IEC 61850対応プロトコルスタック・ソフトウェア

MMS Lite

サンプルプログラム(scl_srvr)の実装例

実装例1：任意の周期で MMS データ領域へアクセス

第1版

2019/02/07

株式会社日新システムズ
http://www.co-nss.co.jp
京都本社：600-8482 京都府京都市下京区堀川通四条下ル左側 (堀川四条ビル)
東京事務所：101-0024 東京都千代田区神田泉町1番地 (神田泉町ビル)

目次

1. はじめに

1.1 本書について

MMS Lite をご購入いただき、ありがとうございます。

本書は、ユーザーが MMS Lite を使って IED を実装する場合の手順を紹介する資料です。

MMS Lite に付属している基本的なサンプルプログラム(client.c, scl_srvr.c, 等)をベースに説明をします。

付属のサンプルプログラムは全ての事象に対応した物ではなく、必要に応じてユーザーが編集する必要がありますので、実際のユーザーの仕様に合わせて編集する時に参考としていただければ幸いです。

MMS Lite のサンプルプログラムを既にご存知である事を前提にサンプルプログラムを説明しています。

従いまして、サンプルプログラムの使用方法や使用している API 等の説明は行っていないため、開発の際には MMS Lite のドキュメント類、IEC61850規格書を参照してください。

本書は、下記バージョンの SISCO 社の製品向けに作成されました。他のバージョンの製品を使用する場合、異なる部分を置き換えて読んでください。

- MMS Lite V6.2000

1.2 MMS Lite サンプルプログラム(サーバー/クライアント)

MMS Lite は IEC 61850 サーバーとクライアントのサンプルプログラムが用意されています。

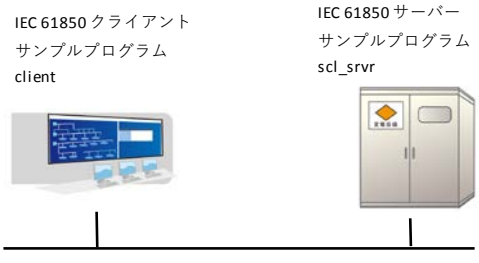
IEC61850サーバーを開発する場合はサーバー用サンプルプログラム(scl_srvr)を、クライアントを開発する場合はクライアント用サンプルプログラム(client)を参考にします。

ただし、サンプルプログラムは全ての機能を実装している物ではありませんので、開発するシステムに応じて設計/開発が必要です。また、サンプルプログラムで実装している内容についても、あくまでサンプルですので他の方法で実装を行っていただく事は問題ありません。

サーバーとクライアントのサンプルプログラムは、以下ディレクトリに用意されています。

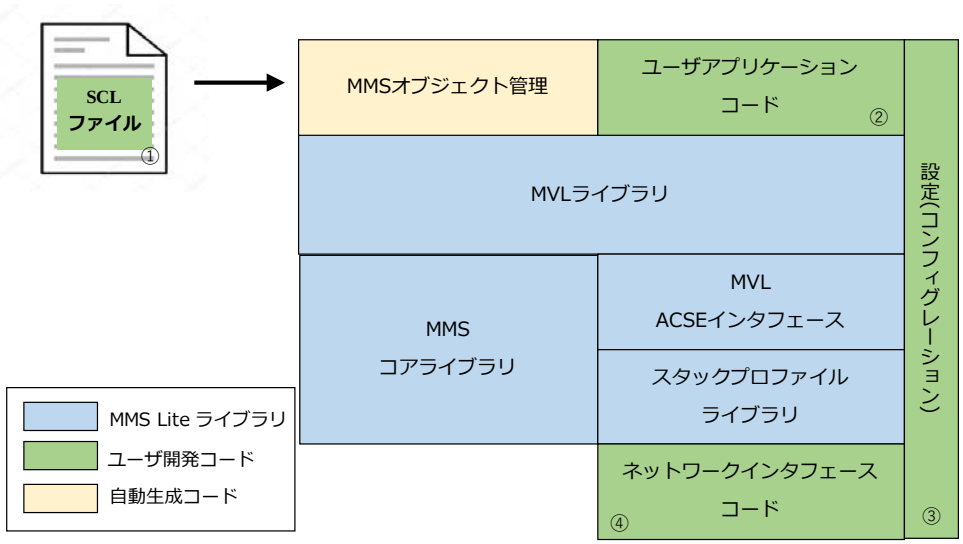
IEC 61850 サーバー {mmslite}/mvl/usr/scl_srvr

IEC 61850 クライアント {mmslite}/mvl/usr/client



1.3 ユーザーの開発対象

MMS-Lite が提供するソフトウェアは下図のような構成です。その中で、ユーザーがプログラム開発を行う場合は基本的には、mmslite/mvl/usr ディレクトリ以下を参照して修正、作成を行います。ユーザーが修正、作成するのは SCL ファイル、ユーザーアプリケーションコード(サンプルアプリケーション)、設定(コンフィグレーション)、ネットワークインタフェースコードが該当します。



① SCL ファイル

SCL ファイルは複数の構成があり MMS-Lite にはサンプルとして複数の SCL ファイル(例：sisco_sample_ed2.scd)が用意されており、IED の機器構成に合わせて編集を行います。SCL ファイルには複数の IED を定義する事はできますが、MMS-Lite サーバーは IED を 1 つのみサポートしていますので、対応する IED を startup.cfg ファイルで定義します。

② ユーザアプリケーションコード

IEC 61850サーバー(scl_srvr)、IEC 61850クライアント(client)のサンプルプログラムを元に、開発するシステムに合わせて修正、新規作成を行います。

③ 設定（コンフィグレーション）

SCL ファイルの他に、ユーザーアプリケーションが起動時に読み込む、起動時設定ファイル(startup.cfg)やネットワーク設定ファイル(osicfg.xml)等の設定を行います。

④ ネットワークインタフェース

MMS-Lite はネットワークプログラミングはソケットを使用していますので、基本的には用意されたプログラムを使用する事ができますが、OS への移植が必要な場合は修正が必要な場合があります。

2. 任意の周期で MMS データ領域へアクセス

MMS Lite に付属している IEC 61850 サーバーのサンプルプログラム(scl_srvr)は、実デバイスから計測値等の値を読み込み管理する仕組みは実装されていません。ユーザーは実デバイスと通信するプログラムを作成して、MMS オブジェクトの属性(stVal, etc.)等の実機器から収集した値を MMS Lite のライブラリが動的に確保したデータ領域(MMS データ領域)に書き込む必要があります。

2.1 アクセス方法

MMS データ領域へアクセスするには下記の方法があります。

- **リーフ関数が呼び出された時にアクセス**

  既存の MMS-Lite サンプルプログラム(scl_srvr)が実装している、リーフ関数(MMS Lite が Request 受信時に呼び出すコールバック関数)が呼び出された時に、MMS データ領域へアクセスする方法です。

  リーフ関数は例えばクライアントからの読み込み要求時に、含まれる各属性(stVal, q, t 等)のそれぞれに対して呼び出されます。

  本資料では、詳細は説明していません。

- **任意の周期でアクセス**

  周期実行のスレッドを作成して、MMS データ領域の各属性毎のアドレスを特定して書き込む方法です。

  リーフ関数が呼び出された時とは違い、どのタイミングで、どの属性に対して書き込むかをユーザーが作成できます。

  本資料では、以降に詳細を説明します。

以降で、２つの処理概要を説明します。

2.1.1 リーフ関数が呼び出された時にアクセスするとは

MMS-Lite に付属している IEC 61850 サーバーのサンプルプログラム(scl_srvr)は、リーフ関数で MMS データを書き込む事を想定しています。

下記は、読み込み用リーフ関数の概要です

```
ST_VOID u_SOME_rd_ind (MVLU_RD_VA_CTRL *mvluRdVaCtrl)
{
(略)
/* ここで、データオブジェクトの書き込み先の mvluRdVaCtrl->primData に、
* センサー値等の、実データを書き込む処理を行います。
* (mvluRdVaCtrl->primData は map_entry->dataPtr と同じアドレス)
*/
(略)
mvlu_rd_prim_done (mvluRdVaCtrl, retcode);
}
```

全てのリーフ関数は、 **MVLU_RD_VA_CTRL *mvluRdVaCtrl** を引数に持ちます。
リーフ関数の中で、呼び出された属性を判断して MMS データ領域に書き込みます。MMS データ領域の該当する書込み先は、 **mvluRdVaCtrl->primData** で示されるアドレスです。

2.1.2 任意の周期でアクセスするとは

リーフ関数が呼び出された時にアクセスするのではなく、ユーザーアプリケーションが任意のタイミングでアクセスする為には、ユーザーが独自に仕組みを実装する必要があります。

ここで紹介する方法はその１例ですので、開発するシステムの仕様に適切な実装を行っていただきますようにお願いいたします。ご参考になれば幸いです。

datamap.cfg に設定したデータマップ情報から MMS データ領域の各属性へのアドレスを特定します。ユーザーアプリケーションは任意のタイミングで、特定したアドレスをアクセスする事が出来ます。

2.1.2.1 datamap.cfg にマッピング情報を設定

データ書き込みを行う為に datamap.cfg で設定したデータマップ情報から書き込み先の MMS オブジェクトへのポインタを特定します。

DATA_MAP 型は usermap.h で以下のように定義されており、 datamap.cfg で設定したユーザテキストは usr_data_info のバッファにコピーされた状態になっています。

DATA_MAP の dataPtr は MMS オブジェクトのデータのアドレスであり、リーフ関数で使用する mvluRdVaCtrl->primData と同じアドレスです。 下記コードの**赤色太字**部分参照。(usermap.h から抜粋)

```
/************************************************************************/
/*                      DATA_MAP structure                              */
/* This structure is used to store each entry from a data mapping       */
/* configuration file.                                                  */
/* Each structure represents 1 line from the file.                      */
/************************************************************************/
typedef struct
  {
  /* NOTE: "ldevice" deleted. Get "ldevice" from DATA_MAP_HEAD.         */
  ST_CHAR leaf [MAX_FLAT_LEN+1];        /* "leaf" name          */
  /* This is a user-defined parameter to help the user to map MMS data  */
  /* to real user data.  This may be replaced with any                  */
  /* parameter that may be useful for mapping data.                     */
  /* In this example it is just a string read from the mapping file.    */
  ST_CHAR usr_data_info [MAX_IDENT_LEN+1];

  ST_VOID *dataPtr;     /* Pointer to data storage for this     */
                        /* leaf. Also used as unique handle     */
                        /* for sorting/finding                  */
  RUNTIME_TYPE *dataType;       /* data type of this leaf       */

  ST_VOID *deadband_info;       /* deadband info for this attr          */
                                /* NULL if this attr not deadbanded     */
                                /* Points to DEADBAND_INT32 or          */
                                /* DEADBAND_FLOAT struct (cast as needed)*/
  ST_INT valKind; /* valKind converted to ST_INT (see SCL_VALKIND_* defines)*/
  } DATA_MAP;
```

例えば、 TotW (有効電力：Active Power) のマッピングは、 datamap.cfg で次の様に設定します。

```
E1Q1SB1C1  MMXU1$MX$TotW$instMag$f  “E1Q1 Total active power”          type=Float
```

1番目の要素が論理デバイス名、 2番目の要素がリーフ名、そして 3番目の要素が「ユーザーテキスト」(ユーザー情報) です。アプリケーションはこの「ユーザーテキスト」をキーに MMS オブジェクトにアクセスを行います。

2.1.2.1 MMS オブジェクトのアドレスを特定

datamap.cfg に設定したユーザーテキスト(例“E1Q1 Total active power”)をキーに、起動時に SCL ファイル及び datamap.cfg を元に展開した領域から該当する MMS オブジェクトのアドレスを特定します。

MMS オブジェクトへのアドレスを特定するには、MVL オブジェクトを指すグローバル変数 MVL_VMD_CTRL mvl_vmd から、MVL_DOM_CTRL **dom_tbl、MVL_VAR_ASSOC **var_assoc_tbl、struct mvl_var_assoc *base_va、DATA_MAP_HEAD *user_info をたどり、 DATA_MAP 型のポインタ配列 map_arr[]から、指定したユーザーテキストと一致する usr_data_info を検索します。一致した場合、*dataPtr が該当する MMS オブジェクトのアドレスです。以下の**赤色太字**参照。(mvl_defs.h から抜粋)

```
/* MVL_VAR_ASSOC                                                */
typedef struct mvl_var_assoc
  {
(略)
  ST_UCHAR flags;               /* MVL_VAR_FLAG_UCA, etc.       */
  MVL_VAR_PROC *proc;           /* User defined pre/post processing */
  ST_VOID *user_info;           /* MVL user can use this for 'whatever' */
  ST_VOID *usr_ind_ctrl;

  struct mvl_var_assoc *va_to_free; /* Used in NVL processing   */

  struct mvl_var_assoc *base_va;    /* VA from which this was derived  */
  ST_INT offset_from_base;          /* Used only for static data buffer */
  ST_UINT prim_num;             /* Index to first prim in RUNTIME_TYPE  */
                                /* (used only when MVL_UCA defined)     */
(略)
  } MVL_VAR_ASSOC;
```

```
/* Domain Scope Objects                                         */
typedef struct mvl_dom_ctrl
  {
(略)
  ST_INT max_num_var_assoc;
  ST_INT num_var_assoc;
  MVL_VAR_ASSOC **var_assoc_tbl;

  ST_INT max_num_nvlist;
  ST_INT num_nvlist;
  MVL_NVLIST_CTRL **nvlist_tbl;
(略)
  } MVL_DOM_CTRL;
```

```
/* VMD Scope Objects                                                */
typedef struct mvl_vmd_ctrl
  {
(略)
  ST_INT max_num_dom;
  ST_INT num_dom;
  MVL_DOM_CTRL **dom_tbl;

  ST_INT max_num_jou;
  ST_INT num_jou;
  MVL_JOURNAL_CTRL **jou_tbl;
(略)
  } MVL_VMD_CTRL;
extern MVL_VMD_CTRL mvl_vmd;
```

2.1.2.2 MMS オブジェクトのアドレスへ書き込む

有効電力の瞬時値 MMXU1MXTotW$instMag$f に計測値を書き込む場合、ユーザテキストの "E1Q1 Total active power" を指定して MMS オブジェクトにアクセスするためのデータマップ情報 (DATA_MAP 型のポインタ) を得るコードを下記の様に実装します。下記コードの**赤色太字**部分参照。(ied_measure.c usermap_set_value_without_update_flag()から抜粋)。

memcpy を使って DATA_MAP へのポインタ(data_map->dataPtr)へ、計測値(value)をデータタイプのサイズ(data_map->dataType->el_size)書き込みます。

```
ST_RET usermap_set_value_without_update_flag(DATA_MAP *data_map, ST_VOID *value)
{
    if (data_map == NULL) {
        MVL_LOG_ERR0("no data map was directed.");
        return SD_FAILURE;
    }

    memcpy(data_map->dataPtr, value, data_map->dataType->el_size);

    return SD_SUCCESS;
}
```

2.2 改造内容

IEC 61850サーバーのサンプルプログラム(scl_srvr)は MMS データ領域を更新しませんので、クライアントへ送信するレポートやクライアントからの Read リクエストの応答に含まれる MMS オブジェクトの属性値は変化しません。

ここでは、修正後のサンプルプログラムのコードを抜粋して概要を説明します。

MMS オブジェクトに任意の周期で書き込む処理を実装する為に、修正する箇所を以下に挙げます。詳詳細は以降の項で説明します。

① **設定ファイル（修正）**

startup.cfg, datamap.cfg を今回の修正に合わせて編集します。

※ここでは、SCL ファイルはデフォルト設定で使用する為、編集は行いません。

② **scl_srvr プロセス（修正）**

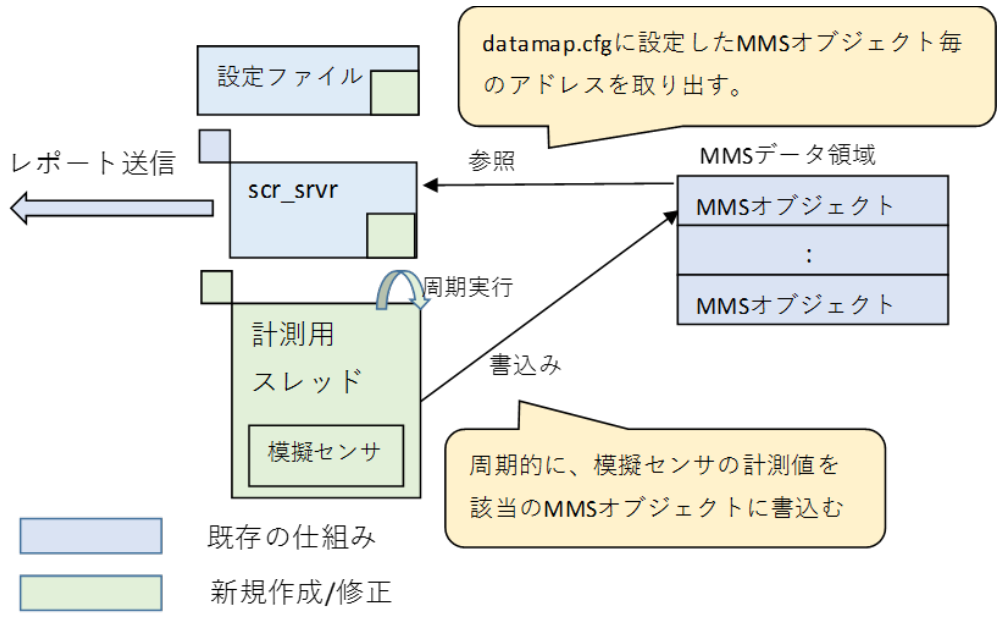
MMS-Lite のパッケージに含まれている IEC 61850サーバーのサンプルプログラムです。

scl_srvr.c に、計測スレッドの起動、MMS データ領域へのアクセスする為の初期化、等を追加します。

③ **計測スレッド(新規作成)**

今回新規作成するスレッド(ied_measure_thread)です。

scl_srvr からスレッドを生成して、周期的に模擬センサーから計測値を読み取り、MMS オブジェクト毎に値を書き込みます。



2.2.1 各種設定ファイルを修正する

startup.cfg, datamap.cfg を今回の修正に合わせて編集します。

＊ここでは、SCL ファイルはデフォルト設定で使用する為、編集は行いません。

2.2.1.1 startup.cfg ファイル

startup.cfg ファイルは、サーバー起動時に与えるパラメータを設定しています。今回は使用する SCL ファイルを変更しますので、SCLFileName の設定を sisco_sample.cid から sisco_sample¥ed2.scd に変更します。下記**赤色太字**を参照。

```
#SCLFileName    sisco_sample.cid
#NOTE: Use this SCL file instead for IEC 61850 Edition 2.
SCLFileName     sisco_sample_ed2.scd
IEDName E1Q1SB1
AccessPointName S1
ReportScanRate  2.0

#NOTE: BRCBBufferSize was not configurable before. Now it is.
BRCBBufferSize  10000

#NOTE: The old function "scl2_ld_create_all" ignores LogScanRateSeconds and LogMaxEntries.
#      You must use the new function "scl2_ld_create_all_scd" to use these.
LogScanRateSeconds      2.0
LogMaxEntries   1000
```

コメントアウトする

コメントアウトを解除する

2.2.1.2 datamap.cfg ファイル

datamap.cfg は、サーバーのサンプルプログラム(scl_srvr)が使用する MMS オブジェクト属性と実データとをマッピングする為に使用するファイルです。今回は MMS オブジェクトの属性と実データとのマッピングの設定を、datamap.cfg の３桁目にある”ユーザー定義”を使用します。その為に、下記のように”ユーザー定義”を編集します。下記**赤色太字**を参照。

```
E1Q1SB1C1   MMXU1$MX$TotW$instMag$f     “E1Q1 Total active power”              type=Float
E1Q1SB1C1   MMXU1$MX$TotW$q            “E1Q1 Total active power (Quality)”    type=BVstring13
E1Q1SB1C1   MMXU1$MX$TotW$t            “E1Q1 Total active power (Timestamp)”  type=Utctime
E1Q1SB1C1   MMXU1$MX$TotVAr$instMag$f  “E1Q1 Total reactive power”            type=Float
E1Q1SB1C1   MMXU1$MX$TotVAr$q          “E1Q1 Total reactive power (Quality)”  type=BVstring13
E1Q1SB1C1   MMXU1$MX$TotVAr$t          “E1Q1 Total reactive power (Timestamp)”       type=Utctime
```

datamap.cfg の３桁目(ユーザー定義)にはデフォルトで"dummy_data_mapping_string"(もしくは、"my data mapping string")が定義されています。これを、ユーザーがプログラムで使用する為に任意の文字列を設定して、これをキーとしてオブジェクト名から該当する MMS データ領域へのポインタを取得します。

ユーザー定義がデフォルトの場合は、マッピング情報を取得できませんので、書き込み対象から除外します。

2.2.2 scl_srvr プロセスを修正する

scl_srvr.c に、計測スレッドの起動、MMS データへのアクセス処理を準備する処理の呼び出し、を追加します。

2.2.2.1 データマップを利用したオブジェクトアクセスの初期化

MMS データ領域へユーザーが任意のタイミングで直接書き込む為に、各 MMS オブジェクトの属性毎に確保された領域へのポインタを事前に抽出しておき、必要な時に書き込む処理を用意します。

MMS Lite ではメインループ内で、MMS の送受信、レポート、ログ等のサービスを実行する為に定期的に処理を繰り返し実行しています。初期処理を追加する場合は、このメインループが始まる前に追加します。今回は SCL ファイルの構文解析を行いオブジェクトを作成した後に、データマップを利用したオブジェクトアクセス処理の初期化 usermap_init_obj_access()を追加します。下記の**赤色太字**を参照。(scl_srvr.c main()から抜粋)

```
  if (ret == SD_SUCCESS)
    {
    if (mvl_vmd.num_dom == 0)
      {
      printf ("¥n ERROR: SCL parse OK, but no Logical Devices created. Check configuration.");
      exit (1);
      }
    /* Log all SCL info stored.                          */
    /* NOTE: this only logs if SXLOG_DEC enabled.        */
    scl_log_all (&scl_info);
    }

  /* データマップを利用したオブジェクトアクセスを初期化する */
  if (usermap_init_obj_access() != SD_SUCCESS)
    {
    printf ("¥n ERROR: Can't make resources for accessing data with usermap.¥n");
    exit (1);
    }
```

1）データマップの探索用変数の定義

データマップのユーザー情報からの検索用のインデックスとデータマップのユーザー情報からの検索用のインデックスのデータ数を格納する変数を定義します。下記の**赤色太字**を参照(usermap_opt_access.c から抜粋)

```
/**
 * データマップのユーザー情報からの検索用のインデックス
 */
static DATA_MAP **usermap_sorted_user_text_index = NULL;
/**
 * データマップのユーザー情報からの検索用のインデックスのデータ数
 */
static ST_UINT usermap_sorted_user_text_index_size = 0;
```

2）データマップ(DATA_MAP)の２分探査準備

データ属性(リーフ)毎の MMS データ領域と datamap.cfg のユーザー定義文字列をマッピングする為に、２分探査用のインデックスを作成します。

datamap.cfg を元に、データ属性(リーフ)毎に動的に確保されたデータマップ(DATA_MAP)の数を数えます。以降、データマップを数えるときの最大数として使用します。下記の**赤色太字**参照

(usermap_opt_access.c usermap_init_obj_access()から抜粋)

```
ST_RET usermap_init_obj_access(ST_VOID)
{
    ST_INT didx;
    ST_INT vidx;
    MVL_DOM_CTRL *dom;
    MVL_VAR_ASSOC *var;
    DATA_MAP_HEAD *map_head_sorted;
    DATA_MAP *map_entry;
    ST_UINT j;
    int count = 0;

(略)
    /* 最初に、メモリ確保の必要な容量を知るためにデータマップの数を確認する */
    count = 0;
    for (didx = 0; didx < mvl_vmd.num_dom; didx++) {
        dom = mvl_vmd.dom_tbl [didx];
        for (vidx = 0; vidx < dom->num_var_assoc; vidx++) {
            var = dom->var_assoc_tbl[vidx];
            map_head_sorted = (DATA_MAP_HEAD *) var->user_info;
            for (j = 0; j < map_head_sorted->map_count; j++) {
                map_entry = map_head_sorted->map_arr[j];
                if (map_entry) {
                    count++;
                }
            }
        }
    }
```

3）ユーザー情報をキーに２分探索用インデックスのメモリを確保

usermap_sorted_user_text_index に２分探索用インデックスを格納する為に、SISCO ライブラリの chk_calloc()を用いて、データマップ(DATA_MAP)を datamap.cfg を元に作成した個数分の領域を確保します。

(usermap_opt_access.c usermap_init_obj_access())

```
    /* ユーザー情報からの検索の 2分探査用のインデックスのメモリを確保する */
    usermap_sorted_user_text_index = chk_calloc(count, sizeof(DATA_MAP *));
    if (usermap_sorted_user_text_index == NULL) {
        MVL_LOG_ERR1("Can't allocate memory for %d bytes.", count * sizeof(DATA_MAP *));
        usermap_cleanup_obj_access();
        return SD_FAILURE;
    }
```

4）必要なデータマップのデータを２分探索用のインデックスにセットする

マッピング用ユーザー情報を設定したデータマップを、２分探査用のインデックステーブルにセットします。

MMS のオブジェクト VMD(Virtual Manufacturing Device)は、IEC61850ではロジカルデバイス(ド

メイン)として使用しています。「ロジカルデバイス」（ドメイン）へのアドレス（dom）を取り出し、「ロジカルノード」へのアドレス（var）を取り出し、メンバ user_info に設定してある DATA_MAP_HEAD 構造体へのアドレス（map_head_sorted）を取り出し、DATA_MAP 構造体へのポインタ配列から datamap.cfg のユーザー定義にマッピング情報を設定した有効なマップ情報(map_entry)を取り出し、２分探索用インデックス配列に登録します。下記の**赤色太字**参照

(usermap_opt_access.c usermap_init_obj_access()から抜粋)

```
    /*「ロジカルデバイス」（ドメイン）の全てをループで見る */
    for (didx = 0; didx < mvl_vmd.num_dom; didx++) {
        dom = mvl_vmd.dom_tbl [didx];

        /*「ロジカルノード」の全てをループで見る */
        for (vidx = 0; vidx < dom->num_var_assoc; vidx++) {
            var = dom->var_assoc_tbl[vidx];
            map_head_sorted = (DATA_MAP_HEAD *) var->user_info;

            /* 属性のデータ（マップ情報）の全てをループで見る */
            for (j = 0; j < map_head_sorted->map_count; j++) {
                map_entry = map_head_sorted->map_arr[j];
                /* ダミーデータを除き、有用なデータマップへののポインタを設定する */
                if (map_entry) {
                    /* ユーザー情報からの検索用インデックスを追加する */
                    if (strcmp(map_entry->usr_data_info, "dummy_data_mapping_string") != 0
                        && strlen(map_entry->usr_data_info) > 0) {
                        usermap_sorted_user_text_index[usermap_sorted_user_text_index_size] = map_entry;
                        usermap_sorted_user_text_index_size++;
                    }
                    /* データマップ中の検索用インデックス番号を初期化する */
                    map_entry->usermap_sorted_index = -1;
(略)
                }
            }
        }
    }
```

５） ２分探索用インデックスをソートする

データマップのユーザー情報からの検索用のインデックスをソートします。以下の**赤色太字**参照

(usermap_opt_access.c usermap_init_obj_access()から抜粋)

```
    /* データマップのユーザー情報からの検索用のインデックスをソートする */
    qsort(usermap_sorted_user_text_index,
          usermap_sorted_user_text_index_size,
          sizeof(DATA_MAP *),
          usermap_user_text_cmpstr);
    /* データマップが、検索用インデックス中のどの位置にあるかを記録する */
    /* これは、同じユーザー情報を持つものに値を設定する目的で、インデックス中の */
    /* 前後のデータへアクセスするために使用する。 */
    for (j = 0; j < usermap_sorted_user_text_index_size; j++) {
        map_entry = usermap_sorted_user_text_index[j];
        map_entry->usermap_sorted_index = j;
    }
(略)
```

```
    return SD_SUCCESS;
}
```

２分探索用インデックスのソートする qsort()と探索する bsearch()用の比較関数 usermap_user_text_cmpstr()は、探索用インデックス配列に格納した DATA_MAP 構造体へのポインタを比較します。以下の**赤色太字**参照(usermap_opt_access.c sermap_user_text_cmpstr()から抜粋)

```
static ST_INT usermap_user_text_cmpstr(const ST_VOID *data1, const ST_VOID *data2)
{
    return strcmp((*(DATA_MAP **)data1)->usr_data_info, (*(DATA_MAP **)data2)->usr_data_info);
}
```

2.2.2.2 計測処理の開始

IED の模擬計測処理を行う周期実行のスレッド(計測スレッド)を起動します。

IED サンプルの計測処理(周期実行のスレッド起動) ied_start_measure()の呼び出しは、メインループの前に追加します。下記の**赤色太字**参照。(scl_srvr.c main()から抜粋)

ied_measure.c の ied_start_measure()では、模擬センサーデータの確保、データ属性毎の MMS データ領域を抽出する為の DATA_MAP 領域へのポインタを取得、周期実行用のイベントセマフォ作成、周期実行用のスレッド起動を行います。

```
#if defined(GOOSE_TX_SUPP)
  TimeOld = sGetMsTime ();      /* init old time        */
#endif
#if defined(GOOSE_RX_SUPP)
  GooseRxTimeOld = sGetMsTime ();      /* init old GOOSE RX time        */
#endif
  /* IED サンプルの計測処理を開始する */
  ret = ied_start_measure();
  if (ret != SD_SUCCESS)
    {
    printf ("Can't start measure of IED sample.¥n");
    exit (99);
    }
(略)
  while (doIt)
    {
    timeNow = sGetMsTime ();
```

2.2.2.1 模擬センサーの作成

センサーデータ領域は sim_sensor.h で定義しており、value(センサーの値)と正方向と負方向の触れ幅(パーセント)の領域を確保ています。(sim_sensor.h から抜粋)

```
typedef struct sim_sensor_t {
    ST_DOUBLE value;  /* センサーの値 */

    ST_DOUBLE upper_rate;  /* 正方向（プラス）方向の触れ幅（パーセント）*/
    ST_DOUBLE lower_rate;  /* 負方向（マイナス）方向の触れ幅（パーセント）*/
} SIM_SENSOR;
```

模擬センサーの領域確保は sim_sensor_create() で行います。第1引数で計測値の初期値(POWER_INITIAL_VALUE,)を設定します。複数の模擬センサーを作成する場合は、模擬センサー分 sim_sensor_create()を実行して模擬センサー毎に戻り値の SIM_SENSOR 型へのポインタを保管します。ここでは、有効電力センサー(Active_power_sensor)と無効電力センサー(Reactive_power_sensor)の２つを作成します。(ied_measure.c ied_start_measure()から抜粋)

```
ST_RET ied_start_measure(ST_VOID)
{
    ST_RET ret = SD_FAILURE;

    /* 有効電力センサーを作成する */
    Active_power_sensor = sim_sensor_create(POWER_INITIAL_VALUE,
                                            POWER_UPPER_JITTER_RATE,
                                            POWER_LOWER_JITTER_RATE);
    if (Active_power_sensor == NULL) {
        MVL_LOG_ERR0("Can't make sensor.");
        return SD_FAILURE;
    }

    /* 無効電力センサーを作成する */
    Reactive_power_sensor = sim_sensor_create(POWER_INITIAL_VALUE,
                                              POWER_UPPER_JITTER_RATE,
                                              POWER_LOWER_JITTER_RATE);
    if (Reactive_power_sensor == NULL) {
        MVL_LOG_ERR0("Can't make sensor.");
        return SD_FAILURE;
    }
```

2.2.2.1 MMS オブジェクトへのアドレスを取得

データを書き込む MMS オブジェクトへのアドレスを datamap.cfg で設定しているユーザーテキストをキーに取得します。下記の**赤色太字**参照(ied_measure.c ied_start_measure()から抜粋)

```
/* データを書き込むオブジェクトの情報を datamap.cfg で設定している
     * ユーザテキストをキーに得る */
    Data_map_active = usermap_find_by_user_text(USER_INFO_E1Q1_ACTIVE_POWER);
    Data_map_active_Quality = usermap_find_by_user_text(USER_INFO_E1Q1_ACTIVE_POWER_QUALITY);
    Data_map_active_Timestamp = usermap_find_by_user_text(USER_INFO_E1Q1_ACTIVE_POWER_TIMESTAMP);
    Data_map_reactive = usermap_find_by_user_text(USER_INFO_E1Q1_REACTIVE_POWER);
    Data_map_reactive_Quality = usermap_find_by_user_text(USER_INFO_E1Q1_REACTIVE_POWER_QUALITY);
    Data_map_reactive_Timestamp = usermap_find_by_user_text(USER_INFO_E1Q1_REACTIVE_POWER_TIMESTAMP);
```

MMS オブジェクトへのアドレスを datamap.cfg で設定しているユーザーテキストをキーに取得する関

数 usermap_find_by_user_text()は、ユーザーテキストをキーにインデックス配列から 2 分探索で検索（bsearch）を行い、指定した属性に該当する MSS オブジェクトへのアドレスを取得します。以下の**赤色太字**参照。(usermap_obj_access.c usermap_find_by_user_text())

```
DATA_MAP *usermap_find_by_user_text(ST_CHAR *user_text)
{
    DATA_MAP key_data;
    DATA_MAP *key = &key_data;
    DATA_MAP **found = NULL;

    /* ユーザテキストをキーに、データを二分探査で検索を行う */
    if (strlen(user_text) + 1 > sizeof(key->usr_data_info)) {
        MVL_LOG_ERR1("Too long user text '%s'.", user_text);
        return NULL;
    }
    strncpy(key->usr_data_info, user_text, sizeof(key->usr_data_info));
    found = bsearch(&key,
                    usermap_sorted_user_text_index,
                    usermap_sorted_user_text_index_size,
                    sizeof(DATA_MAP *),
                    usermap_user_text_cmpstr);
    if (found == NULL) {
        return NULL;
    }

    return *found;
}
```

2.2.2.2 周期実行用イベントセマフォを作成

計測用の周期実行スレッドが周期的に動作する為に、イベントセマフォを SISCO ライブラリ gs_get_event_sem()で作成します。下記の**赤色太字**参照(ied_measure.c ied_start_measure()から抜粋)

```
/* Sleep 用のセマフォ作成 */
ied_measure_event_sem = gs_get_event_sem (SD_FALSE);
if (ied_measure_event_sem == NULL) {
    MVL_LOG_ERR0("Can't make semaphore for measure.");
    return SD_FAILURE;
}
```

2.2.2.3 計測スレッドの作成

SISCO ライブラリ gs_start_thread()で計測用スレッド(ied_measure_thread)を作成します。下記の**赤色太字**参照(ied_measure.c ied_start_measure()から抜粋)

```
/* 実行のフラグを設定 */
ied_measure_thread_stop = SD_FALSE;

/* 実行用のスレッドを作成 */
ret = gs_start_thread(ied_measure_thread,
                      NULL,
                      &ied_measure_thread_handle,
```

```
                        &ied_measure_thread_id);
    if (ret != SD_SUCCESS) {
        MVL_LOG_ERR0("Can't start thread for measure.");
        gs_free_event_sem(ied_measure_event_sem);
        return SD_FAILURE;
    }

    return SD_SUCCESS;
```

2.2.3 計測スレッド(ied_measure_thread)

MMS オブジェクトの属性(stVal, etc.)等の値を実機器から読み込む代わりに、模擬センサーを用意して、周期的に模擬センサーから読み込んだ値を MMS データ領域へ書き込む処理を呼び出します。

2.2.3.1 任意の周期時間を待つ

計測スレッド(ied_measure_thread)は、停止用フラグ(ied_measure_thread_stop)が偽の間は処理を継続します。周期時間はイベントセマフォ(SISCO ライブラリ gs_wait_event_sem)のタイムアウト機能をつかって実施しています。タイムアウト値は gs_wait_event_sem の第2引数(ied_measure_period_ms)に指定しています。下記の**赤色太字**参照(ied_measure.c ied_start_measure()から抜粋)

```
static ST_THREAD_RET ST_THREAD_CALL_CONV ied_measure_thread(ST_THREAD_ARG arg)
{
    ST_RET ret = SD_FAILURE;

    /* スレッドのループ */
    while (!ied_measure_thread_stop) {
        /* 次の計測周期まで待つ */
        ret = gs_wait_event_sem(ied_measure_event_sem, ied_measure_period_ms);
        if (ret != SD_SUCCESS && ret != SD_TIMEOUT) {
            MVL_LOG_ERR0("measure : wait event error.");
            break;
        }
```

2.2.3.1 計測値の読み込みと書き込み

指定した周期に達した時に、定期的な計測の処理を行う ied_do_measure()を呼び出して模擬センサーから計測値を読み込み MMS オブジェクトの値を更新します。下記の**赤色太字**参照(ied_measure.c ied_start_measure()から抜粋)

```
        /* 待っている間に終了フラグが設定されているかのチェック */
        if (ied_measure_thread_stop) {
            break;
        }

        /* 計測を行い MMS オブジェクトを更新する */
        ied_do_measure();
    }

    return ST_THREAD_RET_VAL;
}
```

ied_do_measure()は、電力の計測を行う ied_measure_total_power()を呼び出します。下記の**赤色太字**参照(ied_measure.c ied_do_measure ()から抜粋)

```
static ST_VOID ied_do_measure(ST_VOID)
                {
    /* 電力の計測を行う */
    ied_measure_total_power();
}
```

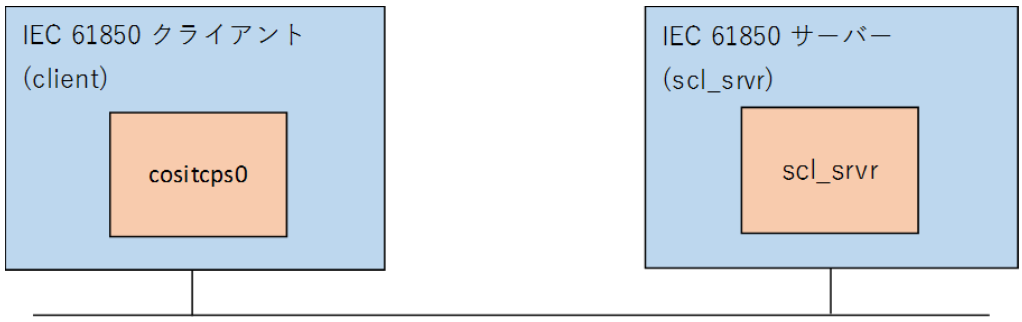
ied_measure_total_power()は、模擬センサーから計測値を読み込み、その値を MMS オブジェクトへ書き込みます。下記の**赤色太字**参照(ied_measure.c ied_measure_total_power()から抜粋)

```
static ST_VOID ied_measure_total_power(ST_VOID)
{
    ST_FLOAT active;  /* 有効電力 (Active power) */
    ST_FLOAT reactive;  /* 無効電力 (Reactive power) */
    QUALITY_BVSTRING quality;
    MMS_UTC_TIME timestamp;

        /* 計測 (センサーからの値の読み込み) */
    active = (ST_FLOAT)sim_sensor_get_value(Active_power_sensor);
    reactive = (ST_FLOAT)sim_sensor_get_value(Reactive_power_sensor);

    /* データオブジェクトに書き込むデータの Quality の値を得る
     * (※このプログラムはサンプルのため、常に Good を設定します。
     *     他の値を設定する方法は iec_data.h のコメントを参照してください。)
     */
    quality.len = IEC_QUALITY_BIT_LEN;
    quality.data[0] = 0;
    quality.data[1] = 0;
    BSTR_SET_QUALITY_GOOD(quality.data);

    /* データのタイムスタンプの値を得る
     * (※計測時間として現在時刻を使用しています。)
     */
    ied_get_current_timestamp(&timestamp);

    /*** 以下は MMS のオブジェクトのデータの値を書き込む処理 ***/

    /* このスレッドが値を更新している途中に送信が処理されないように
     * IEC 61850 サーバのプロトコル処理をロックする
```

```
     */
    gs_mutex_get(&Srvr_mutex);

    /* 有効電力 (Active Power の値の書き込み */
    usermap_set_value(Data_map_active, &active);
    usermap_set_value(Data_map_active_Quality, &quality);
    usermap_set_value(Data_map_active_Timestamp, &timestamp);

    /* 無効電力 (Reactive Power の値の書き込み */
    usermap_set_value(Data_map_reactive, &reactive);
    usermap_set_value(Data_map_reactive_Quality, &quality);
    usermap_set_value(Data_map_reactive_Timestamp, &timestamp);

    /* IEC 61850 サーバのプロトコル処理のロックを解除する */
    gs_mutex_free(&Srvr_mutex);
}
```

2.3 プログラムの実行

サンプルプログラムの client と scl_srvr の構成で使用します。２台の PC を使用した場合の構成は下図の通りですが、１台の PC でも両方を動作させる事が可能です。



2.3.1 scl_srvr のビルド

既存のファイルを直接編集する場合は、ビルド環境の編集は不要ですが、新しくファイルを追加する場合はビルド環境の編集が必要です。

{mmslite}/cmd/以下の OS 毎にあるディレクトリ(gnu, vs2010, win32)がありますので、Linux の場合は gnu 以下の関連ファイルを編集して、ビルドします。

Linux(64bit)の場合は、下記コマンドでビルドします。

```
$ ./mmslite802.sh LINUX 64
```

ビルド環境の編集）

今回のサンプルプログラム編集では、新規作成した下記ファイルを scl_srvr.mak ファイルに追加します。

ied_measure.c	計測に関する処理
usermap_obj_access.c	データマップによるオブジェクト指定に関する処理
sim_sensor.c	模擬センサーに関する処理

・OBJECTS に追加オブジェクトファイルを追加します。

```
# Define objects used in the exectable
OBJECTS = ¥
        $(OBJDIR)/db_61850.o ¥
        $(OBJDIR)/scl_srvr.o ¥
        $(OBJDIR)/startup.o ¥
        $(OBJDIR)/mmsop_en.o ¥
        $(OBJDIR)/mvlop_en.o ¥
        $(OBJDIR)/mvl_acse.o ¥
        $(OBJDIR)/event2.o ¥
        $(OBJDIR)/logcfgx.o ¥
        $(OBJDIR)/uca_obj.o ¥
        $(OBJDIR)/userleaf.o ¥
        $(OBJDIR)/userleaf2.o ¥
        $(OBJDIR)/userleaf_beh.o ¥
        $(OBJDIR)/userleaf_health.o ¥
```

```
        $(OBJDIR)/userwrite.o ¥
        $(OBJDIR)/reject.o ¥
        $(OBJDIR)/slogipc.o ¥
        $(OBJDIR)/dib_list.o ¥
        $(OBJDIR)/scl_dib.o ¥
        $(OBJDIR)/usermap.o ¥
        $(OBJDIR)/u_ifile.o ¥
        $(OBJDIR)/ied_measure.o ¥
        $(OBJDIR)/usermap_obj_access.o ¥
        $(OBJDIR)/sim_sensor.o ¥
```

新規作成ファイルを追加

・オブジェクトファイルのターゲット依存関係を追加します。

```
# Object file target dependencies. All use the pattern rule above.
$(OBJDIR)/db_61850.o:   db_61850.c $(INCLUDES)
$(OBJDIR)/scl_srvr.o:   scl_srvr.c $(INCLUDES)
$(OBJDIR)/startup.o:    startup.c $(INCLUDES)
$(OBJDIR)/mmsop_en.o:   mmsop_en.c $(INCLUDES)
$(OBJDIR)/mvlop_en.o:   mvlop_en.c $(INCLUDES)
$(OBJDIR)/mvl_acse.o:   mvl_acse.c $(INCLUDES)
$(OBJDIR)/event2.o:     event2.c $(INCLUDES)
$(OBJDIR)/logcfgx.o:    logcfgx.c $(INCLUDES)
$(OBJDIR)/uca_obj.o:    uca_obj.c $(INCLUDES)
$(OBJDIR)/userleaf.o:   userleaf.c $(INCLUDES)
$(OBJDIR)/userleaf2.o:  userleaf2.c $(INCLUDES)
$(OBJDIR)/userleaf_beh.o:       userleaf_beh.c $(INCLUDES)
$(OBJDIR)/userleaf_health.o:    userleaf_health.c $(INCLUDES)
$(OBJDIR)/usermap.o:    usermap.c $(INCLUDES)
$(OBJDIR)/userwrite.o:  userwrite.c $(INCLUDES)
$(OBJDIR)/reject.o : reject.c $(INCLUDES)
$(OBJDIR)/slogipc.o : $(MMSSRCDIR)/slogipc.c $(INCLUDES)
$(OBJDIR)/dib_list.o:   dib_list.c $(INCLUDES)
$(OBJDIR)/scl_dib.o:    scl_dib.c $(INCLUDES)
$(OBJDIR)/u_ifile.o :   u_ifile.c $(INCLUDES)
$(OBJDIR)/ied_measure.o :       ied_measure.c $(INCLUDES)
$(OBJDIR)/usermap_obj_access.o :        usermap_obj_access.c $(INCLUDES)
$(OBJDIR)/sim_sensor.o :        sim_sensor.c $(INCLUDES)
```

新規作成ファイルを追加

2.3.2 scl_srvr の実行

今回のサーバーサンプルプログラム scl_srvr は、IEC61850 Edition2の SCL ファイルを使うので、実行時の引数に “-m scd” を指定します。

scl_srvr はルート権限で実行する必要があります。(以下は、LinuxOS Ubuntu の場合です)

```
$ sudo ./scl_srvr_ld -m scd
```

2.3.3 client の実行

今回のクライアントサンプルプログラム client は、レポートの受信をテストするモード (TEST_MODE_IEC_RPT)を使用するので、起動時引数に”-m iecrpt”を指定します。

```
$ ./cositcps0_ld -m iecrpt
```

2.4 計測値の確認

2.4.1.1 パケットキャプチャで確認

IEC61850サーバー(scl_srvr)から定期的に送信するレポートをパケットキャプチャ(Wireshark)で確認します。レポートパケットはパケット一覧には下記の様に”unconfirmed-PDU”と表示します。

パケット詳細を展開すると下記の様に表示します。

datamap.cfg でマッピング情報としてユーザー定義を記述した” MMXU1MXTotVAr$instMag$f”は、SCL ファイルに定義した DataSet の構成から、パケット詳細の”floating-point: 084245a135”が該当する事が分かります。模擬データとして周期的にこの値が変化している事が分かります。

(datamap.cfg)

```
E1Q1SB1C1  MMXU1$MX$TotVAr$instMag$f     “E1Q1 Total reactive power”          type=Float
```

(パケット詳細)

```
unconfirmed-PDU
    unconfirmedService: informationReport (0)
        informationReport
            variableAccessSpecification: variableListName (1)
                variableListName: vmd-specific (0)
                    vmd-specific: RPT
            listOfAccessResult: 13 items
                AccessResult: success (1)
                    success: visible-string (10)
                        visible-string: E1Q1Measurands
                AccessResult: success (1)
                    success: bit-string (4)
                        Padding: 6
                        bit-string: 7c80
                AccessResult: success (1)
                    success: unsigned (6)
                        unsigned: 2
                AccessResult: success (1)
                    success: binary-time (12)
                        binary-time: Aug 31, 2018 07:26:24.478000000 UTC
                AccessResult: success (1)
                    success: visible-string (10)
                        visible-string: E1Q1SB1C1/LLN0$Measurands
                AccessResult: success (1)
                    success: unsigned (6)
                        unsigned: 0
                AccessResult: success (1)
                    success: bit-string (4)
                        Padding: 6
                        bit-string: ff
                            [Expert Info (Warn/Undecoded): Bits set in padded area: 0x3f]
                                [Message: Bits set in padded area: 0x3f]
                                [Severity level: Warn]
                                [Group: Undecoded]
                AccessResult: success (1)
                    success: visible-string (10)
```

レポート制御ブロック属性の該当項目を記載する

(RptID)

(OptFlds)

(SqNum)

(RptTim)

(DataSet)

(Inclusion)

```
            visible-string: E1Q1SB1C1/MMXU1$MX$TotW$mag
    AccessResult: success (1)
        success: visible-string (10)
            visible-string: E1Q1SB1C1/MMXU1$MX$TotVAr
    AccessResult: success (1)
        success: structure (2)
            structure: 2 items
                Data: integer (5)
                    integer: 0                                  (Value:i)
                Data: floating-point (7)
                    floating-point: 08425d6992                  (Value:f)
    AccessResult: success (1)
        success: structure (2)
            structure: 5 items
                Data: structure (2)
                    structure: 2 items
                        Data: integer (5)
                            integer: 0                          (Value:intMag:i)
                        Data: floating-point (7)
                            floating-point: 084245a135          (Value:intMag:f)
                Data: structure (2)
                    structure: 2 items
                        Data: integer (5)
                            integer: 0                          (Value:mag:i)
                        Data: floating-point (7)
                            floating-point: 084245a135          (Value:mag:f)
                Data: integer (5)
                    integer: 0                                  (range)
                Data: bit-string (4)
                    Padding: 3
                    bit-string: 0000                            (q)
                Data: utc-time (17)
                    utc-time: Oct 18, 2019 20:56:32.001950383 UTC   (t)
    AccessResult: success (1)
        success: bit-string (4)
            Padding: 2
            bit-string: 08
    AccessResult: success (1)
        success: bit-string (4)
            Padding: 2
            bit-string: 08
```

E1Q1SB1C1/MMXU1MXTotVAr
の各属性を記載する

周期的に変化

2.5 関連情報

2.5.1 SCL ファイル

今回サンプルプログラムが使用する SCL ファイルは、IED61850 Edition2の SCL ファイルである sisco_sample_ed2.scd を使用します。SCL ファイルの編集は不要です。

なお、MMS-Lite サンプルのデフォルトは異なる SCL ファイル(sisco_sample.cid)を使用していますので、startup.cfg を編集してください。

以下は、レポート送信に関連する箇所を参考までに記述します。

・レポート送信対象の DataSet に項目を設定している箇所

```
<DataSet name="Measurands">
    <FCDA ldInst="C1" prefix="" lnInst="1" lnClass="MMXU" doName="TotW" daName="mag" fc="MX"/>
    <FCDA ldInst="C1" prefix="" lnInst="1" lnClass="MMXU" doName="TotVAr" fc="MX"/>
</DataSet>
```

・レポート送信の対象を設定している箇所

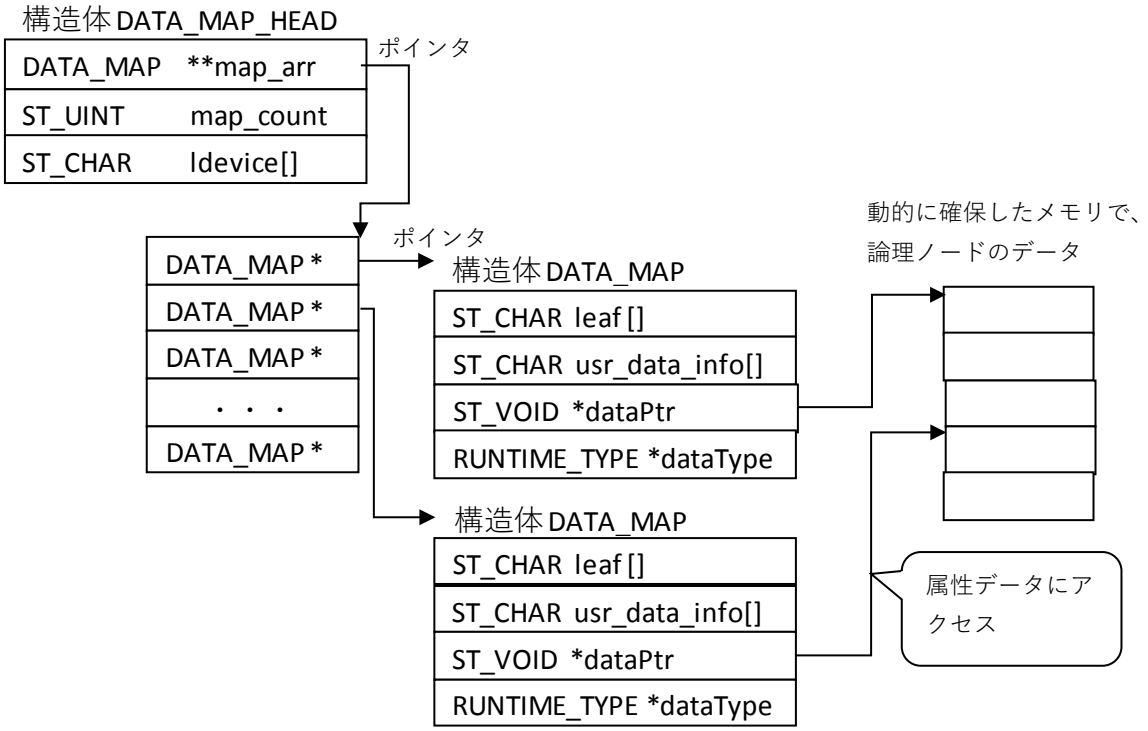
```
<ReportControl name="MeaReport" rptID="E1Q1Measurands" datSet="Measurands" intgPd="2000" confRev="0">
    <TrgOps dchg="false" qchg="false" period="true"/>
    <OptFields seqNum="true" timeStamp="true" dataSet="true" reasonCode="true" dataRef="true"/>
    <RptEnabled max="5">
       <ClientLN iedName="Client1" apRef="S1" ldInst="LD0" lnInst="1" lnClass="IHMI"/>
    </RptEnabled>
</ReportControl>
```

DataSet name 定義を記述

2.5.2 データ構成

MSS オブジェクトの属性を管理するデータ領域は、SCL ファイルに記述された論理デバイス毎に、起動時に動的に確保されています。

ユーザーアプリケーションは、あらかじめ設定したマッピング情報を元に DATA_MAP 構造体へのポインタを抽出して、DATA_MAP 構造体のメンバ *dataPtr をアクセスします。



2.5.3 サンプルプログラム改造の関数一覧

修正対象のソースファイルは、{install dir}/mvl/usr/scl_srvr にあります。

2.5.3.1 scl_srvr.c(修正)

SISCO 社提供のサンプルプログラム(IEC61850対応のサーバー)です。main 関数に初期処理と終了処理を追加します。

関数名	処理名
main	データマップを利用したオブジェクトアクセスの初期化
	IED サンプルの計測処理を開始(スレッド起動)
	IED サンプルの計測処理を終了(スレッド終了)
	データマップでのオブジェクトアクセスで使用したリソースを解放する

2.5.3.2 ied_measure.c (新規作成)

新規作成で、計測用スレッド及び模擬計測値の作成、MMS データへの書込み関連です。

関数名	処理名
ied_measure_total_power	総電力 (Total power) の計測を行う (ダミーの計測データを使用)
ied_do_measure	定期的な計測の処理を行う
ied_measure_thread	ダミーの計測を行うスレッド

2.5.3.3 usermap_obj_access.c (新規作成)

新規作成で、MMS データへのアクセス関連です。

関数名	処理名
usermap_user_text_cmpstr	データマップのユーザ情報での qsort() 用の比較用関数
usermap_dataPtr_cmpptr	データマップの dataPtr の qsort() 用の比較用関数
usermap_init_obj_access	データマップへのアクセス処理の初期化
usermap_cleanup_obj_access	データマップへのアクセス処理のクリーンアップ
usermap_find_by_user_text	datamap.cfg に登録されたユーザデータ情報から、該当するデータマップを検索する
usermap_set_value	ユーザマップ (DATA_MAP) で指定されたデータに値を書き込む
usermap_set_value_without_update_flag	ユーザマップ (DATA_MAP) で指定されたデータに値を書き込む (dupd のためのデータ更新フラグを設定しない)

2.5.3.4 sim_sensor.c (新規作成)

新規作成で、模擬センサー関連です。

関数名	処理名
sim_sensor_create	開閉器模擬の初期化を行う
sim_sensor_destroy	センサー模擬の終了
sim_sensor_get_value	乱数で振れ幅を持たせた計測のダミーの値を得る