目次

DataServerELS, UNIFY4.0 / DataServer, UNIFY2000 Database系

ACCELL/IDS , ACCELL/SQL 系

Unify Vision系

UNIFY

UNIFY4.0からUNIFY DataServer ELS へのコンバージョン

DataServerELSインストールディレクトリ　　/home/ELS

1. データベースの移行

Step1 DataServerELSの環境変数の設定

```
DBPATH=/home/ELS_DB; export DBPATH
PATH=/home/ELS:/home/ELS/bin:$PATH ; export PATH
UNICAP=/home/ELS/lib/unicap ; export UNICAP
TERMCAP=/home/ELS/lib/termcap ; export TERMCAP
UNIFY=/home/ELS/lib ; export UNIFY
```

Step2 データベースのコピー

UNIFY 4.0 のDBPATH に設定されているディレクトリ以下の全ファイルを
UNIFY DataServer ELS の環境へコピーする。

［例］

```
$ cd /home/U40_DB            unify 4.0 のディレクトリ
$ cp * /home/ELS_DB/.    DataServer ELSの$DBPATHへ全ファイルをコピー
```

Step3 共有メモリIDの削除

ipcsで共有メモリIDを調べ、ipcrm –mで削除する。

Step4 自動変換

シェルスクリプト unify を実行し、UNIFY DataServer ELS を起動する。
データベースは、この時点でUNIFY DataServer ELS へ自動変換される。

Step5 自動変換後の確認

ELS のメインメニューより「3. ＳＱＬ-問合せ／ＤＭＬ言語」 を選択し
sql> のプロンプトから tables を実行する。

［例］

```
sql> tables
tbl1     tbl2     tbl3     tbl4     tbl5     tbl6
```

全テーブルが正しく表示されていることを確認する。
問題なければ、データベースの移行は正常終了である。

２．データベース移行後の作業

Step1 システム･タイトルの変更

画面に表示されるシステム･タイトルは、移行前の状態になっている。
″ＵＮＩＦＹ ＤＢＭＳ 4.0″
「システムパラメータの変更（parmnt）」を使用して、システム･タイトルを次の名称に変更する。
″UNIFY DataServer ELS″

Step2 Ｂツリーインデックスの再構成
「Ｂツリーインデックスの追加／削除（idxmnt）」を使用して、
全Ｂツリーインデックスの再構成を実行する。

Step3 日付データの定義
（１）属性DATEの表示フォーマットは、UNIFY4.0ではDATETPで設定したが、
UNIFY DataServer ELSでは、DATETPを使用せず、かわりにSDATFMTを
を使用する。

［設定例］
```
SDATFMT=YY/MM/DD
export  SDATFMT
```

（２）UNIFY DataServer ELSは、２桁年フォーマットの日付データを2000台の日付として認識させる機能を持っている。
これを実現させる環境変数CENTURY_CUTOFFを設定する。

［設定例］
```
CENTURY_CUTOFF=20
export   CENTURY_CUTOFF
```

UNIFY

DataServer ELS からDataServer へのコンバージョン

DataServer　ELSインストールディレクトリ　/home/ELS
DataServer インストールディレクトリ　/home/DS

1. ＳQL/Aデータベース設計変換スクリプトの作成

Step1　DataServer ELSの環境変数の設定

```
UNIFY=/home/ELS/lib
DBPATH=/home/ELS_DB
DBNAME=file.db
PATH=/home/ELS:/home/ELS/bin:$PATH
REL2000=/home/DS
CONV=/ home/DS/conv
PATH=/ home/DS/conv:$PATH
```

以下の2つはここでは必要ないが、ACCELL/IDSのアプリケーションを実行するときには必要になる。

```
TERMCAP=$UNIFY/termcap
UNICAP=$UNIFY/unicap
```

Step2 ファイルのセーブ

DBPATH内のすべてのファイルをセーブし、DBPATHには3つのファイルのみ残しておく。
残しておいた3つのファイル

```
file.db
file.dbr
unify.db
```

Step3 DataServer用のデータベースを作成するためのディレクトリを作成し、ディレクトリを移る。

```
$ mkdir $DBPATH/../DS_DB
$ cd $DBPATH/../DS_DB
```

Step4 変換準備ユーティリティの実行

```
$ $CONV/prep.sh
```

Step5 変換ユーティリティの修正

$CONV/conv.sh中の下記の箇所をコメントにする。

```
# insert.err should have 3 lines saying 'recognized query', else error
#if [ ! `cat insert.err
#then
#   echo "Error:   Trouble loading data into conversion control data base."
   cat insert.log insert.err
#   echo "Exiting ....                                          " `date`
#   exit 2
```

```
#fi
#rm -f insert.log insert.err
```

Step6 変換ユーティリティの実行

```
$ $CONV/conv.sh > conv.log 2>&1 &
```

Step7 変換ユーティリティ実行後のスクリプト確認

変換ユーティリティが正常に実行されれば、DataServer ELSのデータベース設計がＤＤＬ形式に変換されている。失敗した場合は、スクリプトファイルは作成されず、ワーク用ディレクトリworktmpが残っているので、失敗の原因を追及した後、$DBPATH/../DS_DBの内容をすべてクリアし、コンバージョンを再実行する。
成功した場合に作成される主なDDLスクリプトを掲げる。

Ucreatedb.sql	データベースを作成するためのDDL
Utables.sql	テーブルを作成するためのDDL
Usynonyms.sql	列のシノニムを作成するためのDDL
Uhashtabs.sql	ハッシュインデックスを作成するためのDDL
Ulinks.sql	リンクインデックスを作成するためのDDL
Ubtrees.sql	Bツリーインデックスを作成するためのDDL
U4.dump.sql	全テーブルのレコードをファイル落すSQLコマンド
U2000.load.sql	DataServer にレコードをロードするためのSQLコマンド
U2000.sh	データベースを作成するためのシェルスクリプト
Uaddcgp.sh	カラムグループを定義するためのシェルスクリプト

[注釈]

1) Utables.sqlに記されたテーブル名にはショート名が、列名にはロング名が使われている。列のショート名はシノニムという形式でUsynonyms.sqlに出力されている。

2)コンビネーションフィールド（COMB）に該当する機能がDataServerにはなく、Utables.sqlではカットされ、コンポーネントフィールドのみ落ちている。
グループカラムは、Uaddcgp.shで定義する。

3)DataServerは、ハッシュインデックスが自動で作成されないため、一次キーに対して Uhashtabs.sqlで明示的に作成しなくてはならない。

4)セキュリティの定義は変換できない。

5)従来のエンター画面はDataServerに移行することはできない。

2. DataServerデータベースの作成

Step1 DataServer の環境変数の設定

```
UNIFY=/home/DS/lib
DBPATH=/home/DS_DB
DBNAME=file.db
PATH=/home/DS/bin:$PATH
```

以下の2つはここでは必要ないが、ACCELL/SQLのアプリケーションを実行するときには必要になる。

```
TERMCAP=$UNIFY/termcap
UNICAP=$UNIFY/unicap
```

Step2　file.cf（コンフィギュレーションファイル）の作成。

$UNIFYにあるprod.cfをfile.cfにコピーする。

(例) cp　$UNIFY/prod.cf　　$DBPATH/file.cf

コピー後、file.cfを修正し、SHMKEYにデータベース単位でユニークな値(例えば6904など)を設定する。LMSHMKEYの行はコメントにする。

Step3　conv.shで作成したU2000.shを実行し、DataServerのデータベースを作成する。

```
$ U2000.sh > sql.log 2>&1
```

Step4 列グループの登録

```
$ Uaddcgp.sh
```

3. データ転送(DataServer ELS->DataServer)

Step1 DataServer ELSの環境変数の設定

```
UNIFY=/home/ELS/lib
DBPATH=/home/ELS_DB
DBNAME=file.db
PATH=/home/ELS:/home/ELS/bin:$PATH
```

Step2 DataServer ELSレコードのダンプ

全テーブルに対し次の作業を実行し、データをバイナリファイルにダンプする。

```
$ cd /home/DB_DS/data
$ SQL
$ sql> select * from テーブル名 into　bin ‘テーブル名.dat’ /
```

Step3 DataServerの環境変数の設定

```
UNIFY=/home/DS/lib
DBPATH=/home/DS_DB
DBNAME=file.db
PATH=/home/DS/bin:$PATH
```

Step4 レコードを DataServerでデータのロード

全テーブルに対し次の作業を実行し、Step２でダンプしたバイナリファイルから DataServerにレコードを登録する。

```
$ cd /home/DB_DS/data
$ SQL.
$ insert into テーブル名 values from bin ‘テーブル名.dat’;
```

［注釈］

変換対象がUNIFY2000の場合は、$CONV/conv.sh実行時に作成される U40dump.sqlとU2000.load.dbldを使用してロードを行なうことができ、この方が手順としては簡単である。

変換対象がDataServerの場合、U40dump.sqlとU2000.load.dbldにレコードダンプを行なうための情報が書き出されないため、Step２とStep4の手順の通り、SQL コマンドを直接実行して変換を行なう必要がある。

UNIFY

UNIFY2000 Release1からDataServer へのコンバージョン

UNIFY2000 Release1インストールディレクトリ　/home/U2000R1
DataServerインストールディレクトリ　/home/DS

１．データベースの変換

Step1 DataServerの環境変数の設定

```
UNIFY=/home/DS/lib
DBPATH=/home/DS_DB
DBNAME=file.db
PATH=/home/DS/bin:$PATH
```

［注釈］
DBPATHは、変換対象のUNIFY2000リリース１のデータベースディレクトリに設定する。

Step2 .cfファイルを編集し以下の設定を行う。

```
LOGFM=TRUE
AUTOSTART=TRUE
```

Step3 /home/DS/bin にある変換用シェルスクリプトdbcnv13-21を使って変換を実行する。

```
$ dbcnv13-21
```

［注釈］
ACCELL付でないDataServer単独の商品の場合、bin下にdbcnv13-21のないリリースがあるので、実行前に必ず確認する。

Step4 Bツリーインデックス再作成
すべてのBツリーインデックスをドロップし、作成し直す。

［注釈］
UNIFY2000 では、各Bツリーインデックスは単独ファイル（*.idx）の形で管理されていたが、DataServer ではデータベースファイル（file.db）内に記録される。
Bツリーインデックス作成のためのDDL文はdumpddユーティリティを使用して抽出する。

Step5 remkview実行　（Solaris版のみ）

```
$remkview
```

UNIFY

UNIFY2000 Release2 からDataServer へのコンバージョン

UNIFY2000 Release2インストールディレクトリ　/home/U2000R2
DataServerインストールディレクトリ　/home/DS

１．データベースの変換（自動）

Step1 DataServer の環境変数の設定

```
UNIFY=/home/DS/lib
DBPATH=/home/DS_DB
DBNAME=file.db
PATH=/home/DS/bin:$PATH
```

［注釈］
DBPATHは、変換対象のUNIFY2000リリース２のデータベースディレクトリに設定する。

Step2 自動コンバート
startdbを実行すると、データベースはDataServer対応に自動コンバートされる。

Step3 Bツリーインデックス再作成
すべてのBツリーインデックスをドロップし、作成し直す。

［注釈］
UNIFY2000 では、各Bツリーインデックスは単独ファイル（*.idx）の形で管理されていたが、DataServerではデータベースファイル（file.db）内に記録される。
Bツリーインデックス作成のためのDDL文は、dumpddユーティリティを使用して抽出する。

Step4 remkview実行　（Solaris版のみ）

```
$remkview
```

ACCELL/IDS で上位Releaseへのバージョンアップ

ACCELL/IDS　現リリースインストールディレクトリ　　/home/AIDS
ACCELL/IDS　新リリースインストールディレクトリ　/home/AIDS_NEW

［注釈］
ACCELL/IDS(UNIFY4.0対応)でACCELL/IDS(UNIFY DataServerELS対応)にバージョンアップを行なう場合は、データベースの移行作業（UNIFY4.0からUNIFY DataServerELS）を行なった後に、下記の作業を行なう。

［注釈］
ACCELL/IDS(UNIFY DataServerELS対応)で上位リリースにバージョンアップを行なう場合は、データベースの移行作業は不要。

1. ACCELLアプリケーションの変換 Part.1(ACCELL/IDS 現リリース)

Step1 ACCELL/IDS 現リリースの環境変数の設定

```
DBPATH=/home/ELS_DB ; export DBPATH
PATH=/home/AIDS:/home/AIDS/bin:$PATH ; export PATH
UNICAP=/home/AIDS/lib/unicap ; export UNICAP
TERMCAP=/home/AIDS/lib/termcap ; export TERMCAP
UNIFY=/home/AIDS/lib ; export UNIFY
```

Step2 フォームをASCIIフォーマットに変換

```
マスターフォームの場合、Q2ASC  -a フォーム名 > フォーム名.az
一般フォームの場合、    Q2ASC  フォーム名 > フォーム名.fz
ヘルプフォームの場合、  H2ASC  フォーム名 > フォーム名.hz
```

２. ACCELLアプリケーションの変換 Part.2(ACCELL/IDS 新リリース)

Step1 ACCELL/IDS　新リリースの環境変数の設定

```
DBPATH=/home/ELS_DB ; export DBPATH
PATH=/home/AIDS_NEW:/home/AIDS_NEW/bin:$PATH ; export PATH
UNICAP=/home/AIDS_NEW/lib/unicap ; export UNICAP
TERMCAP=/home/AIDS_NEW/lib/termcap ; export TERMCAP
UNIFY=/home/AIDS_NEW/lib ; export UNIFY
```

Step2 アスキー化したフォームファイルから、ACCELL/IDS　新リリース対応のフォームファイルを作成する。

```
マスターフォームの場合、ASC2Q  -a フォーム名 < フォーム名.az
一般フォームの場合、    ASC2Q  フォーム名 < フォーム名.fz
ヘルプフォームの場合、  ASC2H  フォーム名 < フォーム名.hz
```

Step3 makeファイルを作成し、コンパイル／結合／リンク

```
$makeamake
$make
```

Step4 システム･タイトルの変更

シェルスクリプト accell を実行してメインメニューを立ちあげた際、画面に表示されるシステム表題が、前のリリースのバージョンである場合は、「システムパラメータの変更（parmnt）」を使用して、システム表題を新しいリリースのバージョンに併せて変更する。

"ACCELL Release 1.4E"
↓ 変更
"ACCELL Release 8.1H"

UNIFY

ACCELL/IDS から ACCELL/SQL へのコンバージョン

ACCELL/IDS インストールディレクトリ　　/home/AIDS
ACCELL/SQL インストールディレクトリ　　/home/ASQL

1. SQL/Aデータベース設計変換スクリプトの作成

Step1 ACCELL/IDSの環境変数の設定

```
UNIFY=/home/AIDS/lib
DBPATH=/home/ELS_DB
DBNAME=file.db
PATH=/home/AIDS:/home/AIDS/bin:$PATH
REL2000=/home/ASQL
CONV=/ home/ASQL/conv
```

以下の2つはここでは必要ないが、ACCELL/IDSのアプリケーションを実行するときには必要になる。

```
TERMCAP=$UNIFY/termcap
UNICAP=$UNIFY/unicap
```

Step2 ファイルのセーブ

DBPATH内のすべてのファイルをセーブし、DBPATHには3つのファイルのみ残しておく。

残しておいた3つのファイル

```
file.db
file.dbr
unify.db
```

Step3 DataServer用のデータベースを作成するためのディレクトリを作成し、ディレクトリを移る。

```
$ mkdir $DBPATH/../DS_DB
$ cd $DBPATH/../DS_DB
```

Step4 変換準備ユーティリティの実行

```
$ $CONV/prep.sh
```

Step5 変換ユーティリティの実行

```
$ $CONV/conv.sh > conv.log 2>&1 &
```

Step6 prod.cfのコピー

$UNIFYにあるprod.cfをfile.cfにコピーしSHMKEYは適当な数値を入れる。(例えば6904など)LMSHMKEYはコメントにしておく。

2. ACCELL/SQL(DataServer)データベースの作成

Step1 ACCELL/SQL (DataServer)の環境変数の設定

```
UNIFY=/home/ASQL/lib
DBPATH=/home/DS_DB
DBNAME=file.db
PATH=/home/ASQL/bin:$PATH
```

以下の2つはここでは必要ないが、ACCELL/SQLのアプリケーションを実行するときには必要になる。

```
TERMCAP=$UNIFY/termcap
UNICAP=$UNIFY/unicap
```

Step2 ACCELL/SQL (DataServer)のデータベースを作成する。

```
$ U2000.sh > sql.log 2>&1
```

Step3 列グループの登録

```
$ Uaddcgp.sh
```

3. データ転送(DataServer ELS->DataServer)

Step1 ACCELL/IDSの環境変数の設定

```
UNIFY=/home/AIDS/lib
DBPATH=/home/ELS_DB
DBNAME=file.db
PATH=/home/AIDS:/home/AIDS/bin:$PATH
```

Step2 データのダンプ

```
$ cd /home/DB_DS/data
$ SQL ../U4.dump.sql > dump.log 2>&1 &
```

Step3 ACCELL/SQL (DataServer)の環境変数の設定

```
UNIFY=/home/ASQL/lib
DBPATH=/home/DS_DB
DBNAME=file.db
PATH=/home/ASQL/bin:$PATH
```

Step4 データのロード

```
$ sh ../U2000.load.dbld >dbld.log 2>&1 &
```

4. ACCELLアプリケーションの変換

Step1 ACCELL/IDSの環境変数の設定

```
UNIFY=/home/AIDS/lib
DBPATH=/home/ELS_DB
DBNAME=file.db
PATH=/home/AIDS:/home/AIDS/bin:$PATH
TERMCAP=$UNIFY/termcap
UNICAP=$UNIFY/unicap
```

Step2 シェルを使ってフォームをASCIIフォーマットに変換

```
Q2ASC -a application_name.aq > application_name.az
for file in *.fq; do
        Q2ASC $file > `basename $file .fq`.fz
done
for file in *.hlp; do
        H2ASC $file > `basename $file .hlp`.hz
done
```

Step3 “= UNDEFINED”と”<> UNDEFINED”を”IS UNDEFINED”と”IS NOT UNDEFINED”に変換(= undefinedと<> undefinedについても同様)

```
for file in *.?s; do
        sed s/”= UNDEFINED”/”IS UNDEFINED”/g $file > $file.1
        sed s/”<> UNDEFINED”/”IS NOT UNDEFINED”/g $file.1 > $file.2
        mv $file $file.tmp
        mv $file.2 $file
        rm $file.1 $file.tmp
done
```

Step4 ACCELL/SQL (DataServer)の環境変数の設定

```
UNIFY=/home/ASQL/lib
DBPATH=/home/DS_DB
DBNAME=file.db
PATH=/home/ASQL/bin:$PATH
```

Step5 TRANを使ってACCELL/SQLフォーマットに変換

```
$ TRAN -S PUBLIC *z *s
```

Step6 z2q2zを使って、フォームへ変換

```
$ export CONV=/home/ASQL/conv
$ $CONV/z2q2z
```

Step7 makeファイルを作成し、コンパイル

```
$makeamake
$make
```

1. ACCELLユーザ関数(C-Hooks)の相違点

AVAL構造体が変更されたため、リターン値を定義する場合、avalmacs.hにあるマクロを使用することになった。それに伴い、ユーザ関数の変更が必要である。変更箇所は以下の通り。

旧コード(ACCELL/IDS)	新コード(ACCELL/SQL)
aval.dfflg = 0;	MKVALUDF(&aval);
aval->dfflg = 0;	MKVALUDF(avalp);
aval.dfflg = 1;	MKVALDFNN(&aval);
aval->dfflg = 1;	MKVALDFNN(avalp);
n/a	NULLを設定するには: MKVALDFNL(&avalp);
aval.dfflg == 1	ISVALDF(&aval)
aval->dfflg == 1	ISVALDF(aval)
aval.dfflg == 0	ISVALUDF(&aval)
aval->dfflg == 0	ISVALUDF(aval)

2. カスタムマネージャの再リンク

Step1ユーザ関数の変更

上記の対応表をもとに該当箇所をACCELL/SQLのコードに置き換える。

Step2 ACCELL/SQL 環境でカスタムマネージャの再リンク

通常の方法でカスタムマネージャを再リンクする。ただし、ACCELL/SQLでアプリケーションのチェックを行わない場合には、このステップは省略できる。

例えば、namechk.cというファイルをコンパイルし、カスタムマネージャにリンクするには、以下のようにする。

```
$ cc -c -I$UNIFY/.. namechk.c
$ cc -c -I$UNIFY/.. chooktb.c

$ amgr.ld CUSTAMGR namechk.o chooktb.o
```

ACCELL/SQL Release1から上位Releaseへのバージョンアップ

ACCELL/SQL Release1インストールディレクトリ　/home/ASQL1
ACCELL/SQL 新リリースインストールディレクトリ　/home/ASQL_NEW

現在のACCELL/SQL対応データベースがUNIFY2000で、ACCELL/SQL　新リリースの対応データベースがDataServerとなる場合は、データベース移行処理(UNIFY2000 → DataServer)を行なった後に、下記の作業を行なう。

1. ACCELLアプリケーションの変換Part.1(ACCELL/SQL Release1)

Step1 ACCELL/SQL Release1の環境変数の設定

```
UNIFY=/home/ASQL1/lib
DBPATH=/home/DB_DS
DBNAME=file.db
PATH=/home/ASQL1/bin:$PATH
TERMCAP=$UNIFY/termcap
UNICAP=$UNIFY/unicap
```

Step2 フォームをASCIIフォーマットに変換

```
マスターフォームの場合、Q2ASC　–a フォーム名 > フォーム名.az
一般フォームの場合、　Q2ASC　フォーム名 > フォーム名.fz
ヘルプフォームの場合、　H2ASC　フォーム名 > フォーム名.hz
```

２. ACCELLアプリケーションの変換Part.2(ACCELL/SQL)

Step1 ACCELL/SQL 新リリースの環境変数の設定

```
UNIFY=/home/ASQL_NEW/lib
DBPATH=/home/DS_DB
DBNAME=file.db
PATH=/home/ASQL_NEW/bin:$PATH
```

Step2 TRANを使って、フォームファイルと４ＧＬスクリプトファイルを
ACCELL/SQL フォーマットに変換

```
$ TRAN –s PUBLIC *z *s
```

Step3 アスキー化したフォームファイルから、ACCELL/SQL 新リリース用のフォームファイルを作成する。

```
マスターフォームの場合、 ASC2Q　–a フォーム名 < フォーム名.az
一般フォームの場合、　ASC2Q　フォーム名 < フォーム名.fz
ヘルプフォームの場合、　ASC2H　フォーム名 < フォーム名.hz
```

Step4　makeファイルを作成し、コンパイル／結合／リンク

```
$makeamake
$make
```

ACCELL/SQL Release2 以降で上位Releaseへのバージョンアップ

ACCELL/SQL 現インストールディレクトリ　　/home/ASQL
ACCELL/SQL 新リリースインストールディレクトリ　　/home/ASQL_NEW

現在のACCELL/SQL対応データベースがUNIFY2000で、ACCELL/SQL　新リリースの対応データベースがDataServerとなる場合は、データベース移行処理(UNIFY2000 → DataServer)を行なった後に、下記の作業を行なう。

1. ACCELLアプリケーションの変換Part.1(ACCELL/SQL 現リリース)

Step1 現在のACCELL/SQL の環境変数の設定

```
UNIFY=/home/ASQL/lib
DBPATH=/home/DB_DS
DBNAME=file.db
PATH=/home/ASQL/bin:$PATH
TERMCAP=$UNIFY/termcap
UNICAP=$UNIFY/unicap
```

Step2 フォームをASCIIフォーマットに変換

```
マスターフォームの場合、Q2ASC　–a フォーム名 > フォーム名.az
一般フォームの場合、　　Q2ASC　フォーム名 > フォーム名.fz
ヘルプフォームの場合、　H2ASC　フォーム名 > フォーム名.hz
```

２. ACCELLアプリケーションの変換Part.2(ACCELL/SQL 新リリース)

Step1 ACCELL/SQL 新リリースの環境変数の設定

```
UNIFY=/home/ASQL_NEW/lib
DBPATH=/home/DS_DB
DBNAME=file.db
PATH=/home/ASQL_NEW/bin:$PATH
```

Step2 アスキー化したフォームファイルから、ACCELL/SQL 新リリース用のフォームファイルを作成する。

```
マスターフォームの場合、 ASC2Q　–a フォーム名 < フォーム名.az
一般フォームの場合、　　ASC2Q　フォーム名 < フォーム名.fz
ヘルプフォームの場合、　ASC2H　フォーム名 < フォーム名.hz
```

Step3 makeファイルを作成し、コンパイル／結合／リンク

```
$makeamake
$make
```

ACCELL/SQL からUnifyVISION Release2へのコンバージョン

ACCELL/SQL インストールディレクトリ　/home/ASQL
UnifyVISION Release2インストールディレクトリ　/home/VISION2

1. UnifyVISIONアプリケーションの変換

Step1 UnifyVISIONとDataServer の環境変数の設定

```
PATH=/home/VISION2/bin:$PATH
VISION_HOME=/home/VISION2
GALAXYHOME=/home/VISION2/gui
LANG=japanese
LANG_DIR=jpn_jae
DISPLAY=jupiter
UNIFY=/home/ASQL/lib
CLIENTINFO=/home/DS_DB
DCMFILE=/export/home/ohmura/vdev/tutorial/tutorial.dcm
DBCONN=TUTORIAL._U2K
```

【tutorial.dcmの内容】

```
[TUTORIAL_U2K]
DBTYPE=U2000
DBHOST=jupiter
DBUSER=ohmura
DBPASSWORD=BeEsXRO
DBNAME=file.db
DBPATH=/home/DS_DB
DBSCHEMA=PUBLIC
```

注）これらの設定は、使用する環境により変更する。
注）*.az, *.fzファイルがない場合にはStep1の前に作成しておく。

Step2 makefileの作成と編集

ACCELLアプリケーションのあるディレクトリに移り、acl2uvユーティリティを実行する。

```
$ acl2uv
```

makefileのvcpl部分を以下のように変更する。

【変更前】

```
-vcpl -warm all -log $*.log -script $?
```

【変更後】

```
-vcpl -warm all -log $*.log -script $? -dcmfile $(DCMFILE)
        -dbconn $(DBCONN)
```

Step3 makeの実行

```
$ make -f vision.mak
```

Step4 visionを起動しStep3で作成したアプリケーションを取り込む。

プロファイルのデータベース・プリファレンスを設定して、接続を確認する。

Step5 Step3で作成されたログ(filename.log)を参照して4GLの編集を行う。
以下の変更なしでもアプリケーションのコンパイルは問題なくできるが、将来にわたってUnifyVISIONを使用することを前提に変更すべき箇所を以下に示す。
ここに示したものは、使用頻度が高いと思われるもの。

UnifyVISIONで変更された4GL文(左側:ACCELL 右側:UnifyVISION)

●APPLCAITION master_form	FORM CLASS master_form
●AFTER APPLICATION	ON DESTROY
●BEFORE FORM	ON CREATE
◎ENABLE ZOOM TO	ENABLE ZOOM TO CHILD FORM form_name OF CLASS form_name
●CHOOSE NEXT FORM	ON CHOOSE NEXT FORM
●ON PREVIOUS FORM	ON DISMISS FORM
●PREVIOUS_FORM	DISMISS_FORM
●NEXT_FIELD	NEXT_FIELD_NAME
●CUR_NEXT_FIELD	CURRENT_NEXT_FIELD_NAME

UnifyVISIONで追加されたもの
◎FIELD SECTIONにBEGIN ENDを入れる

UnifyVISIONで削除されたもの
◎DISPLAY TRIM
◎REFRESH SCREEN
◎REQUIRED FORMS

◎BLINK
◎LOW_INTENSITY
◎REVERSE
◎UNDERLINE

UnifyVISIONで設定が細かくなったもの

AUTO_COMMITについては、追加･削除・更新などについて別々に指定できるようになった。
◎ AUTO_COMMIT　　TX_MODE_ADD_RECORD
　　TX_MODE_DELETE_RECORD
　　TX_MODE_UPDATE_RECORD

SET COMMANDはなくなり、アトリビュート(属性) で指定するようになった。例えば、

◎SET COMMAND “NEXT_FORM”:ACTION TO ‘DISABLED’

は、以下のように変える必要がある。

SET “NEXT_FORM”:AUD_ACTION TO ‘DISABLED’

また、ACTIONについてもAUD_ACTIONとFIND_ACTIONの2つを指定する必要がある。

UnifyVISIONで条件が厳しくなったもの

FINDなど、QUEUE COMMANDに置き換えが必要なものがある。
◎NEXT ACTION IS　　QUEUE COMMAND

以上の変更は、acl2uvを実行する前にシェルスクリプトなどで行っても良い。ただし、ログにはACCELLでは未知のキーワードであるとのメッセージが出力される。

NULLの設定は、NULLを指定して関数を使って変換する。例えば、DATEタイプでは、以下のように行う。

◎SET TODAY_DATE TO str_to_date$(‘**/**/**’)

は、以下のように変更する。

SET TODAY_DATE TO to_date$(NULL)

UnifyVISIONデザイナで修正が必要なもの

◎リターンを押しても次フィールドにいかない
各フィールドの“タブストップ”属性をTRUEに設定する。
(フォーム・プロパティパネルのフィールド順で“タブストップ”をクリックする)

◎マウスでクリックしてフィールド間を移動できない
フォーム属性のフィールド・クリックをTRUEに設定する。
(フォーム・プロパティパネルの対話型操作で“フィールド・クリック”をクリックする)

ACCELL/SQL からUnifyVISION Release2へのコンバージョン(C-Hooks)

ACCELL/SQL インストールディレクトリ　　/home/ASQL
UnifyVISION Release2インストールディレクトリ　/home/VISION2

1. UnifyVISIONユーザ関数(C-Hooks)の相違点

ACCELL/IDSからのコンバージョンの場合には、 “ACCELL/IDSからACCELL/SQL へのコンバージョン(C-Hooks)”の“1. ACCELLユーザ関数(C-Hooks)の相違点”を参照のこと。
ACCELL/SQLとUnifyVISIONは同一仕様。

ただし、インクルードしているchookincl.hをchookinc.hに変更すること。

2. カスタムマネージャの再リンク

Step1 ユーザ関数の変更
上記の対応表をもとに該当箇所をACCELL/SQL(VISION)のコードに置き換える。

Step2 UnifyVISIONカスタムマネージャの再リンク
通常の方法でカスタムマネージャを再リンクする。

例えば、namechk.cというファイルをコンパイルし、カスタムマネージャにリンクするには、以下のようにする。リンク前にはHDATYPEを設定する必要がある。PATHの設定やデータベース固有の設定(例えばORACLE_HOMEなど)もしておく必要がある。

```
$ cc -c -I$UNIFY/.. namechk.c
$ cc -c -I$UNIFY/.. chooktb.c

$ HDATYPE="U2000 ORACLE"; export HDATYPE
$ vision.ld custvisn namechk.o chooktb.o
```

メモリを大量に消費するため、失敗した場合には、以下のようにswapを追加して再度行う。

```
Solaris2.xの例
      # mkfile 30M /export/home0/swap
      # swap -a /export/home0/swap
```

UnifyVISION Release2からRelease3以降の上位リリースへのバージョンアップ

UnifyVISION Release2インストールディレクトリ　　c:¥vision2
UnifyVISION 新リリースインストールディレクトリ　　c:¥vision_new

注意：UnifyVISION 新リリースがインストールされている必要があります。
UnifyVISION Release2はコンバージョン作業には必要ありません。

1. UnifyVISION Releaseアプリケーションの変換

Unify VISION Releaseの開発環境には、Release2のアプリケーションを 新リリースフォーマットに自動的にアップグレードする機能がある。（パーティションに分割されたアプリケーションを除く）
ここでは、その手順について説明する。

Step1 UnifyVISION 新リリースの環境変数の設定を行う

```
PATH=c:¥vision_new¥bin:$PATH
VISION_HOME=c:¥vision_new
LANGDIR=jpn_sjis
DISPLAY=hostname:0.0
HOME=c:¥vis_home
```

（ HOMEはRelease2の開発環境のHOMEを指定する）
注）これらの設定は、使用する環境により変更する。

Step2 新リリースの開発環境を起動する。

Step3 新リリースの開発環境で、Release2のプロジェクトまたはフォルダをオープンすると自動的にアップグレードを行うためのアップグレード・ダイアログが表示される。

Step4 アップグレード・ダイアログの使用方法

ダイアログでは、個々のアプリケーション、ライブラリ、またはリポジトリを段階的にアップグレードするか、1度に全てをアップグレードするかを選択できる。
以下のオプションの中から選択しアップグレードを実行する。
実行後、新リリースのクラス・ライブラリが自動的に作成される。

ｱｯﾌﾟｸﾞﾚｰﾄﾞ(U)	指示されたオブジェクトだけを新リリースのクラス・ライブラリにアップグレードする。 アップグレードされる項目の名前を表示するには、ドロップダウン・リストを使用する。
全てｱｯﾌﾟｸﾞﾚｰﾄﾞ(A)	全ての項目（アプリケーション、リポジトリ、オブジェクト・ライブラリ）を新リリースのクラス・ライブラリにアップグレードする。 アップグレードの後でクラス・ライブラリのアイコンがFolder ウィンドウに表示される。
ｽｷｯﾌﾟ(S)	指示されたオブジェクトをアップグレードしない。

完了(D)　　アップグレード・プロセスを終了する。

Step5 プロファイルの確認
開発環境で、アップグレード後のクラス･ライブラリをオープンし、プロファイルの設定を確認する必要がある。
正しく設定されたプロファイルを選択してから、「ファイルメニュー」→ 「カレントプロファイルにする」を選択しプロファイルをカレントにする。正しいプロファイルがカレントになっていない場合には、次のコンパイルに失敗する可能性がある。

Step6 アップグレード後の全コンパイル
クラス・ライブラリ内の全てのオブジェクトをコンパイルし直す必要がある。
オープンしたクラス・ライブラリに対してウィンドウの「ファイルメニュー」→「構築」→「全て」を実行し、全てのオブジェクトをコンパイルする。

Step7 アップグレード後のRelease2アプリケーションについて
新リリースのクラス･ライブラリへのアップグレードの際、Release2のアプリケーション(*.uva,*.uvl)はそのまま残る。

Step8 アップグレードの再実行
アップグレードが期待通りに行われなかった場合は、Release2のアプリケーションを修正し、アップグレードをやり直すことが可能。2度目のアップグレードを開始するには、アップグレード済みのクラス･ライブラリをフォルダから削除する必要がある。
クラス・ライブラリを削除するには、クラス･ライブラリのアイコンを選択し、「編集」→「削除」を選択する。

２．パーティショニング・アプリケーションの変換

Release2でパーティションに分割されたアプリケーションを新リリースにアップグレードするためには、手作業でアップグレードを実行する必要がある。
ここでは、その手順について説明する。

Step1 UnifyVISION 新リリースの環境変数の設定を行う

```
PATH=c:¥vision_new¥bin:$PATH
VISION_HOME=c:¥vision_new
LANGDIR=jpn_sjis
DISPLAY=hostname:0.0
HOME=c:¥vis_home
```

(HOMEはRelease2の開発環境のHOMEを指定する)

注) これらの設定は、使用する環境により変更する。

Step2 新リリースの開発環境を起動する。

Step3 パーティショニング・アプリケーションに必要な新しい構成要素を追加
1.アプリケーションの各パーティションのプロファイルで、［分散］パネルの［分散機能を有効にする］ダイアログオプションを選択する。

2.［分散］パネルでパーティション・グループ・プロファイルの名前を指定する。パーティション・グループはランタイムにuoroutерユーティリティ・プロセスによって管理される。
3. アプリケーション・ネットワークの全てのホストに対して、ホスト構成ファイルを使用してパーティションの起動コマンドを指定する。
（デフォルト:$VISION_HOME¥lib¥uohost.ini）
4. uohostdユーティリティを使用してグローバル・ネームサービスを起動。
（ uvnamedユーティリティは使用できない）

Step4 手作業によるサービス・クラスのアップグレードの手順

1. 新リリースの関連するクラス･ライブラリのVISIONクラス･ブラウザ･ウィンドウでServiceクラスのサブクラスを作成する。
2. クラスのスクリプトを編集する。
3. サービスにPRIVATE変数があれば、PRIVATEセクションを追加し、_CREATEグローバル関数の各変数をPRIVATEセクションにコピーする。
4. Release2のサービスにコンストラクタ _CREATEがあれば、ON CREATEメソッドを追加し、_CREATEグローバル関数に入っている文をON CREATEメソッドにコピーする。
5. Release2のサービスにデストラクタ_DESTROYがあれば、ON DESTROYメソッドを追加し、_DESTROYグローバル関数に入っている文をON DESTROYメソッドにコピーする。
6. Release2のサービス･インタフェース内で宣言されている関数ごとに同じ名前でMETHODとパラメータ宣言を作成する。メソッドのグローバル関数内の文を新しいサービス･クラスのMETHODにコピーする。
7. Release2サービス内のグローバル関数ごとに（サービス･インタフェースにはない、サービスのローカル関数）新しいサービスクラス内にLOCAL FUNCTIONセクションを作成し、Release2のグローバル関数の内容に対応するローカル関数にコピーする。

３．Release2アプリケーションをランタイムでそのまま使用する

（パーティションに分割しているアプリケーションを除く）

パーティションに分割していないRelease2のUnify VISIONアプリケーションは、新リリースのVISIONランタイム・マネージャでそのまま実行することができる。
アプリケーションを再コンパイルする必要もない。

Step1 UnifyVISION 新リリースの環境変数の設定を行う

```
PATH=c:¥vision_new¥bin:$PATH
VISION_HOME=c:¥vision_new
LANGDIR=jpn_sjis
DISPLAY=hostname:0.0
HOME=c:¥vis_home
```

（ HOMEはRelease2の開発環境のHOMEを指定する）

注）これらの設定は、使用する環境により変更する。

Step2 新リリースの開発環境を起動する。

Step3 Release2のアプリケーション、オブジェクト・ライブラリ、リポジトリ･コンテナは、Release2のアイコンのまま 新リリースの開発環境で表示される。
これらに対して実行できる操作は、カット、コピー、ペースト、削除、エクスポート、実行、デバッグに限られる。

ACCELL/IDSからUnifyVISION Release3以降のリリースへのコンバージョン

ACCELL/SQLインストールディレクトリ　/home/AIDS
UnifyVISION Release　インストールディレクトリ　c:¥vision

ACCELL/IDSからUnifyVISION Release3以降のリリースへのコンバージョンは、1度に行うことができない。
ACCELL/IDS →ACCELL/SQL　→ UnifyVISION Release2 → UnifyVISION 新リリースと３段階のコンバージョン作業を行う必要がある。

1. ACCELL/IDSからACCELL/SQL へのコンバージョン

コンバージョン作業の詳細は、「ACCELL/IDSからACCELL/SQL へのコンバージョン」に関するドキュメントを参照する。

2. ACCELL/SQL からUnify VISION Release2へのコンバージョン

コンバージョン作業の詳細は、「ACCELL/SQL から UnifyVISION Release2へのコンバージョン」に関するドキュメントを参照する。

3. Unify VISION Release2からRelease3以降の上位リリースへのバージョンアップ

コンバージョン作業の詳細は、「UnifyVISION Release2からRelease3以降の上位リリースへのバージョンアップ」に関するドキュメントを参照する。

ACCELL/SQL からUnifyVISION Release3以降のリリースへのコンバージョン

ACCELL/SQLインストールディレクトリ /home/ASQL
UnifyVISION リリースインストールディレクトリ c:¥vision

ACCELL/SQL からUnifyVISION Release3以降のリリースへのコンバージョンは、1度に行うことができない。
ACCELL/SQL → UnifyVISION Release2 → UnifyVISION 新リリースと２段階のコンバージョン作業を行う必要がある。

1. ACCELL/SQL からUnify VISION Release2へのコンバージョン

コンバージョン作業の詳細は、「ACCELL/SQL から UnifyVISION Release2へのコンバージョン」に関するドキュメントを参照する。

2. Unify VISION Release2からRelease3以降の上位リリースへのバージョンアップ

コンバージョン作業の詳細は、「UnifyVISION Release2からRelease3以降の上位リリースへのバージョンアップ」に関するドキュメントを参照する。

UnifyVISION Release3 以降で上位リリースへのバージョンアップ

UnifyVISION 現リリースインストールディレクトリ　　c:¥vision
UnifyVISION 新リリースインストールディレクトリ　　c:¥vision_new

注意：UnifyVISION 新リリースがインストールされている必要があります。
UnifyVISION 現リリースはコンバージョン作業には必要ありません。

1. UnifyVISION 現リリースクラス・ライブラリをそのまま使用する

UnifyVISION Release3以降で開発したクラスライブラリは、アップグレードを行わずにUnify VISION 新リリースの開発環境、及びランタイム・マネージャでそのまま使用できる。
（日本語名オブジェクトを持つクラス・ライブラリを除く）

２.日本語名オブジェクトを持つクラス・ライブラリの変換

UnifyVISION Release3以降で開発したクラス・ライブラリに日本語名のオブジェクト（テーブル名を含む）がある場合は、新リリースでそのまま使用できない。
uvcheck -fixユーティリティを使用して日本語名オブジェクトのタグを修正する必要がある。ここでは、その手順について説明する。

Step1 UnifyVISION 新リリースの環境変数の設定を行う。

```
PATH=c:¥vision_new¥bin:$PATH
VISION_HOME=c:¥vision_new
LANGDIR=jpn_sjis
DISPLAY=hostname:0.0
HOME=c:¥vis_home
```

（ HOMEは現リリースの開発環境のHOMEを指定する）
注）これらの設定は、使用する環境により変更する。

Step2 uvcheck -fixユーティリティで日本語オブジェクトのタグを修正する。
コマンドラインから該当するクラス・ライブラリに対して以下を実行する。
$ uvcheck -fix *.ucl（日本語名のオブジェクトを持つクラス･ライブラリ名）

３.Unify VISION/Webアプリケーションの変換

現リリースのUnify VISION/Webのアプリケーションを新リリースにアップグレードするためには、新リリースの Unify VISION/Web開発環境で、全てのアプリケーションを再コンパイルする必要がある。
ここでは、その手順について説明する。

Step1 UnifyVISION 新リリースの環境変数の設定を行う。

```
PATH=c:¥vision_new¥bin:$PATH
VISION_HOME=c:¥vision_new
LANGDIR=jpn_sjis
DISPLAY=hostname:0.0
```

HOME=c:¥vis_home
(HOMEは現リリースの開発環境のHOMEを指定する)
注) これらの設定は、使用する環境により変更する。

Step2 新リリースの開発環境を起動する。

Step3 新リリースの開発環境で、現リリースで作成したクラス・ライブラリをオープンする。

Step4 Unify VISION/Webのコンパイラ･オプションの指定
クラス・ライブラリ・ウィンドウの「ファイルメニュー」→「コンパイラ・オプション」を実行すると、ライブラリ・コンパイラ・オプション・エディタが表示される。
クラス・ライブラリによってターゲット環境の指定を変更する。

Webクライアント・クラス・ライブラリ:「VISION/Web」を選択する。
「標準Unify VISION」の選択も可能
Webサーバ・クラス・ライブラリ :「標準Unify VISION」のみを選択する。

Step5 プロファイルの確認
オープンしたクラス・ライブラリのプロファイルの設定を確認する必要がある。
正しく設定されたプロファイルを選択してから、「ファイルメニュー」→ 「カレントプロファイルにする」を選択しプロファイルをカレントにする。正しいプロファイルがカレントになっていない場合には、次のコンパイルに失敗する可能性がある。

Step6 クラス・ライブラリの全コンパイル
クラス・ライブラリ内の全てのオブジェクトをコンパイルする。
ウィンドウの「ファイルメニュー」→「構築」→「全て」を実行し、全てのオブジェクトをコンパイルする。