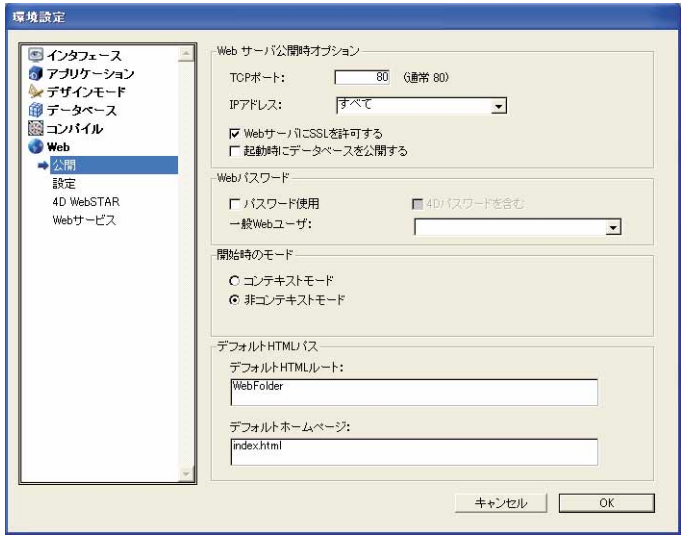
4th Dimension 2003 では、数多くのメソッドに対してこのオプションを選択する場合のために、ショートカットが提供されています。「エクスプローラ」の新しいコンテキストメニュー（Windows では右クリック、MacOS では Ctrl+ クリック）から「属性の一括設定...」を選択するか、あるいは「メソッドプロパティ」ウインドウの「一括編集...」ボタンをクリックします。



このコマンドによりダイアログボックスが表示され、ここで一連のメソッドの属性を一度に変更することができます。詳細については、前述の「メソッド属性の一括設定」の節を参照してください。

新しいデフォルトパラメータ

4th Dimension 2003 では、4D Web サーバのデフォルトパラメータ（新しく作成されたサイトに適用）が変更されました。



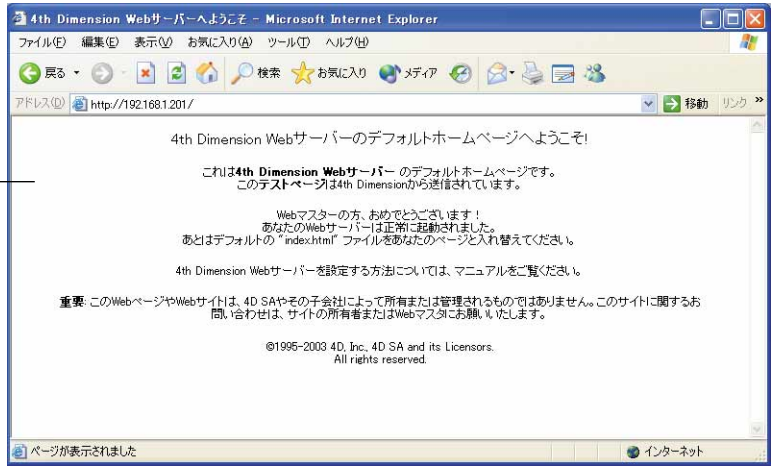
Web サーバのデフォルトの公開用パラメータ

今後、ユーザがデフォルトで設定されたパラメータを変更しない場合、それぞれの設定は次のようになります。

- Webサービスは非コンテキストモードで開始します。
- 「ユーザ」パスワードオプションは選択されません。
- 「新しいコンテキスト参照モードを使用」オプションが選択されます。
- Webサーバの初回起動時に、HTMLルートフォルダが作成されている場合、デフォルトホームページはこのフォルダに納められます。

デフォルトである“index.html”という名前のこのページを使用して、Webサービスへの接続を検証することができます。

4D Webサーバのデフォルトホームページ(index.html)



4Dアプリケーションのパフォーマンスや使いやすさを向上するため、バージョン2003の4th Dimensionには2種類の最適化が導入されています。

キャッシュ書き込み

4th Dimension 2003において、データベースキャッシュに保持するデータ（インデックスページ、レコード、削除マーカ）のディスクへの書き込みが最適化されました。4Dデータベースのパフォーマンスはめざましく向上しています。

このパフォーマンスの向上が特に顕著なのは、次の場合です。

■ MacOS上において

■ ディスクが断片化されている場合

■ 大量のデータがディスク上に書き込まれる場合

4th Dimension 2003では、デフォルトとしてこの新しいアルゴリズムがアクティブになっています。ただし、**SET DATABASE PARAMETER**コマンドを使用して、プログラムからこれを非アクティブにすることができます（前述の**SET DATABASE PARAMETER**コマンドを参照）。

マウスホイールのサポート（Windows）

4th Dimension 2003では、Windows上でのマウスホイールのサポート状況が向上しています。

大部分の4Dウインドウやダイアログボックスでマウスホイールを利用できるようになりました。特に、次のウインドウでの使用が可能になります。

■ メソッドエディタ

■ エクスプローラ

- 出力フォーム、サブフォーム、**DISPLAY SELECTION**と**MODIFY SELECTION**を用いて表示されるフォーム
- “スクロール可能な”4Dオブジェクト：スクロールエリア、階層リスト、グループ化された配列、テキストタイプのフィールドと変数、コンボボックス（展開時）、ドロップダウンリスト

マウスホイールは4Dプラグインでもサポートされます。

付録 A ： XML 用語集

XML 言語には数々の特別な用語や頭文字が使われています。この節では、4th Dimension 2003 のコマンドや関数で使用される主な XML の概念について説明します。

■ **属性**：XML のサブタグで、要素と関連付けられています。属性は常に名前と値を持ちます（下図を参照）。

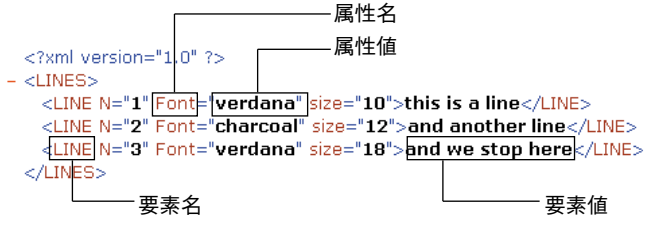
■ **整形式**：XML 文書が一般的な XML 規格に準拠している場合、パーサにより“整形式”であるものと宣言されます。

■ **DTD**：Document Type Declaration（文書型定義）の略。DTD には、XML が従わなくてはならない特定の規則とプロパティ一式が記録されています。これらの規約として、特に各タグの名前と内容、ならびにそのコンテキストが定義されています。各要素を形式的に定義することにより、XML 文書の整合性をチェックすることができます（その場合、文書は“妥当”であると宣言されます）。

DTD は、XML 文書内（内部 DTD）あるいは別の文書内（外部 DTD）に組み込まれます。DTD は必須ではないという点に注意してください。

■ **要素**：XML タグ。ある要素は常に名前と値を持ちます。オプションとして、要素には属性を含めることができます（図を参照）

要素と属性



■ **子**：XML 構造で他のすぐ下のレベルに位置する要素。

■ **親**：XML 構造で他のすぐ上のレベルに位置する要素。

■ **解析、パーサ**：有用な情報を取り出すために、構造化オブジェクトの内容を解析する処理。「XML」テーマのコマンドは、任意の XML オブジェクトの内容を解析するために使用されます。

■ ルート：XML構造の一番始めのレベルに位置する要素。

■ 要素参照：4DのXMLコマンドでXML構造を特定するために使用するXMLへの参照。この参照は、16進表記の8つのコード化文字、すなわち16桁で構成されます。

■ XML構造：構造化されたXMLオブジェクト。このオブジェクトは、文書、変数、要素のいずれでも構いません。

■ 妥当性検証：XML文書は、それが“整形式”であり、DTD仕様に準拠している場合、パーサにより“妥当”であるものと判断されます。

■ XML：eXtensible Markup Language（拡張可能なマーク付け言語）の略。電子化されたデータの変換を行う際の標準言語で、データならびにその構造を転送することができます。XML言語は、HTML言語を踏まえて、タグと特定のシンタックスの使用を基本としています。ただし、HTMLとは異なり、XML言語ではカスタマイズしたタグを定義することができます。

■ XSL：eXtensible Stylesheet Languageの略。XSL文書内容の処理や表示に使用するスタイルシートを定義する言語。

付録 B ：マクロ用 DTD

この付録では、4th Dimension 2003のマクロファイル（Macros.xml）のDTD（Document Type Declaration：文書型定義）を提供します。

このDTDは、次の目的のために使用することができます。

■ マクロファイル内で使用されるマーカーのXMLシンタックスや文法を正確に表示する。

■ XML規格に確実に準拠するように、カスタマイズしたマクロファイルの妥当性を検証する（XMLの妥当性検証については、前述の付録A「XML用語集」を参照してください）。

これを行うには、DTDをテキストファイルにコピーしなおし、例えば“Macros.dtd”のような名前をこのファイルに付けます。次に、XMLパーサーを使用します（例えば、4th Dimension 2003のXMLコマンド）。パーサー側でこのDTDを考慮するには、マクロファイルの最初の行に次の命令文を追加します。

<!DOCTYPE macros SYSTEM"c:\macros.dtd">
（この文で“c:¥macros.dtd”は、作成されたDTDファイルへのアクセスパスを指定しています）

マクロ用DTD：

```
<?xml version="1.0" encoding="UTF-8"?>
<!--Grammar for 4th Dimension methods macros. 4D SA.-->
<!-- caret position after macro expansion-->
<!ELEMENT caret EMPTY>
<!-- placeholder for selection-->
<!ELEMENT selection EMPTY>
<!-- placeholder for operating system user name-->
<!ELEMENT user_os EMPTY>
<!-- placeholder for 4D user name-->
<!ELEMENT user_4d EMPTY>
<!-- placeholder for method name-->
<!ELEMENT method_name EMPTY>
<!-- placeholder for current time-->
```

```
<!-- format= 0 to 8-->
<!ELEMENT time EMPTY>
<!ATTLIST time
    format CDATA#IMPLIED
>
<!-- placeholder for current date-->
<!-- format= 0 to 6-->
<!ELEMENT date EMPTY>
<!ATTLIST date
    format CDATA#IMPLIED
>
<!-- placeholder for clipboard contents-->
<!-- index= 0 to 9-->
<!ELEMENT clipboard EMPTY>
<!ATTLIST clipboard
    index CDATA#IMPLIED
>
<!-- macro contents-->
<!ELEMENT text(#PCDATA| user_os| user_4d| time| date| method_name| caret|
                                                                    selection)*>
<!-- macro-->
<!-- name= name as displayed in contextual menu-->
<!-- type_ahead_text= text to type to activate the macro using type-ahead(default
                                                        is the macro name) -->
<!-- type_ahead= should this macro be available using type-ahead? (default is true)
-->
<!-- note: if the macro contents use the selection placeholder, it cannot be activated
                                                            using type-ahead-->
<!-- in_menu= should this macro be displayed in contextual menu? (default is true)
-->
<!ELEMENT macro(text?)>
<!ATTLIST macro
    name ID#REQUIRED
    type_ahead_text CDATA#IMPLIED
    type_ahead(true| false) "true"
    in_menu(true| false) "true"
>
<!-- a macro file contains macros-->
<!ELEMENT macros(#PCDATA| macro)*>
```

索引

記号

数字

A

B

C

D

G

I

L

M