はじめに

バージョン 1.5 以降、すべてのコネクティビティプラグインは 4th Dimension V6 と完全に互換します。新しくリリースする理由は 2 つあります。製品間や国ごとに異なっていたバージョンを統一し、互換性の問題を解決しやすくするということと、4D Server V6 の利点を活かすために加えられたアーキテクチャ上の変更を取り入れたためです。

このドキュメントについて

このドキュメントでは、前のドキュメントの各所で見つかったさまざまな情報を簡単に説明しています。

このドキュメントには、次の章があります。

- 新しい機能
- システム構成
- 正誤表

第 1 章　新しい機能

4D のコネクティビティプラグインはすべて、はじめの 1.5 のリリースとともに開始したアーキテクチャ上の大きな変更を経験しました。目的は、配備の点において、さらに柔軟性をもたせることでした。バージョン 1.5 の時点で実現された最初のステップは、プラットフォームに応じてリンクモデルを共有ライブラリやダイナミックリンクライブラリへと変換することでした。2 番目のステップは、今回のリリースで実現されましたが、プラグインと必要となるサポートライブラリとの結びつきを " 弱める " ことでした。現在、コネクティビティプラグインは、その基礎となるライブラリがない場合でも、ロードできます。必要となる API (SQL*Net、OpenClient、ODBC) がインストールされていなくても、ユーザはコネクティビティプラグインを使用するデータベースを開始することができます。したがって、使用する予定のない API をクライアントマシンにインストールする必要はなくなり、4D Server V6 でもこれら API がインストールされている必要はありません。

この節では、新しいコマンドや機能、前回のドキュメント以降に出荷されたリリースにともなう変更点について説明します。

4D ODBC 6.0.5 の新機能

OC GET TYPE INFO　*OC GET TYPE INFO*(Cursor_ID;DataType_Attribute; Array_name)

引数	タイプ	説明
Cursor_ID	倍長整数	カーソル ID
DataType_Attribute	整数	データタイプ属性 ID
Array_name	文字列	4D 文字列配列の名前

OC GET TYPE INFO コマンドにより、利用可能なすべてのデータタイプの属性をアプリケーションから取得することができます。

Cursor_ID には、有効なカーソル ID を指定します。

DataType_Attribute は求めるデータタイプ属性の参照番号です。

Array_name は、すべてのデータタイプ属性を受け取る配列の名前です。

▼ 次の例題では、データソース内のデータの名前、長さ、ヌル属性を取得します。

```
ARRAY STRING(30;arName;0)
ARRAY STRING(30;arLength;0)
ARRAY STRING(30;arNull;0)
$result:=OC GET TYPE INFO LIST(cursor_id;1;"arName")
$result:=OC GET TYPE INFO LIST(cursor_id;3;"arLength")
$result:=OC GET TYPE INFO LIST(cursor_id;7;"arNull")
```

データタイプ	属性 ID
名前	1
データタイプ	2
精度	3
接頭辞	4
接尾辞	5
引数作成	6
ヌル可	7
大文字小文字識別	8
検索可	9
符号なし	10
通貨自動増加	11
ローカルタイプ名	12
Min スケール	13
Max スケール	14

参照： OC GET TYPE INFO LIST.

OC Clone 4D Table

ダイアログに新しいオプションが追加され、blob フィールドの複製をスキップできるようになりました。

オプション	説明
kWithout=16	OC Clone 4D Table では、Blob タイプのフィールドが無視される。

OC GET DRIVER CAPABILITIES

OC GET DRIVER CAPABILITIES(Login_ID;Command_ID)

引数	タイプ	説明
Login_ID	倍長整数	ログイン ID
Command_ID	整数	4D ODBC コールの ID

OC GET DRIVER CAPABILITIES コマンドにより、ODBC コールがサポートされるドライバに問い合わせを行い、どの 4D ODBC コマンドが利用できるかを調べることができます。

このコマンドのもうひとつの使用方法では、Command_ID を引数として渡し、特定の 4D ODBC コマンドについて調べます。

4D ODBC コール	ID
OC Login dialog	1
OC Login	2
OC LOGOUT	3
OC Query exec	4
OC Execute SQL	5
OC Clone 4D table	6
OC Create cursor	7
OC DROP CURSOR	8
OC SET CURSOR NAME	9
OC Get cursor name	10
OC Set SQL in Cursor	11
OC Execute cursor	12
OC Execute direct cursor	13
OC Bind	14
OC Load row	15
OC More results	16
OC Number of columns	17
OC Describe column	18
OC Column attributes	19
OC Number rows processed	20
OC Create context dialog	21
OC Create context	22
OC DROP CONTEXT	23
OC ADD TO CONTEXT	24
OC Load context file	25
OC Load context picture	26
OC SAVE CONTEXT FILE	27
OC Save context picture	28
OC EDIT CLAUSES IN CONTEXT	29
OC SET CLAUSE IN CONTEXT	30
OC Get clause in context	31
OC Activate context	32
OC DEACTIVATE CONTEXT	33
OC First in context	34
OC GET DRIVER CAPABILITIES	35
OC Previous in context	36
OC Last in context	37
OC Goto in context	38

4D ODBC コール	ID
OC Load rows context	39
OC Update in context	40
OC Insert in context	41
OC Delete in context	42
OC Get info	43
OC Get function	44
OC GET TYPE INFO LIST	45
OC Get login option	46
OC Set login option	47
OC Get cursor option	48
OC Set cursor option	49
OC GET DSN LIST	50
OC GET TABLE LIST	51
OC GET COLUMN LIST	52
OC GET PRIMARY KEY LIST	53
OC GET TABLE PRIVILEGE LIST	54
OC GET COLUMN PRIVILEGE LIST	55
OC GET PROCEDURE LIST	56
OC GET PROCEDURE COLUMN LIST	57
OC CANCEL LOADING	58
OC TRANSACT COMMAND	59
OC OPEN DEBUG WINDOW	60
OC CLOSE DEBUG WINDOW	61
OC DEBUG MESSAGE	62
OC SET ERROR HANDLER	63
OC Bind parameter	64
OC Describe parameter	65
OC Number of parameters	66
OC Clone ODBC table	67
OC Check configuration	68
OC Next in context	69
OC SET PACK OPTIONS	70
OC Get pack options	71
OC GET TYPE INFO	72

4D ODBC 6.0.2 の新機能

OC Next in context

OC Next in context (CID) → 整数

OC Next in context 関数はカレントローの前の結果ローをロードします。カレントローが定義されていない場合、OC Activate context の実行後と同様に OC Next in context は何も行いません。この機能は以前に、OC load rows context (CID;1) により実現されていました。

CID には必ずあらかじめ作成され、アクティブな状態のコンテキスト ID を指定します。OC Next in context はカレントローに関連する 4th Dimension オブジェクトを更新します。バインドが設定されているフィールドについては、カレントレコードが更新されますが保存されません。カレントレコードが存在しない場合、OC Next in context は対応するバインドを無視します。

処理が正常に終了すると、OC Next in context より 1 が返され、前のローが存在しない場合には 0 が、エラーの場合には -1 が返されます。

結果	説明 :
1	エラーなし。すべて正常に終了。 MacO および Windows
0	エラーなし。ロードする前のローがない。
-1	ローのロード時にエラー発生。

初期設定ファイル (プレファレンスファイル)

4D ODBC プレファレンスファイルの名前がバージョン 1.5 と 6.0 の 4D ODBC で共通に使用できるように、Windows では ODBV6Prf.RSR に、Macintosh では ODBV6Prf に変更されました。

OC OPEN DEBUG WINDOW

OD OPEN DEBUG WINDOW

4D ODBC のデバッグウインドウは、別プロセス内に維持され、各種 4D プロセスから共有して 4D ODBC のデバッグメッセージを表示できるように設計されています。4D ODBC バージョン 6.0.2 より、デバッグウインドウへのメッセージ表示が必要となる 4D ODBC コールはすべて、メッセージを表示する前にウインドウのプロセスが起動終了するまで待機します。この変更により、OD OPEN DEBUG WINDOW のコールと実際のウインドウ表示との中間フェーズでデバッグメッセージが “ 迷子 ” になるのを防ぐことができます。

4D ODBC 6.0.1 の新機能

OC Bind parameter

OC BIND PARAMETER (Cursor_ID;param#;4DObject;IO Type;Conversion) → 整数

引数	タイプ	説明
Cursor_ID	倍長整数	カーソル ID
parameter num	整数	参照番号
4DObject	文字列	4D 変数またはフィールドの名前
IO Type	整数	IO タイプ
Conversion	整数	“ ブラインド ” 変換の場合は 1

バージョン 6.0.1 から “ ブラインド変換 ” オプションが追加されました。ODBC に引数を渡す際に、4D ODBC では 4D の内部データ形式から ODBC で受け入れられるデータ形式へ変換する必要があります。

これを実現するため、4D ODBC では SQL_API_SQLDESCRIBEPARAM という API コールを呼び出し、ODBC ドライバに対して引数の性質（データタイプ）を問い合わせます。ドライバの中にはこのコールがサポートされず、4D ODBC でどのような変換を行うのかを調べることができないものもあります。このようなドライバのために、4D ODBC にブラインド変換が導入されました。必要があれば、4D ODBC は 4D オブジェクトのデータタイプをもとにして引数の値を最も近い ODBC データタイプに変換します。

▼ ストアドプロシージャは 2 つの引数を受け取る。author_id は整数タイプで、book_id は文字列タイプ。

```
C_STRING (30;$Author_id)
C_STRING (30;$Book_id)
C_STRING (30;$unused)
$Author_id:="1245"
$Book_id:="GMN9422"

$DescribeSupported: = OC Get function(Login_ID; 58; "$unused")

If ($DescribeSupported=1)
` デフォルトのデータタイプ変換メカニズムを使用
   $r:=OC Bind parameter (CursID;1;"$author_id";1;0)
   ` 引数の説明に応じて、$Author_id の C_STRING タイプの値が
   ` 整数タイプに変換される
   ` このコールが動作するには、ODBC ドライバで
   ` SQL_API_SQLDESCRIBEPARAM コールがサポートされている必要がある
   ` 大部分のドライバではこのコールはサポートされていない
Else
   $r:=OC Bind parameter (CursID;1;"$Author_id ";1;1)
   ` データタイプ変換メカニズムを無効にする
   ` $Author_id の C_STRING タイプの値 は
   ` 最も近い SQL データタイプである VARCHAR に変換される
   ` 大部分のドライバでは正常に動作するが、柔軟性に欠ける
```

```
   ` データソースの中には暗黙のデータタイプ変換をサポートするものも
   ` あるが、サポートしないものもある
End if
```

ストアドプロシージャの定義がわかっていて、変更されないものであれば、4D の変換メカニズムを使い、***OC Bind parameter*** コマンドに渡す前にデータをあらかじめ変換しておくことをお勧めします。上の例題は、次のように書き換えられます。

```
C_STRING (30;$Author_id)
C_INTEGER ($NumAuthor_id)
C_STRING (30;$Book_id)
C_STRING (30;$unused)

$Author_id:="1245"
$Book_id:="GMN9422"

$DescribeSupported: = OC Get function(Login_ID; 58; "$unused")

If ($DescribeSupported=1)
` デフォルトのデータタイプ変換メカニズムを使用
   $r:=OC Bind parameter (CursID;1;"$author_id";1;0)
   ` 引数の説明に応じて、$Author_id の C_STRING タイプの値が
   ` 整数タイプに変換される
   ` このコールが動作するには、ODBC ドライバで
   ` SQL_API_SQLDESCRIBEPARAM コールがサポートされている必要がある
   ` 大部分のドライバではこのコールはサポートされていない
Else
   $NumAuthor_id := Num ($Author_id)
   ` データタイプ変換メカニズムを無効にする
   $r:=OC Bind parameter (CursID;1;"$NumAuthor_id ";1;1)
   ` $NumAuthor_id の C_INTEGER タイプの値 は
   ` 最も近い SQL データタイプである INTEGER に変換される
   ` 大部分のドライバでは正常に動作するが、柔軟性に欠ける
End if
```

次の表に従ってブラインド変換が行われます。

4D オブジェクト	ODBC データタイプ
C_STRING	SQL_CHAR
C_TEXT	SQL_CHAR
C_BOOLEAN	SQL_CHAR
C_DATE	SQL_TIMESTAMP
C_TIME	SQL_TIMESTAMP
C_REAL	SQL_DOUBLE
C_LONGINT	SQL_INTEGER
C_INTEGER	SQL_SMALLINT
C_PICTURE	SQL_BINARY

OC Check configuration

OC Check configuration (\) → 整数

API の使用可能状況を調べるために、すべてのコネクティビティプラグインには同様の関数が追加されました（XX Check configuration）。この関数を呼び出すと、整数が返され、システム構成の状況がわかります。

この関数により、コマンドでクライアントのシステム構成を調べ、プラグインが正しく動作しない場合に問題となる原因を知ることができます。

結果	説明
1	エラーなし。テスト結果は正常。MacOs および Windows。
- 1	ライブラリが見つからない。MacOS。
- 2	ライブラリをロードするためのメモリ不足。MacOS。
- 3	見つかったライブラリのバージョンがこのバージョンの 4D ODBC ではサポートされない。またはライブラリがロードされない。MacOS。
-101	DLL が見つからない。Windows。
-102	DLL をロードできない。Windows。

▼ Example

```
   ` Startup データベースメソッド
$check:=OC Check configuration
Case of
   \($check=1)
      <>SeverAvailable:=True
   \($check=-1)` API not present
      <>ServerAvailable:=False
   Else
      ALERT (" The ODBC API が一部見つからない、または正しくありません ")
      <>ServerAvailable:=False
End case
If (<>ServerAvailable)
   DO_Synchro ([Customers])
End if
```

OC Clone table

OC Clone table **(Table_ID;{Table_Name}) → 整数**

引数	タイプ	説明
Login_ID	倍長整数	ログイン ID
Table_Name	文字列	4D テーブルの名前

OD Clone table 関数は、選択したデータソーステーブルと同じストラクチャを持つ 4th Dimension テーブルを作成します。ダイアログが表示され、ユーザはカレントユーザからアクセスできるテーブル一覧の中からひとつを選択することができます。選択後、4D ODBC により同じ名前と同様のフィールドリストを持つ 4D テーブルが作成されます。

2 番目の引数としてテーブル名を渡すと、ユーザにテーブルを選択させる代りに、渡された名前を持つテーブルのクローンが作成されます。4D テーブルは同じ名前を持つことになります。

▼ 例題

```
` OC CloneDB (CursorID)
` データソース上で見つかったすべてのテーブルのクローンをローカルに作成
` CursorID : 倍長整数、OC Create cursor より

C_LONGINT ($curs)
$curs := $1

ARRAY STRING(15;t1;0)
ARRAY STRING(15;t2;0)
ARRAY STRING(60;tab_name;0)
ARRAY STRING(15;tab_type;0)
ARRAY STRING(15;t5;0)

OC GET TABLE LIST ($curs;"t1";"t2";"tab_name";"tab_type";"t5")
For ($i;1;Size of array(tab_type))
   If (tab_type{$i}="table")
     MESSAGE(" クローンテーブル :"+tab_name{$i})
     $r:=OC Clone ODBC table (l;tab_name{$i})
   End if
End for
```

第 2 章　システム構成

この節では、サポートされているマシンおよびシステムにインストールを正しく行うために必要となる事柄について説明します。

動作環境

Windowsクライアント

Windows 版 4D ODBC の使用に際して、次のソフトウェアが最低限必要となります。

- Windows 95、Windows NT 3.5 以上のバージョン
- 4th Dimension 3.5 以上、4D Engine 3.5 以上、4D Runtime classic 3.5.3 以上 , 4D server および 4D client 1.5 以上のバージョン
- 12 MB 以上の RAM を搭載した x86 PC 互換機 (Windows NT の場合は、16 MB)
- 使用するデータソースに特定の 32 ビットの ODBC ドライバ。通常、ドライバは、データソースやサーバの販売元、またはこのタイプのミドルウェアを取り扱うベンダより提供される。

MacOs クライアント

Macintosh 版 4D ODBC の使用に際して、次のソフトウェアが最低限必要となります。

- System 6.0.7
- 4th Dimension 3.0.2 以上、4D Server または 4D Client バージョン 1.0.2 以上
- 使用するデータソースに特定の ODBC ドライバ。通常、ドライバは、データソースやサーバの販売元、またはこのタイプのミドルウェアを取り扱うベンダより提供される。
- Visigenic の ODBC ドライバマネージャ (VISIGENIC 社または APPLE 社より提供)

第 3 章　正誤表

この節では、前のドキュメントから変更された点および誤りについて説明します。

OC GET TYPE INFO LIST

***OC GET TYPE INFO LIST*(Cursor_ID; Array_name1;Array_name2; Array_name3; Array_name4; Array_name5; Array_name6; Array_name7; Array_name8; Array_name9; Array_name10; Array_name11; Array_name12; Array_name13; Array_name14; Array_name15)**

引数	タイプ	説明
Cursor_ID	倍長整数	カーソル ID
Array_name1	文字列	名前の配列名
Array_name2	文字列	データタイプ ID の配列名
Array_name3	文字列	精度の配列名
Array_name4	文字列	接頭辞の配列名
Array_name5	文字列	接尾辞の配列名
Array_name6	文字列	引数作成の配列名
Array_name7	文字列	ヌル可の配列名
Array_name8	文字列	大文字小文字識別の配列名
Array_name9	文字列	検索可の配列名
Array_name10	文字列	符号なしの配列名
Array_name11	文字列	通貨の配列名
Array_name12	文字列	自動増加の配列名
Array_name13	文字列	ローカルタイプ名の配列名
Array_name14	文字列	Min スケールの配列名
Array_name15	文字列	Max スケールの配列名

OC GET TYPE INFO LIST コマンドにより、利用可能なすべてのデータタイプの全データ属性をアプリケーションから取得することができます。配列はすべて文字列配列として定義する必要があります。

参照：OC GET TYPE INFO

OC TRANSACT COMMAND

OC TRANSACT COMMAND(Login_ID; CursorID;Commit/Rollback)

引数	タイプ	説明
Login_ID	倍長整数	ログイン ID
CursorID	倍長整数	カーソル ID
Commit/Rollback	ブール	コミットは “ False ”、ロールバックは “ True ”

OC Execute direct cursor

OC Execute direct cursor(CursorID; SQLCommand) → 整数

引数	タイプ	説明
CursorID	倍長整数	カーソル ID
SQLCommand	テキスト	実行する SQL コマンド

OC Execute SQL

OC Execute SQL(CursorID; SQLCommand;Limit;ArrayList) → 整数

引数	タイプ	説明
CursorID	倍長整数	カーソル ID
SQLCommand	テキスト	実行する SQL コマンド
Limit	整数	返す最大行数
ArrayList	文字列配列	配列名の配列

```
…
ARRAY STRING(30;aRep;0)
ARRAY STRING(30;arCust;0)
ARRAY STRING(30;arProduct;0)
ARRAY STRING(30;arArrays;3)
arArrays{1}:="arRep"
arArrays{2}:="arCust"
arArrays{3}:="arProduct"

OC OPEN DEBUG WINDOW
$sql:="select REPID, CUSTID, PRODNAME from sales"
$res:=OC Execute SQL (curs;$sql;-1;aArrays)
```

OC Set login option

OC Set login option(LoginID; OptionID; OptionValue)

引数	タイプ	説明
LoginID	倍長整数	ログイン ID
OptionID	整数	オプションの ID
OptionValue	文字列	オプションに設定する値

```
…
$result:=OC Set login option(loginID;102;"0")
…
```

OC GET DSN LIST

OC GET DSN LIST(CursorID; SQLCommand)

引数	タイプ	説明
CursorID	倍長整数	カーソル ID
SQLCommand	テキスト	実行する SQL コマンド

接続する必要がないため、このコマンドにカーソル ID は不要です。

互換性の理由および既存のアプリケーションのため、カーソル ID がこのコマンドの引数として残されています。