4D バージョン３（4D Server バージョン１）から バージョン６, 6.5 への移行について

このドキュメントは、4D バージョン 3（4D Server バージョン１）のデータベースをバージョン 6 または 6.5 へ移行する際の一般的な手順と注意を説明したものです。

● はじめに

基本的に、新しいバージョンで古いバージョンで作成されたストラクチャを開くと、自動的に変換されます。新しいバージョンでアプリケーションが実行される場合は、古いバージョンのまま内部的にシミュレーションをするような形で動作します。新しいバージョンで一度デザイン環境で開いて保存しなおすと（例えばフォーム）、新しいバージョンの形式で内部的にも保存されます。
多くの処理は、問題なく動作しますが、プログラム的に変更を必要とする場合がありますので注意してください。

● バージョン 3 とバージョン 6 の主な違い

- **用語とコマンド体系の変更**
  用語が変更になりました。レイアウト→フォーム、プロシージャ→メソッドなど。それによるコマンドの取り扱いも変更になったものがあります。詳しくは『4th Dimension 用語集』（Glossary.pdf）をご参照ください。
- **クライアント／サーバの最適化による変更**
  セットの扱いなどがかわりました。
- **インタープロセス記号の変更（Macintosh 版）**
  これまで半角のラを使用していたものを、Windows 版と同じく <> に統一しました。
- **フォームイベントの変更**
  レイアウトの実行サイクルの処理（Before, During, After）は、フォームのイベントとして、プログラミング方法も変更になりました。
- **階層リストの導入**
  リストが階層リストにかわりました。リンクリストの扱いがかわります。
- **外部ルーチンの扱い**
  古い外部ルーチン（Proc.Ext 形式）は基本的に使用できません。Mac4DX/Win4DX の形式になったプラグインをご利用ください。また、ストラクチャに統合されたプラグインは使用できません。変換時には取り外しておく必要があります。詳しくは外部ルーチンを開発された方にお問い合わせください。
- **グラフ**
  グラフ関係のコマンドはバージョン 6 でも使用できますが、内蔵された 4D Chart コマンドに置き換えてご利用されることをお勧めします。

● アップグレードの手順

１． 古いバージョンでの修正

プログラミングの内容によっては、古いバージョンで修正を加えてから変換をした方がよい場合があります。別記「プログラム修正のポイント」の内容をご参照になり、プログラムの内容を確認してください。

２． バックアップ

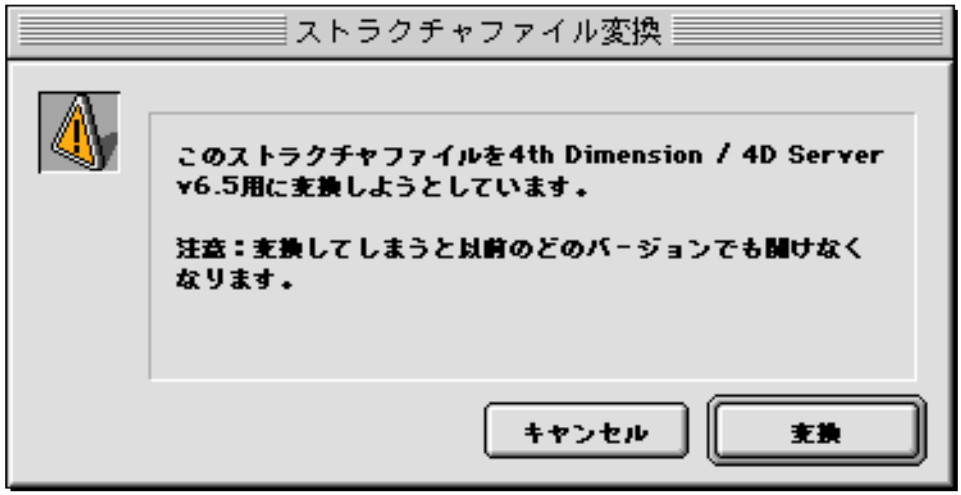
古いバージョンのストラクチャとデータファイルを安全な場所にバックアップします。

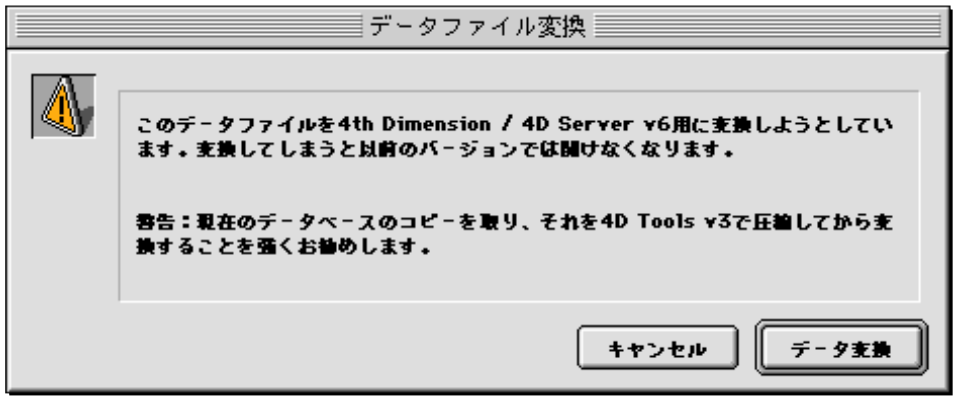
３． 4D Tools での圧縮

データファイルとストラクチャファイルを、古いバージョンの 4D Tools を使用し圧縮します。

４． 新しいバージョンに変換

新しい 4th Dimension で、古いストラクチャとデータを開きます。ストラクチャとデータのそれぞれについて、変換の際に確認のダイアログが表示されます。





５． 新しいバージョンでの動作確認
変換後は、「バージョン 6 での設定変更」を参照し必要な設定を行い、新しいバージョンでの動作テストを行ってください。

● プログラム修正の主なポイント

○ インタープロセス記号

4D バージョン 6 からは、インタープロセス記号が半角のラから <> に変更されています（Macintosh 版）。インタープロセス変数名は、バージョン 6 で開いた場合に自動的に変換されます。

バージョン 3 での記述
ラ Name := “ 山田太郎 ”
バージョン 6 での記述
<>Name := “ 山田太郎 ”

インタープロセスセットやインタープロセス命名セレクション名を使用している場合は、変換後に修正が必要となります。

バージョン 3 での記述
USE SET(“ラ MySet ”)
バージョン 6 での記述
USE SET(“ <>MySet ”)

この問題を解消するために、バージョン 3 でインタープロセスセットを使用する場所では、次のように記述しておく方法があります。

バージョン 3 での記述
USE SET(ラ IP+ ”MySet ”)

このようにインタープロセスセットを使用する場合は、インタープロセス変数を使うようにします。この場合の変数 ラ IP は変換時に自動的に <>IP に置き換えられます。On Startup メソッドでラ IP := “ラ” と定義しておき、変換後に 1 箇所だけを修正すればよくなります。
また、同様に、Get pointer でインタープロセス変数のポインタを文字列から取得する場合も考慮する必要があります。

○ その他の記号の変更

ポインタと文字列参照の記号も変更になっています。

	バージョン３	バージョン６
ポインタ	^^	->
文字列参照	<<　　>>	[[　　]]

○ コマンド名等の変更

4D バージョン 6 からはコマンド名が変更されているものがあります。メソッド内で使用している場合には、自動的に変換されますが、Execute コマンドを使用している場合は、文字列ですので、4D が自動的に変換することができません。Execute コマンドを使用している箇所の修正が必要になります。
なお、Command name コマンドを使用することで、コマンド名の変更があった場合でも問題がないようにプログラミングをすることが可能です。なお、これは英語コマンドとフランス語コマンドなどを使いわける場合などにも有効な手段です。

○ クライアント／サーバでのセットの扱いの変更

4D バージョン 3 では、すべてのセットはサーバで作成された後にクライアント側にコピーされ、サーバ上のセットは消去されていました。バージョン 6 では、ローカルセット以外はサーバ上に存在し、クライアント側にコピーされません。ローカルセットを使用したセットの演算を行っているプログラムなどでは、処理内容を確認する必要がります。

例.

```
DIFFERENCE("CurrentSet";"$SelectedSet";"$SelectedSet ")
```

セットの演算は、サーバ側かクライアント側（ローカル）かの同じ側にある場合にのみ可能です。必要があれば、COPY SET コマンドを使い、セットをコピーする必要があります。
※ シングルユーザモードでテストをすると、サーバ側とクライアント側の区別がありませんので問題が露見しません。ご注意ください。

○ フォームイベントの変更

4D バージョン 3 では、レイアウトの実行サイクルとして Before, During, After フェーズなどの制御をレイアウトプロシージャで行っていました。4D バージョン 6 からはフォームイベントを基本としたプログラミング制御となります。詳しくはアップグレードマニュアルをご覧ください。基本的な形式は、次の通りです。なお、フォームイベントはオブジェクトメソッド（以前のオブジェクトのスクリプトに相当）でも同様に利用することができます。フォームに関するイベント処理はフォームメソッド、オブジェクトに関するイベント処理はオブジェクトメソッドに記述するのが一般的な方法です。

```
` フォームメソッド
$FormEvent:=Form event
  Case of
   ¥ ($FormEvent=On Load )
        ` フォーム読み込み前の処理
   ¥ ($FormEvent=On Close Box )
        `  クローズボックスクリック時の処理
   ¥ ($FormEvent=On Unload )
        `  フォーム終了時の処理
  End case
```

※下線が引いてあるものは、バージョン 6 から導入された定数です。

○ 階層リストの導入

4D バージョン 3 では、リストの関連づけを行うリンクリスト機能がありました。4D バージョン 6 ではリストは階層リストとして新しくなりました。データベースの変換の前にリンクリストの定義ははずしておく必要があります。なお、リンクリストに基づいたプログラミングは、階層リストを利用したプログラミングに変更しなければなりません。

○ファイルプロシージャの扱い

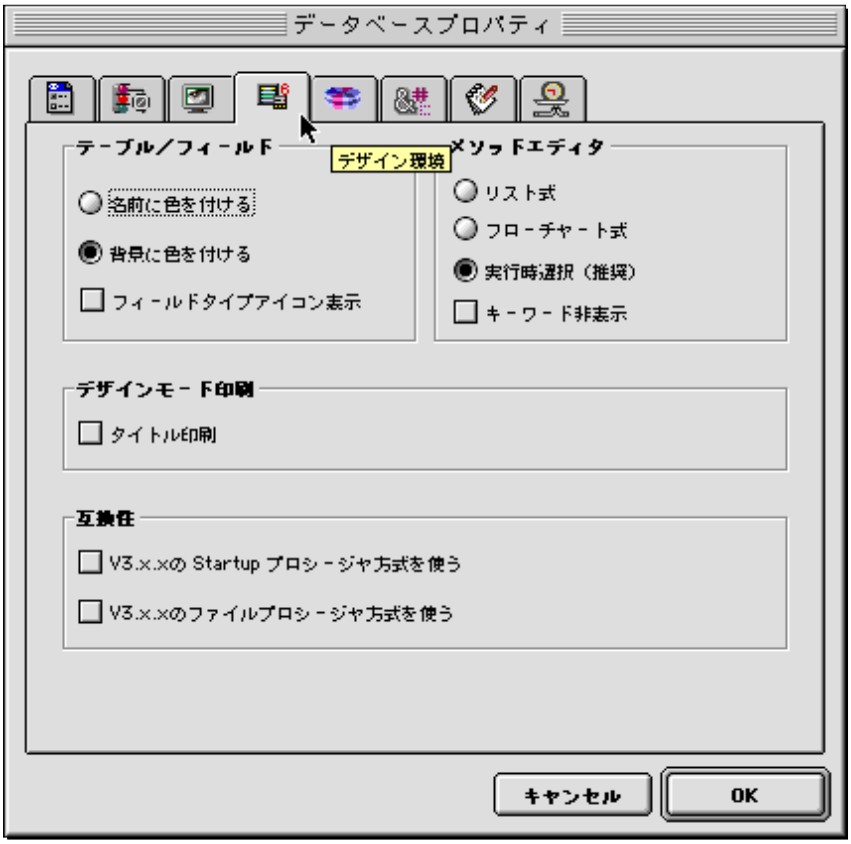
4D バージョン 6 では、トリガが使用できるようになりました。ファイルプロシージャで行っていた処理は、トリガで書き直す必要があります。よりパワフルなトリガをご使用ください。

● バージョン 6 での設定変更

バージョン 6 に変換し終わったら、データベースの設定を確認します。

○ データベースプロパティの設定

デザインモードでデータベースプロパティの設定の互換性の箇所を設定します。



これらの互換性のチェックボックスは、どちらもつけないようにします。これらのチェックボックスは、短時間での変換確認のための機能であり、プログラムをバージョン 6 に書き直す場合には、このチェックに頼らないようにプログラミングを変更することを推奨します。

「v3.x.x の Startup プロシージャ方式を使う」とは、「Startup」という名前のメソッドを起動時に自動的に実行する機能を指します。バージョン 6 では、データベースメソッドの「On startup」がそれに対応するものです。

「V3.x.x のファイルプロシージャ方式を使う」とは、ファイルプロシージャを認識するためのものです。ファイルプロシージャで行っていた機能は、トリガによって置き換えるように書き換えるようにします。

○ Startup プロシージャの対応

4D バージョン 3 では、「Startup」という名前のプロシージャは、起動時に自動的に実行されていました。バージョン 6 では、データベースメソッドがその役割を果たします。Startup メソッドに書かれている内容を 4D で起動したい場合は、「On startup」データベースメソッドに Startup と記述して呼び出すようにしてください。

● その他の注意点

○バージョン２のデータベースの変換

バージョン 2 のデータベースをバージョン 6 に変換することもできます。

○ プラットフォームの変更

バージョンアップとともに Macintosh のデータベースを Windows に変換する場合、一度 Macintosh 版で最新版までデータベースを変換した後に、4D Transporter を使ってデータベースを Windows 用に変換する方法をお勧めします。その場合は、4D バージョン 3 および６の評価版（デモ版）をお使いいただければ、あらためて途中のバージョンの製品をご購入いただく必要はありません。