



IceWall SSO

Version 10.0

認証 DB 暗号化ライブラリ開発者マニュアル

日本ヒューレット・パッカード株式会社
2010 年 8 月

Printed in Japan

HP Part No. B2873-97016

Rev.100818A

ご注意

本書に記載されました内容は、予告なしに変更することがあります。

本書は内容について細心の注意をもって作成いたしましたが、万一ご不審な点や誤り、記載もれなど、お気づきの点がございましたら当社までお知らせください。

当社は、本書に記載されている誤り、あるいは備品、パフォーマンス、または本書の使用に関連して発生する偶然または必然的な損傷について責任は負いかねます。

本書の著作権は Hewlett-Packard Development Company, L.P. に帰属します。本書の一部あるいは全部についての無断複製、転載を禁じます。

本製品パッケージとして提供した本マニュアル、CD-ROM 等の媒体は本製品用だけにお使いください。

IceWall はヒューレット・パカード社の商標です。

Adobe は Adobe Systems Incorporated の米国及びその他の国における登録商標または商標です。

Java は、米国 Sun Microsystems, Inc. の米国およびその他の国における商標または登録商標です。

Microsoft および Windows は、米国 Microsoft Corporation の、米国、日本およびその他の国における登録商標または商標です。

Oracle は、Oracle Corporation 及びその子会社、関連会社の米国及びその他の国における登録商標です。

— 目次 —

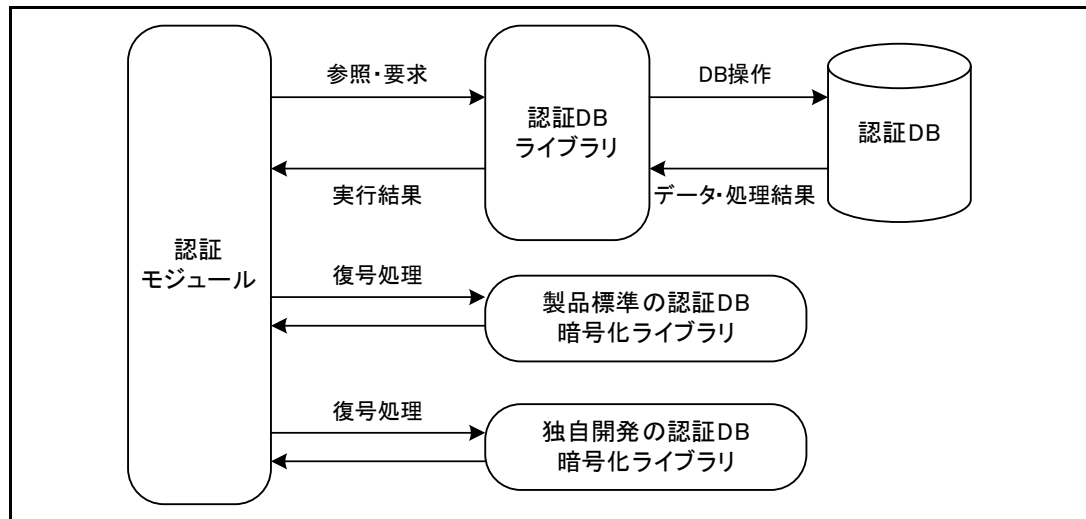
1 はじめに	1
2 認証 DB 暗号化ライブラリとは	2
2.1 製品標準の認証 DB 暗号化ライブラリ	2
2.2 独自開発の認証 DB 暗号化ライブラリ	2
3 開発の手順	3
3.1 仕様の決定	3
3.2 コーディング・ビルド	3
3.3 テスト・デバッグ	3
4 独自開発の認証 DB 暗号化ライブラリ仕様	5
CS_ExDBDecryptColumn	6
5 注意事項	7
5.1 認証 DB 暗号化ライブラリの影響範囲	7
5.2 パフォーマンスへの影響	7
5.3 メモリリーク	7
5.4 サポートについて	7
6 制限事項	8
6.1 OS 共通の制限	8
6.2 HP-UX 11i v3 版 (Itanium)	8
6.3 Linux 版	8
7 開発のヒント	9
7.1 複数の暗号化アルゴリズムの実装	9
8 サンプルソース	10
8.1 接頭辞 ({IWEXIT ~}) により、複数の復号処理に分岐する	10
9 参考	11
9.1 認証 DB 暗号化ライブラリ開発キット	11
9.1.1 スケルトンソース (csexdbcrypto.c)	11
9.1.2 ヘッダーファイル (csexdbcrypto.h)	11
9.1.3 メイクファイル (HP-UX Itanium : Makefile)	12
9.1.4 メイクファイル (Linux 64bit 版 : Makefile)	12

1 はじめに

本書では、認証モジュールに組み込む認証 DB 暗号化ライブラリの開発に必要な情報が記述してあります。

2 認証 DB 暗号化ライブラリとは

認証 DB 暗号化ライブラリは、認証モジュールが認証 DB から取得した暗号化されたデータを復号するためのものです。（データの暗号化は行いません。）



復号対象となる認証 DB のカラムは、認証モジュール設定ファイル (cert.conf) の DBEXATTR 項目で設定された追加カラムのうち、DBCRYPTOATTR 項目で設定したカラムのみとなります。

DBCRYPTOATTR 項目に IceWall SSO 標準のカラムを設定した場合の動作につきましてはサポート対象外となります。

2.1 製品標準の認証 DB 暗号化ライブラリ

製品標準の暗号化アルゴリズムを使用して暗号化されたデータの復号を行います。

認証モジュール設定ファイル(cert.conf)の DBCRYPTOTYPE 項目の値が 1 または 3 で、かつ認証 DB から取得したデータの接頭辞が {IWCRYPTO128} または {IWCRYPTO256} の場合、製品標準の暗号化アルゴリズムを使用していると判断し復号を行います。

2.2 独自開発の認証 DB 暗号化ライブラリ

独自開発された暗号化アルゴリズムを使用して暗号化されたデータの復号を行います。

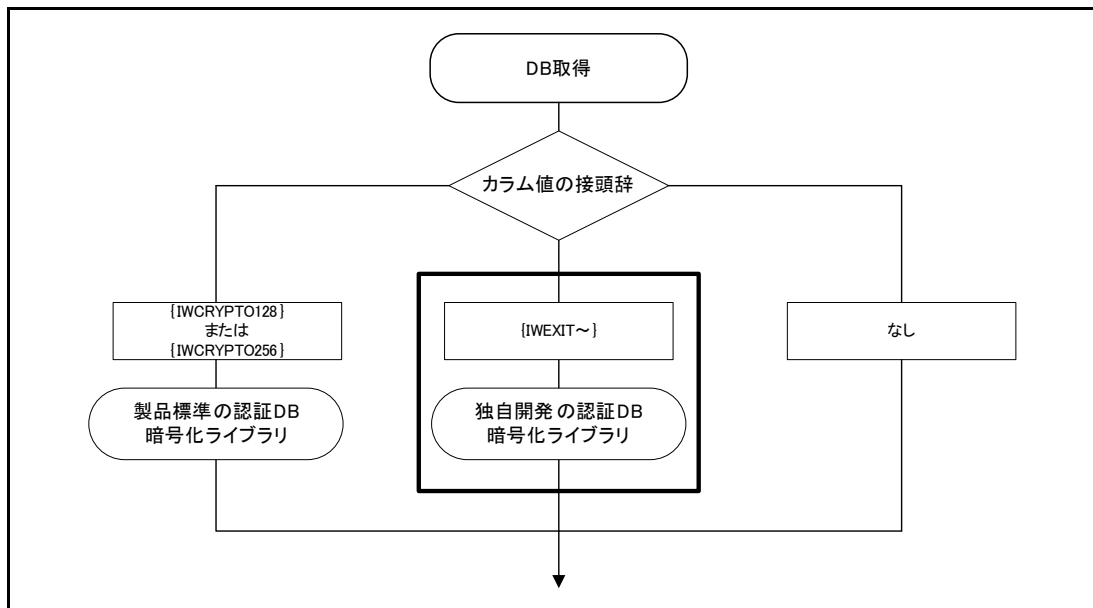
認証モジュール設定ファイル(cert.conf)の DBCRYPTOTYPE 項目の値が 2 または 3 で、かつ認証 DB から取得したデータの接頭辞が {IWEXIT ~} から始まる場合、独自開発の暗号化アルゴリズムを使用していると判断し復号を行います。

3 開発の手順

独自開発の認証 DB 暗号化ライブラリの開発を行うための一般的な手順をご説明いたします。

3.1 仕様の決定

認証 DB 暗号化ライブラリに実装する暗号化アルゴリズムを決定します。



認証 DB 暗号化ライブラリ概要図

(認証 DB 暗号化ライブラリでは復号のみ行いますが、) 暗号化されたデータを認証 DB に格納する場合、データは接頭辞 {IWEXIT ~} から始まる必要があります。{IWEXIT ~} がない場合は、暗号化されていないデータとして扱われます。また、接頭辞 {IWEXIT ~} の長さは” {}” を含め、64 バイトまでとなりますのでご注意ください。

3.2 コーディング・ビルド

決定した仕様に基づき、認証 DB 暗号化ライブラリのソースコードをコーディングします。コーディング後に Shared Library としてビルドします。

コーディングは、開発キットに含まれるスケルトンソースを利用してください。ビルドは開発キットに含まれる Makefile で行います。

```
$ cd /opt/iceawll-ss0/developkit/certd/DBCryptoExit
$ make -f Makefile
```

3.3 テスト・デバッグ

テストプログラムなどにより、作成された Shared Library のテストおよびデバッグを行います。テストが終了した後に、ライブラリを認証モジュールへ組み込み、動作テストを行います。

インストールは以下の手順で行います。

- (1) 認証モジュールが起動している場合は、停止コマンドで停止します。

```
$ /opt/icewall-ss0/certd/bin/end-cert
```

- (2) 標準でインストールされる認証 DB 暗号化ライブラリをバックアップします。

```
$ cd /opt/icewall-ss0/lib  
$ cp -p libiwDBCryptoExit.sl libiwDBCryptoExit.sl.bak
```

- (3) 作成した独自開発の認証 DB 暗号化ライブラリをコピーします。

```
$ cp -p /opt/icewall-ss0/developkit/certd/DBCryptoExit/libiwDBC  
CryptoExit.sl /opt/icewall-ss0/lib
```

- (4) 認証モジュールを起動コマンドで起動します。

```
$ /opt/icewall-ss0/certd/bin/start-cert
```

4 独自開発の認証 DB 暗号化ライブラリ仕様

本章では独自開発の認証 DB 暗号化ライブラリの仕様を解説します。

独自開発の認証 DB 暗号化ライブラリでは以下のものが定義されています。

- ・ 認証モジュールよりコールされるインターフェイス関数
- ・ インターフェイス関数の戻り値は正常と異常のみ

次ページよりこれらの解説を行います。

CS_ExDBDecryptColumn

独自開発の暗号化アルゴリズム用復号インターフェイス関数の仕様は以下のとおりです。

書式 **int CS_ExDBDecryptColumn(char* indata, int inlen, char** outdata, int* outlen)**

機能 接頭辞が {IWEXIT ~} のカラム値を復号する処理を実装します。
引数 indata、inlen から復号前のデータが格納されているアドレスと長さを受け取り、
復号したデータが格納されているアドレスを outdata、長さを outlen に格納します。

引数 **indata** : 接頭辞付の復号対象データを格納したバッファのアドレス

inlen : indata の長さ

outdata : 復号したデータを格納するバッファのアドレス

(outdata の領域確保は認証 DB 暗号化ライブラリ内で行い、開放は復号後のデータ取得後、認証モジュール側にて行います)

outlen : outdata の長さ

戻り値 **EXCS_RESULT_OK** : 正常終了

EXCS_RESULT_NG : 異常終了

EXCS_RESULT_NG を返した場合、フォワーダーはシステムエラーページを返します。

制限事項 なし

5 注意事項

認証 DB 暗号化ライブラリを作成および導入する上で注意しなければならない点をご紹介します。

5.1 認証 DB 暗号化ライブラリの影響範囲

作成されたライブラリのテストを十分に行った上で、認証モジュールへ組み込んでください。また、認証 DB 暗号化ライブラリ内で `exit()` をコールすると、そのモジュールの実行プロセスは終了してしまうため、記述しないでください。

5.2 パフォーマンスへの影響

認証 DB 暗号化ライブラリ内で処理を追加する場合、追加した処理を実行するため、パフォーマンスが低下します。パフォーマンスへの影響を考慮して復号ロジックの決定とコーディングを行ってください。

5.3 メモリリーク

認証 DB 暗号化ライブラリ内で独自に取得したメモリ領域については、必ず解放するようにしてください。解放しない場合はメモリリークの原因となります。

5.4 サポートについて

製品のサポートは、標準で添付される認証 DB 暗号化ライブラリを使用した場合にのみ受けることができます。お客さまが作成された認証 DB 暗号化ライブラリが導入されている状態ではサポートを受けることができません。標準添付のライブラリはサポート時に必要となりますので、必ずバックアップを行ってください。

6 制限事項

認証 DB 暗号化ライブラリを作成する場合には、以下の制限事項があることに注意してください。

以下の制限内で作成されない場合には、各プロセス自体が実行できなくなる、動作が不安定になる原因になります。

6.1 OS 共通の制限

64bit バイナリとしてビルドしてください。

認証モジュールは内部処理でスレッドを使用しており、認証 DB 暗号化ライブラリはスレッドよりコールされるため、標準ライブラリ関数は必ずスレッドセーフ版を使用するようにしてください。また、ユーザー関数を作成する場合は、スレッドセーフとして動作する必要があります。

6.2 HP-UX 11i v3 版 (Itanium)

コンパイルオプションには必ず「+DD64」を付加してください。

認証 DB 暗号化ライブラリを組み込む認証モジュールが動作するプロセッサと同一アーキテクチャのものでビルドしてください。また、リンクするライブラリは 64bit ライブラリのみとしてください。

6.3 Linux 版

コンパイルオプションには必ず「-m64」を付加してください。

認証モジュールのスレッドは「Linux Threads」及び「Native POSIX Library Thread」が使用できます。

7 開発のヒント

認証 DB 暗号化ライブラリの開発を行うにあたり、そのヒントとなる方法をご紹介します。

7.1 複数の暗号化アルゴリズムの実装

複数の暗号化アルゴリズムを実装する場合、独自開発の暗号化ライブラリの接頭辞 {IWEXIT ~} をそれぞれに定義することにより、インターフェイス関数内で判断することが出来ます。

8 サンプルソース

以下に認証 DB 暗号化ライブラリのサンプルソースをご紹介します。

8.1 接頭辞（{IWEXIT ~}）により、複数の復号処理に分岐する

```
#include <stdlib.h>
#include <string.h>

#include "csexdbcrypto.h"

extern int myDBDecrypt001(char *indata, int inlen, char **outdata, int *outlen); /* ※ 1 */
extern int myDBDecrypt002(char *indata, int inlen, char **outdata, int *outlen); /* ※ 1 */
extern int myDBDecrypt003(char *indata, int inlen, char **outdata, int *outlen); /* ※ 1 */

int CS_ExDBDecryptColumn(char *indata, int inlen, char **outdata, int *outlen)
{
    if (indata == NULL || inlen < 1 || outdata == NULL || outlen == NULL) {
        return EXCS_RESULT_NG;
    }

    /* 接頭辞の {IWEXITxxxx} により、それぞれに適した復号処理を行う */
    if (strncasecmp("{IWEXITCRYPTO001}", indata, sizeof("{IWEXITCRYPTO001}") - 1) == 0) {
        /* 復号方式 IWEXITCRYPTO001 の処理 */
        myDBDecrypt001(indata, inlen, outdata, outlen);
    } else if (strncasecmp("{IWEXITCRYPTO002}", indata, sizeof("{IWEXITCRYPTO002}") - 1) == 0) {
        /* 復号方式 IWEXITCRYPTO002 の処理 */
        myDBDecrypt002(indata, inlen, outdata, outlen);
    } else if (strncasecmp("{IWEXITCRYPTO003}", indata, sizeof("{IWEXITCRYPTO003}") - 1) == 0) {
        /* 復号方式 IWEXITCRYPTO003 の処理 */
        myDBDecrypt003(indata, inlen, outdata, outlen);
    } else {
        /* 対応していない暗号化方式。受け取ったデータを変換せずに、認証モジュールに渡す */
        *outdata = (char*)malloc(inlen+1);
        if (*outdata == NULL) {
            return EXCS_RESULT_NG;
        }

        memcpy(*outdata, indata, inlen);
        *(*outdata+inlen) = '\0';
        *outlen = inlen;
    }

    return EXCS_RESULT_OK;
}
```

※ 1 myDBDecrypt001() 関数、myDBDecrypt002() 関数、myDBDecrypt003() 関数は、API として存在するものではありません。暗号化されたカラム値を復号する処理を想定した仮想の関数です。

9 参考

標準でインストールされる認証 DB 暗号化ライブラリ開発キットの内容です。
開発キットの構成は以下のディレクトリおよびファイルとなっています。

ディレクトリおよびファイル					内容
/opt/icewall-ss0	/developkit	/certd	/DBCryptoExit	/csexdbcrypto.c	認証 DB 暗号化ライブラリ 開発キット
				/csexdbcrypto.h	
				/Makefile	

なお、この開発キットで作成されるライブラリ名は標準でインストールされるライブラリ名と同じものになりますので、作成および導入するには必ずバックアップを行ってください。

9.1 認証 DB 暗号化ライブラリ開発キット

9.1.1 スケルトンソース (csexdbcrypto.c)

以下のソースコードでは、認証 DB から取得した値を、変換せずにそのまま認証モジュールに返します。

```
#include <stdlib.h>
#include <string.h>

#include "csexdbcrypto.h"

int CS_ExDBDecryptColumn(char *indata, int inlen, char **outdata, int *outlen)
{
    if (indata == NULL || inlen < 1 || outdata == NULL || outlen == NULL) {
        return EXCS_RESULT_NG;
    }

    *outdata = (char*)malloc(inlen+1);
    if (*outdata == NULL) {
        return EXCS_RESULT_NG;
    }

    memcpy(*outdata, indata, inlen);
    *(*outdata+inlen) = '\0';
    *outlen = inlen;

    return EXCS_RESULT_OK;
}
```

9.1.2 ヘッダーファイル (csexdbcrypto.h)

```
#ifndef CSEX_DBCRYPTO_H
#define CSEX_DBCRYPTO_H
```

```
#define EXCS_RESULT_OK      0
#define EXCS_RESULT_NG     -1

#endif /* #ifndef CSEX_DBCRYPTO_H */
```

9.1.3 メイクファイル (HP-UX Itanium : Makefile)

```
CC      = cc

CCFLAGS  = -D_UNIX -D_HPUX_SOURCE -D_POSIX_C_SOURCE=199506L -Aa +e
          -D_FILE_OFFSET_BITS=64 -z +DD64 +z

LIBS      =

INCLUDE   = -I./

MAKEFILE  = Makefile

OBJS      = csexdbcrypto.o

PROGRAM   = iwDBCryptoExit

SRCS      = csexdbcrypto.c

all:  $(PROGRAM)

$(PROGRAM):$(OBJS)
        ld -b -z -o lib$(PROGRAM).sl $(OBJS) $(LIBS)
.c.o:
        $(CC) $(CCFLAGS) $(INCLUDE) -c $(SRCS)

clean:
rm -f $(OBJS) lib$(PROGRAM).sl core
```

9.1.4 メイクファイル (Linux 64bit 版 : Makefile)

```
CC      = gcc

CCFLAGS  = -m64 -fPIC -DLinux -D_FILE_OFFSET_BITS=64 -
          D_LARGEFILE_SOURCE -D_GNU_SOURCE

LDFLAGS  = -m64 -fPIC

LIBS      =

INCLUDE   = -I./

MAKEFILE  = Makefile

OBJS      = csexdbcrypto.o
```

```
PROGRAM    = iwDBCryptoExit

SRCS       = csexdbcrypto.c

all:  $(PROGRAM)

$(PROGRAM):$(OBJS)
        gcc -shared $(LDFLAGS) -o lib$(PROGRAM).sl $(OBJS) $(LIBS)
.c.o:
        $(CC) $(CCFLAGS) $(INCLUDE) -c $(SRCS)
clean:
        rm -f $(OBJS) lib$(PROGRAM).sl core
```