```
// FeliCaLibSample07.cpp : コンソール アプリケーションのエントリ ポイントを定義します。
// ポーリングとエリア・サービスキーバージョンの取得
// (Polling -> Request Serviceコマンド)

#include "stdafx.h"
#include <cstdio>
#include <cstdlib>

#include "felica.h"

void print_vector(char* title, unsigned char* vector, int length);

int main(void)
{
    fprintf(stdout, "ポーリングとエリア・サービスキーバージョンの取得¥n¥n");

    /* ライブラリの初期化 */
    if (!initialize_library()) {
        fprintf(stderr, "ライブラリの初期化に失敗しました。¥n¥n");
        return EXIT_FAILURE;
    }

    /* リーダ・ライタの自動認識とオープン */
    if (!open_reader_writer_auto()) {
        fprintf(stderr, "リーダ・ライタのオープンに失敗しました。¥n¥n");
        return EXIT_FAILURE;
    }

    /* ポーリングとエリア・サービスキーバージョンの取得（変数と定数設定） */
    // 第1引数とする構造体、ポーリングをするために必要な情報
    structure_polling polling;
    unsigned char system_code[2] = {0x00, 0x00};
    polling.system_code = system_code;
    polling.time_slot = 0x00;

    // 第3引数とする構造体、ポーリングの時に取得したカードの情報
    structure_card_information card_information;
    unsigned char card_idm[8];    // 製造IDブロックの製造ID(IDm)
    unsigned char card_pmm[8];    // 製造IDブロックの製造パラメータ(PMm)
    card_information.card_idm = card_idm;
    card_information.card_pmm = card_pmm;

    // 第2引数とする構造体、エリア・サービスキーバージョンを取得するために必要な情報
    input_structure_request_service input_request_service;
    input_request_service.number_of_services = 2;
    // サービスコード：1008h(0001 0000 0000 1000b)：ランダム、Read/Write 鍵必要
    // サービスコード：100Ah(0001 0000 0000 1010b)：ランダム、Read Only  鍵必要
    unsigned char service_code_list[4] = {0x08, 0x10, 0x0a, 0x10};
    input_request_service.service_code_list = service_code_list;
```

```
    // 第4引数とする構造体、エリア・サービスキーバージョンの情報
    output_structure_request_service output_request_service;
    unsigned char result_number_of_services = 0;
    unsigned char service_key_version_list[4];
    output_request_service.service_key_version_list = service_key_version_list;
    output_request_service.result_number_of_services = &result_number_of_services;

    /* ポーリングとエリア・サービスキーバージョンの取得(Polling -> Request Serviceコマンド)  */
    if (!polling_and_request_service(&polling, &input_request_service,
                                     &card_information, &output_request_service)) {
        fprintf(stderr, "FeliCaカードが見つかりません。¥n¥n");
        return EXIT_FAILURE;
    }

    /* 取得データの表示 */
    fprintf(stdout, "1008h(Random,Read/Write,鍵必要)と100ah(Random,Read Only,鍵必要)のサービス指定¥n¥n");
    fprintf(stdout, "エリア・サービス数              : %d¥n", result_number_of_services);
    print_vector("エリア・サービスキーバージョン:", service_key_version_list,
                                                  2 * result_number_of_services);

    /* リーダ・ライタのクローズ */
    if (!close_reader_writer()) {
        fprintf(stderr, "リーダ・ライタのクローズに失敗しました。¥n¥n");
        return EXIT_FAILURE;
    }

    /* ライブラリの解放 */
    if (!dispose_library()) {
        fprintf(stderr, "ライブラリの解放に失敗しました。¥n¥n");
        return EXIT_FAILURE;
    }

    fprintf(stdout, "¥nプログラムの実行を終了します。¥n¥n");

    return EXIT_SUCCESS; // 正常終了を示す戻り値
}

void print_vector(char* title, unsigned char* vector, int length)
{
    if (title != NULL) {
        fprintf(stdout, "%s ", title);
    }

    int i;
    for (i = 0; i < length - 1; i++) {
        fprintf(stdout, "%02x ", vector[i]);
    }
    fprintf(stdout, "%02x", vector[length - 1]);
    fprintf(stdout, "¥n");
}
```