```
// FeliCaLibSample05.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。
// リーダ・ライタの自動認識とオープン、リーダ・ライタモードの取得

#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, "リーダ・ライタの自動認識とオープン、リーダ・ライタモードの取得¥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;
    }

    /* リーダ・ライタの活性・不活性判定 */
    bool reader_writer_is_alive_flag;
    if (!reader_writer_is_alive(&reader_writer_is_alive_flag)) {
        fprintf(stderr, "リーダ・ライタの活性・不活性をチェックできませんでした。¥n¥n");
        return EXIT_FAILURE;
    }

    if (!reader_writer_is_alive_flag) {
        fprintf(stderr, "リーダ・ライタは不活性です。¥n¥n");
        return EXIT_FAILURE;
    }

    /* リーダ・ライタのオープン・否オープン判定 */
    bool reader_writer_is_open_flag;
    if (!reader_writer_is_open(&reader_writer_is_open_flag)) {
        fprintf(stderr, "リーダ・ライタのオープン・否オープンの判定できませんでした。¥n¥n");
        return EXIT_FAILURE;
    }

    if (!reader_writer_is_open_flag) {
        fprintf(stderr, "リーダ・ライタは否オープンです。¥n¥n");
        return EXIT_FAILURE;
    }
```

```
/* リーダ・ライタのモード取得 */
structure_reader_writer_mode reader_writer_mode;
char port_name[256];
unsigned char kar[8];    // コントローラとリーダ/ライタ間の相互認証に使用される
unsigned char kbr[8];    // 暗号化処理のための鍵、長さ8バイト
reader_writer_mode.port_name = port_name;
reader_writer_mode.kar = kar;
reader_writer_mode.kbr = kbr;

if (!get_reader_writer_mode(&reader_writer_mode)) {
    fprintf(stderr, "リーダ・ライタのモード取得に失敗しました。¥n¥n");
    return EXIT_FAILURE;
}

/* 取得したデータの表示 */
fprintf(stdout, "port name: %s¥n", port_name);
fprintf(stdout, "baud rate: %d¥n", reader_writer_mode.baud_rate);
switch (reader_writer_mode.encryption_mode) {
case 0x00:
    fprintf(stdout, "encryption mode: Triple DES, CBC ON¥n");
    break;
case 0x01:
    fprintf(stdout, "encryption mode: Disable, CBC OFF¥n");
    break;
case 0x02:
    fprintf(stdout, "encryption mode: Single DES, CBC ON¥n");
    break;
case 0x03:
    fprintf(stdout, "encryption mode: Disable, CBC OFF¥n");
    break;
default:
    fprintf(stdout, "Can't recognize encryption mode.¥n");
    return EXIT_FAILURE;
}
// コントローラとリーダ/ライタ間の相互認証に使用される暗号化処理のための鍵、長さ8バイト
fprintf(stdout, "¥nコントローラとリーダ/ライタ間の相互認証に使用される暗号化処理のための鍵¥n");
print_vector("kar:", kar, sizeof(kar));
print_vector("kbr:", kbr, sizeof(kbr));

/* リーダ・ライタをクローズ */
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");
}
```