



IceWall SSO

Version 10.0

IceWall Cert Protocol 2.0 開発者マニュアル

日本ヒューレット・パッカート株式会社
2010 年 8 月

Printed in Japan

HP Part No. B2873-97011

Rev.100819A

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・パッカー社社の商標です。

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
1.1 文中におけるバージョン表示記号について	1
2 IceWall Cert Protocol 2.0 について	2
2.1 IceWall Cert Protocol とは	2
2.2 メッセージ構成	2
2.3 制限事項	2
3 IceWall Cert Protocol 2.0 詳細	4
3.1 メッセージ一覧	4
3.2 メッセージフォーマット	5
3.2.1 リクエストメッセージヘッダー部	5
3.2.2 レスポンスメッセージヘッダー部	6
3.2.3 ボディ部	7
ユーザー ID とパスワードによるログインリクエスト	10
ユーザー ID とパスワードによる強制ログインリクエスト	12
クライアント証明書とパスワードによるログインリクエスト	14
クライアント証明書とパスワードによる強制ログインリクエスト	17
ユーザー ID とパスワードによるログインレスポンス	19
クライアント証明書とパスワードによるログインレスポンス	22
アクセス制御リクエスト (ユーザー ID とパスワード)	25
アクセス制御リクエスト (クライアント証明書とパスワード)	27
アクセス制御レスポンス (ユーザー ID とパスワード)	30
アクセス制御レスポンス (クライアント証明書とパスワード)	32
パスワード変更リクエスト	34
パスワード変更レスポンス	36
ログアウトリクエスト	38
ログアウトレスポンス	40
4 通信メッセージ暗号化ライブラリについて (10.0)	42
4.1 通信メッセージ暗号化ライブラリとは	42
4.2 通信メッセージ暗号化ライブラリの拡張	42
4.2.1 製品標準暗号用の通信メッセージ暗号化ライブラリ	43
4.2.2 独自暗号用の通信メッセージ暗号化ライブラリ	43
4.3 通信メッセージ暗号化ライブラリの開発について	44
4.3.1 製品標準暗号用の通信メッセージ暗号化ライブラリの実装手順	44
4.3.2 独自暗号用の通信メッセージ暗号化ライブラリの実装手順	46
4.4 独自暗号用の通信メッセージ暗号化ライブラリを使用する場合の注意点	47
5 製品標準暗号用の通信メッセージ暗号化ライブラリ仕様	48
5.1 コールバック関数	48
IW_ExEncrypto	49
IW_ExDecrypto	50
5.2 API 関数	51
IW_ExCPTCreateCrypto	52
IW_ExCPTReleaseCrypto	53

6 独自暗号用の通信メッセージ暗号化ライブラリ仕様 10.0	54
6.1 コールバック関数	54
CS_ExEncrypto	55
CS_ExDecrypto	56
6.2 API 関数	57
IW_ExCPTCreateCrypto	58
IW_ExCPTReleaseCrypto	59
CS_ExGetIWEncryptedType	60
7 ICP を実装する際の注意および制限事項	61
7.1 注意事項	61
7.2 制限事項	61
8 通信メッセージ暗号化ライブラリを開発する際の注意および制限事項	62
8.1 注意事項	62
8.2 制限事項	62
9 ICP2.0 クライアントサンプル	64
9.1 クライアントサンプル 1	64
9.2 クライアントサンプル 2	65
9.3 エージェントサンプル (Apache HTTP Server 1.3.x 用)	66
9.4 サンプルソース (C 言語)	68
9.4.1 クライアントサンプル 1 (sample1.c)	68
9.4.2 クライアントサンプル 2 (sample2.c)	71
9.5 サンプルソース (Java 言語)	73
9.5.1 クライアントサンプル 1 (Sample1.java)	73
9.5.2 クライアントサンプル 2 (Sample2.java)	78
9.6 サンプルソース (エージェントサンプル : mod_sample.c)	83
10 通信メッセージ暗号化ライブラリサンプル	95
10.1 サンプルソースファイル (iwcrypto.c)	95
10.2 ヘッダーファイル (iwcrypto.h)	96
10.3 フォワーダー用メイクファイル (HP-UX 版 : Makefile) 10.0	97
10.4 フォワーダー用メイクファイル (Linux 版 : Makefile) 10.0	97
10.5 認証モジュール用メイクファイル (HP-UX 版 : Makefile)	98
10.6 認証モジュール用メイクファイル (Linux 版 : Makefile)	99

1 はじめに

本書は IceWall SSO Version 8.0 以降のバージョンに搭載されているモジュール間通信メッセージ (IceWall Cert Protocol 2.0) のメッセージフォーマット、メッセージ仕様およびモジュール間通信メッセージの暗号化／復号を行うライブラリを開発するために必要な情報を記載しています。

記述されている仕様を実装することで、フォワーダーやエージェントと同様に認証モジュールと通信を行い、認証・認可を行うカスタムモジュールを作成することができるようになります。

なお、本書は大きく分けて次の 2 つの内容が記載されています。

- (1) IceWall Cert Protocol 2.0 メッセージ仕様
- (2) 認証モジュール用通信メッセージ暗号化／復号ライブラリ開発用リファレンス

これらの内容を正しく理解し、IceWall SSO が持つ強力な認証および認可の機能をご利用ください。

1.1 文中におけるバージョン表示記号について

文中に付加されているバージョン表示記号は、以下の意味を表します。

表示記号	意味
10.0	四角で囲まれているバージョンで追加された項目です。この場合、10.0 にて追加されたことを表します。
10.0	楕円で囲まれているバージョンで仕様変更もしくは機能追加された項目です。この場合、10.0 にて仕様変更または機能追加されたことを表します。

2 IceWall Cert Protocol 2.0 について

本章より IceWall Cert Protocol（以下、ICP）2.0 の仕様について記載いたします。

2.1 IceWall Cert Protocol とは

IceWall Cert Protocol とは、IceWall SSO を構成するモジュール間で行われるメッセージ通信のプロトコルです。

2.2 メッセージ構成

ICP 2.0 は大きく分けて次の 2 種類のメッセージで構成されます。

- (1) リクエストメッセージ
- (2) レスポンスメッセージ

リクエストメッセージは、クライアントとなるモジュールから認証モジュールに対して送信する要求メッセージです。このメッセージを認証モジュールが受信しますと、それに応じた処理を行います。

レスポンスメッセージは、クライアントモジュールからのリクエストに対する処理結果を、認証モジュールからリクエスト元へ送信するメッセージです。このメッセージにはリクエストの結果が格納されています。

1 回のシーケンスでは、1 リクエストに対して 1 レスポンスです。1 リクエストに対して複数のレスポンスが送信されることはありません。

2.3 制限事項

ICP 2.0 には以下の制限事項が存在します。

- ・ メッセージはテキストデータとして扱うため、バイナリーデータを扱うことはできません。
- ・ メッセージ自体は暗号化されて通信することが前提となっています。暗号化および復号のアルゴリズムは認証モジュールで使用する「通信メッセージ暗号化ライブラリ」と同じものでなければなりません。
- ・ メッセージ自体の暗号化、復号は製品標準の通信メッセージ暗号化ライブラリ、または独自暗号用の通信メッセージ暗号化ライブラリを使用することが可能です。
- ・ リクエストメッセージヘッダー部のデータ長(Content-length)がボディ部のデータ長と異なる場合は、認証モジュールよりレスポンスとしてシステムエラーが返されます。このとき、レスポンスのボディ部は標準暗号用の通信メッセージ暗号化ライブラリ

リによって暗号化された内容になります。また、認証モジュールのエラーログには次のメッセージが出力されます。 (10.0)

Fatal: Content-length Error. [Content-length に設定した長さ] Body=[実ボディ長] [EL60606-05013]

なお、8.0 R3 以降では次のメッセージも追加で出力されます。

Fatal: EC16408-25115 Not Found Content-Length.
Fatal: EC16407-25114 Content-Length is 0.
Fatal: EC16409-05013 Content-length Error. [%d] Body=[%d]

- 文字コードは Shift_JIS のみサポートされます。

3 IceWall Cert Protocol 2.0 詳細

本章では ICP 2.0 の詳細な内容を記載します。

3.1 メッセージ一覧

ICP 2.0 で実装されているメッセージの一覧です。

メッセージ名	内容
ReqLoginUID	ユーザー ID とパスワードによるログインを行うリクエストです。
ReqForceLoginUID	ユーザー ID とパスワードによる強制ログインを行うリクエストです。
ResLoginUID	ユーザー ID とパスワードによるログインまたは強制ログインリクエストに対するレスポンスです。
ReqLoginCERT	クライアント証明書とパスワードによるログインを行うリクエストです。
ReqForceLoginCERT	クライアント証明書とパスワードによる強制ログインを行うリクエストです。
ResLoginCERT	クライアント証明書とパスワードによるログインおよび強制ログインリクエストに対するレスポンスです。
ReqAccessUID	ユーザー ID とパスワードによるログイン後、アクセス制御を行うリクエストです。
ReqAccessCERT	クライアント証明書とパスワードによるログイン後、アクセス制御を行うリクエストです。
ResAccessUID	ユーザー ID とパスワードによるログイン後のアクセス制御リクエストに対するレスポンスです。
ResAccessCERT	クライアント証明書とパスワードによるログイン後のアクセス制御リクエストに対するレスポンスです。
ReqPasswdChg	パスワード変更を行うリクエストです。
ResPasswdChg	パスワード変更リクエストに対するレスポンスです。
ReqLogout	ログアウトを行うリクエストです。
ResLogout	ログアウトリクエストに対するレスポンスです。

3.2 メッセージフォーマット

メッセージはリクエストメッセージ、レスポンスメッセージともにヘッダー部とボディ部の2つで構成されます。

3.2.1 リクエストメッセージ ヘッダー部

リクエストメッセージのヘッダー部は次のデータで構成されます。

REQ / ICP/2.0	CRLF	リクエスト行
X-icw-encrypted: [enc-type]	CRLF	暗号化方式行
Content-length: [length]	CRLF	データ長行
. . . .		任意のヘッダー行
CRLF		ヘッダー区切り

名称	必須	内容
リクエスト行	○	リクエストを表すキーワード「REQ」、固定文字列「/」およびプロトコルバージョンを設定します。それぞれの要素は空白文字で接続されます。 プロトコルバージョンには、プロトコル名とバージョン番号を「/」で区切り設定します。 ICP 2.0 の場合は「REQ / ICP/2.0」固定となります。 リクエスト行が正しくない場合は、ICP 1.0 リクエストの不正なリクエストとして扱われます。
暗号化方式行 10.0	×	暗号化されたボディ部の通信メッセージ暗号化ライブラリを設定します。 設定方法は次のとおりです。 ・標準暗号の libCryptoExit.sl で暗号化／復号を行う。 X-icw-encrypted: icewall を設定する。 または、 X-icw-encrypted ヘッダーを設定しない。 ・独自暗号の libExCryptoExit.sl で暗号化／復号を行う。 X-icw-encrypted ヘッダーに icewall 以外の独自暗号を示す値を設定する。
データ長行	○	ボディ部のデータ長を設定します。 認証モジュールでは、ここで設定されたデータ長でボディ部を取得します。 データ長がボディ部の長さと異なる場合は、認証モジュールよりレスポンスは返されず接続を切断されます。

名称	必須	内容
任意のヘッダー行 (10.0)	×	任意のヘッダーを設定します。 フォーマットは「ヘッダー名: ヘッダー値」です。 (例) Client-Id: 0001 ヘッダー行の区切りとして行末に改行コードが必要です。 追加可能なヘッダー行の上限はありません。 次のヘッダー名は予約語のため任意ヘッダーとして使用することはできません。 Content-length X-iw-encrypted X-iw-fallback
ヘッダー区切り	○	ヘッダー区切りとして改行コードのみを設定します。 このヘッダー区切りがなければ、ボディ部もヘッダー部と認識され、不正メッセージとして扱われます。

3.2.2 レスポンスメッセージ ヘッダー部

レスポンスメッセージのヘッダー部は次のデータで構成されます。

ICP/2.0 [status] [msg]	CRLF	レスポンス行
X-iw-encrypted: [enc-type]	CRLF	暗号化方式行
Content-length: [length]	CRLF	データ長行
...		任意のヘッダー行
CRLF		ヘッダー区切り

名称	内容
レスポンス行	プロトコルバージョン、リクエストに対する処理結果のステータスコードと文字列が設定されます。それぞれの要素は空白文字で接続されます。 プロトコルバージョンには、プロトコル名とバージョン番号を「/」で区切り設定し、「ICP/2.0」が返されます。 送信されるステータスコードと文字列は次のとおりです。 200 OK: 処理に成功した場合に返されます。 500 NG: 処理に失敗、認証モジュール側で何らかのエラーが発生した場合に返されます。 処理に失敗した場合、失敗した理由についてはボディ部の情報を参照してください。(レスポンス行では大まかな処理の正否のみです)

名称	内容
暗号化方式行 (10.0)	暗号化されたボディ部の通信メッセージ暗号化ライブラリが設定されます。 通信メッセージ暗号化ライブラリが標準暗号か独自暗号かの判別は次の通りです。 - 標準暗号の libCryptoExit.sl で暗号化／復号を行う。 X-iw-encrypted: icewall を設定される。 - または、 X-iw-encrypted ヘッダーが存在しない。 - 独自暗号の libExCryptoExit.sl で暗号化／復号を行う。 X-iw-encrypted ヘッダーに icewall 以外の独自暗号を示す値が設定される。
データ長行	ボディ部のデータ長が設定されます。
任意のヘッダー行 (10.0)	リクエスト時に送信した任意のヘッダー行と、認証モジュール側で追加されたヘッダー (UserExit ルーチン使用時のみ) が設定されます。 フォーマットは「ヘッダー名: ヘッダー値」です。 (例) Client-Id: 0001 ヘッダー行の区切りとして行末に改行コード付加されます。 送信される行数は不定です (最小は 0 行です)。 次のヘッダー名は予約語のため任意ヘッダーとして使用することはできません。 - Content-length 「X-iw-」から始まるヘッダー
ヘッダー区切り	ヘッダー区切りとして改行コードが設定されます。

3.2.3 ボディ部

ボディ部は実際のリクエストおよびレスポンスのデータで構成されます。通信時にはこの部分が暗号化されます。暗号化されたデータの長さがヘッダー部のデータ長となります。

データ 1	CRLF
データ 2	CRLF
...	
データ n	CRLF

ボディ部では必要なデータを改行コードにより区切り、複数個連結します。
1 つのデータは次の要素で構成されます。

名称	:	空白	値
----	---	----	---

- 各データの順序は意識しません。
- 各データの名称は任意に設定可能ですが、次の名称は予約語となり特定の目的以外に使用できません。

AGENT_ID CERT_EXPR CERT_NO ERR_NO IW_INFO IW_PWD IW_UID MSG_ID NEW_PWD PWD_EXPR REMOTE_ADDR SOURCE_ADDR SID_EXPR ACC_URL CERT_UID METHOD USER_AGENT X-iw-encrypted

- 要素数は最大 512 個までです。
- 各要素の名称と値は最大 1024 バイトとなります。
- 名称と値のセパレーターはコロン+空白文字ですが、空白文字は複数個指定しても 1 つのセパレーターとして認識されます。
- データとして 2 バイト文字を扱う場合には「%xx」のように 16 進数変換を行ってください。

例) “日” → %93%FA

ボディ部では各メッセージに必要な要素以外のデータを設定することが可能で、次の 3 種類が存在します。

- (1) 環境情報
- (2) 付加情報
- (3) 補足情報

(1) 環境情報

ログインに関連するリクエストメッセージに追加された情報はリクエスト元の環境情報として認証モジュールに保存され、ユーザーのグループ設定にも使用することが可能です。この情報は、アクセス制御レスポンスでユーザー情報の一部としてリクエスト元に返信されます。なお、この情報はログアウトするまで有効です。

(2) 付加情報

ログイン関連以外のリクエストメッセージに追加された情報はリクエスト時に追加で送られる情報として扱われます。この情報は認証モジュールで保存されることなく、レスポンスの返信とともに消去されます。

(3) 補足情報

すべてのレスポンスメッセージの必要要素以外で追加される情報は補足情報となります。通常、補足情報は付加されることはありませんが、認証モジュールの UserExit ルーチンにてレスポンスメッセージに追加された場合のみ、リクエスト元に返信されます。

次ページより各リクエストおよびレスポンスのフォーマットについて解説します。

ユーザー ID とパスワードによるログインリクエスト

概要 ユーザー ID とパスワードのみで IceWall へ認証を行う場合に使用するログインリクエストです。通常の認証ではこのリクエストを使用します。

フォーマット

MSG_ID: ReqLoginUID_{CRLF}
IW_UID: ユーザー ID_{CRLF}
IW_PWD: パスワード_{CRLF}
REMOTE_ADDR: IP アドレス_{CRLF}
AGENT_ID: リクエスト送信元識別名_{CRLF}
USER_AGENT: ユーザーエージェント名_{CRLF}
任意の要素名: 任意のデータ_{CRLF}

解説

本リクエストを構成する要素は次のとおりです。

MSG_ID
ReqLoginUID を設定します。この要素は必須です。

IW_UID
ログインを行うユーザーのユーザー ID を設定します。ユーザー ID 長は 64 バイトまでです。この要素は必須です。

IW_PWD
ログインを行うユーザーのパスワードを設定します。パスワード長は 128 バイトまでです。この要素は必須です。

REMOTE_ADDR
クライアントの IP アドレスを設定します。IPv4 アドレスの場合はドット区切り表記、IPv6 アドレスの場合は 8 つの 16 ビットフィールドで構成される 128 ビットで、16 進数値で設定します。この要素は、ログイン時のグループ制御やリクエスト制御で使用されるため、設定することを推奨いたします。REMOTE_ADDR は環境情報として格納されます。

AGENT_ID
リクエスト送信元を特定する識別名を設定します。識別名長の制限はありません。この要素は、認証モジュールのアクセスログに出力されるため（ただし、認証モジュール側の設定による）、設定することを推奨いたします。

USER_AGENT
クライアントで使用されるブラウザの User-Agent を設定します。この要素は将来の拡張において使用される場合があるため、設定することを推奨いたします。

任意の要素名
リクエスト送信元の環境情報として任意の情報を送信することが可能です。名称も任意に設定可能です。ここで送信された情報はバックエンド Web サーバーへ通知する情報継承の対象となります。

- 使用例
- (1) ユーザーIDがuser01、パスワードがpasswordで環境情報「REQUEST_TIME」の値「09:12:07」を追加してログインを行う場合
- MSG_ID: ReqLoginUID
IW_UID: user01
IW_PWD: password
REMOTE_ADDR: 192.168.0.1
AGENT_ID: IceWall SSO DFW
USER_AGENT: Mozilla/x.x
REQUEST_TIME: 09:12:07
- (2) 最低限の情報のみでログインを行う場合
- MSG_ID: ReqLoginUID
IW_UID: user01
IW_PWD: password

※ データ区切りの改行コード CRLF は省略しています。

- 備考
- 本リクエストで REMOTE_ADDR を送信しない場合、認証モジュールにて IP アドレスを使用したグルーピング、リクエスト制御は行えなくなります。
 - 本リクエストで AGENT_ID を送信しない場合、認証モジュールのアクセスログに AGENT_ID が出力されなくなります。

ユーザー ID とパスワードによる強制ログインリクエスト

概要 認証モジュール側で「二重ログイン禁止／排他ログイン」設定の場合に、同一ユーザー ID にてログイン済みのユーザーを強制ログアウトした上で、ログインを行うリクエストです。通常はログインレスポンスで「二重ログインエラー」となり、強制的にログインを行う場合に使用します。

フォーマット

MSG_ID: ReqForceLoginUID_{CRLF}
IW_UID: ユーザー ID_{CRLF}
IW_PWD: パスワード_{CRLF}
REMOTE_ADDR: IP アドレス_{CRLF}
AGENT_ID: リクエスト送信元識別名_{CRLF}
USER_AGENT: ユーザーエージェント名_{CRLF}
任意の要素名: 任意のデータ_{CRLF}

解説 本リクエストを構成する要素は次のとおりです。

MSG_ID
ReqForceLoginUID を設定します。この要素は必須です。

IW_UID
ログインを行うユーザーのユーザー ID を設定します。ユーザー ID 長は 64 バイトまでです。この要素は必須です。

IW_PWD
ログインを行うユーザーのパスワードを設定します。パスワード長は 128 バイトまでです。この要素は必須です。

REMOTE_ADDR
クライアントの IP アドレスを設定します。IPv4 アドレスの場合はドット区切り表記、IPv6 アドレスの場合は 8 つの 16 ビットフィールドで構成される 128 ビットで、16 進数値で設定します。この要素は、ログイン時のグループ制御やリクエスト制御で利用されるため、設定することを推奨いたします。REMOTE_ADDR は環境情報として格納されます。

AGENT_ID
リクエスト送信元を特定する識別名を設定します。識別名長の制限はありません。この要素は、認証モジュールのアクセスログに出力されるため（ただし、認証モジュール側の設定による）、設定することを推奨いたします。

USER_AGENT
クライアントで利用されるブラウザの User-Agent を設定します。この要素は将来の拡張において利用される場合があるため、設定することを推奨いたします。

任意の要素名
リクエスト送信元の環境情報として任意の情報を送信することが可能で

す。名称も任意に設定可能です。ここで送信された情報はバックエンド Web サーバーへ通知する情報継承の対象となります。

使用例

- (1) ユーザーIDがuser01、パスワードがpasswordで環境情報「REQUEST_TIME」の値を「09:12:07」を追加して強制ログインを行う場合
MSG_ID: ReqForceLoginUID
IW_UID: user01
IW_PWD: password
REMOTE_ADDR: 192.168.0.1
AGENT_ID: IceWall SSO DFW
USER_AGENT: Mozilla/x.x
REQUEST_TIME: 09:12:07
- (2) 最低限の情報のみでログインを行う場合
MSG_ID: ReqForceLoginUID
IW_UID: user01
IW_PWD: password

※ データ区切りの改行コード CRLF は省略しています。

備考

- 本リクエストで REMOTE_ADDR を送信しない場合、認証モジュールにて IP アドレスを使用したグルーピング、リクエスト制御は行えなくなります。
- 本リクエストで AGENT_ID を送信しない場合、認証モジュールのアクセスログに AGENT_ID が出力されなくなります。

クライアント証明書とパスワードによるログインリクエスト

概要	クライアント証明書を使用してIceWallへ認証を行う場合に使用するログインリクエストです。ユーザー ID とパスワードによるログインよりセキュアな認証を行う場合に、このリクエストを使用します。
フォーマット	MSG_ID: ReqLoginCERT _{CRLF} IW_UID: ユーザー ID _{CRLF} IW_PWD: パスワード _{CRLF} CERT_NO: シリアル No _{CRLF} CERT_EXPR: 有効期限 _{CRLF} REMOTE_ADDR: IP アドレス _{CRLF} AGENT_ID: リクエスト送信元識別名 _{CRLF} USER_AGENT: ユーザーエージェント名 _{CRLF} 任意の要素名: 任意のデータ _{CRLF}
解説	本リクエストを構成する要素は次のとおりです。 MSG_ID ReqLoginCERT を設定します。この要素は必須です。 IW_UID ログインを行うユーザーのクライアント証明書に含まれるユーザーID を設定します。ユーザー ID 長は 64 バイトまでです。この要素は必須です。 IW_PWD ログインを行うユーザーのパスワードを設定します。パスワード長は 128 バイトまでです。この要素は必須です。 CERT_NO クライアント証明書のシリアル No を設定します。証明書の一意性を高めるため、「シリアル No:Issuer」を設定することも可能です。データ長の制限はありません。この要素は必須です。 CERT_EXPR クライアント証明書の有効期限を以下の形式で設定します。 YYYYMMDDhhmmss 有効期限長は 14 バイトまでです。この要素は必須です。 REMOTE_ADDR クライアントの IP アドレスを設定します。IPv4 アドレスの場合はドット区切り表記、IPv6 アドレスの場合は 8 つの 16 ビットフィールドで構成される 128 ビットで、16 進数値で設定します。この要素は、ログイン時のグループ制御やリクエスト制御で利用されるため、設定することを推奨いたします。REMOTE_ADDR は環境情報として格納されます。 AGENT_ID

リクエスト送信元を特定する識別名を設定します。識別名長の制限はありません。この要素は、認証モジュールのアクセスログに出力されるため（ただし、認証モジュール側の設定による）、設定することを推奨いたします。

USER_AGENT

クライアントで使用されるブラウザの User-Agent を設定します。この要素は将来の拡張において使用される場合があるため、設定することを推奨いたします。

任意の要素名

リクエスト送信元の環境情報として任意の情報を送信することが可能です。名称も任意に設定可能です。ここで送信された情報はバックエンド Web サーバーへ通知する情報継承の対象となります。

使用例

- (1) ユーザー ID が user01、シリアル No が 01、有効期限が 2009 年 9 月 30 日のクライアント証明書を使用してログインを行う場合。

MSG_ID: ReqLoginCERT
IW_UID: user01
IW_PWD: password
CERT_NO: 01
CERT_EXPR: 20090930235959
REMOTE_ADDR: 192.168.0.1
AGENT_ID: IceWall SSO DFW
USER_AGENT: Mozilla/x.x

- (2) クライアント証明書の Issuer をシリアル No に付加してログインを行う場合

MSG_ID: ReqLoginCERT
IW_UID: user01
IW_PWD: password
CERT_NO: 01:OU=Development/O=hp.com
CERT_EXPR: 20060930235959
REMOTE_ADDR: 192.168.0.1
AGENT_ID: IceWall SSO DFW
USER_AGENT: Mozilla/x.x

- (3) 必要最低限の情報のみでログインを行う場合

MSG_ID: ReqLoginCERT
IW_UID: user01
IW_PWD: password
CERT_NO: 01
CERT_EXPR: 20060930235959

※ データ区切りの改行コード CRLF は省略しています。

備考

- 本リクエストで REMOTE_ADDR を送信しない場合、認証モジュールにて IP アドレスを使用したグルーピング、リクエスト制御は行えなくなります。
- 本リクエストで AGENT_ID を送信しない場合、認証モジュールのアクセスログに AGENT_ID が出力されなくなります。

クライアント証明書とパスワードによる強制ログインリクエスト

概要 認証モジュール側で「二重ログイン禁止／排他ログイン」設定の場合に、同一ユーザー ID にてログイン済みのユーザーを強制ログアウトした上で、ログインを行うリクエストです。通常はログインレスポンスで「二重ログインエラー」となり、強制的にログインを行う場合に使用します。

フォーマット

MSG_ID: ReqForceLoginCERT_{CRLF}
IW_UID: ユーザー ID_{CRLF}
IW_PWD: パスワード_{CRLF}
CERT_NO: シリアル No_{CRLF}
CERT_EXPR: 有効期限_{CRLF}
REMOTE_ADDR: IP アドレス_{CRLF}
AGENT_ID: リクエスト送信元識別名_{CRLF}
USER_AGENT: ユーザーエージェント名_{CRLF}
任意の要素名: 任意のデータ_{CRLF}

解説 本リクエストを構成する要素は次のとおりです。

MSG_ID
 ReqForceLoginCERT を設定します。この要素は必須です。

IW_UID
 ログインを行うユーザーのクライアント証明書に含まれるユーザーID を設定します。ユーザー ID 長は 64 バイトまでです。この要素は必須です。

IW_PWD
 ログインを行うユーザーのパスワードを設定します。パスワード長は 128 バイトまでです。この要素は必須です。

CERT_NO
 クライアント証明書のシリアル No を設定します。証明書の一意性を高めるため、「シリアル No:Issuer」を設定することも可能です。データ長の制限はありません。この要素は必須です。

CERT_EXPR
 クライアント証明書の有効期限を以下の形式で設定します。
 YYYYMMDDhhmmss
 有効期限長は 14 バイトまでです。この要素は必須です。

REMOTE_ADDR
 クライアントの IP アドレスを設定します。IPv4 アドレスの場合はドット区切り表記、IPv6 アドレスの場合は 8 つの 16 ビットフィールドで構成される 128 ビットで、16 進数値で設定します。この要素は、ログイン時のグループ制御やリクエスト制御で利用されるため、設定することを推奨いたします。REMOTE_ADDR は環境情報として格納されます。

AGENT_ID

リクエスト送信元を特定する識別名を設定します。識別名長の制限はありません。この要素は、認証モジュールのアクセスログに出力されるため（ただし、認証モジュール側の設定による）、設定することを推奨いたします。

USER_AGENT

クライアントで使用されるブラウザの User-Agent を設定します。この要素は将来の拡張において使用される場合があるため、設定することを推奨いたします。

任意の要素名

リクエスト送信元の環境情報として任意の情報を送信することが可能です。名称も任意に設定可能です。ここで送信された情報はバックエンド Web サーバーへ通知する情報継承の対象となります。

使用例

- (1) ユーザー ID が user01、シリアル No が 01、有効期限が 2009 年 9 月 30 日のクライアント証明書を使用して強制ログインを行う場合。

MSG_ID: ReqForceLoginCERT

IW_UID: user01

IW_PWD: password

CERT_NO: 01

CERT_EXPR: 20090930235959

REMOTE_ADDR: 192.168.0.1

AGENT_ID: IceWall SSO DFW

USER_AGENT: Mozilla/x.x

※ データ区切りの改行コード CRLF は省略しています。

備考

- 本リクエストで REMOTE_ADDR を送信しない場合、認証モジュールにて IP アドレスを使用したグルーピング、リクエスト制御は行えなくなります。
- 本リクエストで AGENT_ID を送信しない場合、認証モジュールのアクセスログに AGENT_ID が出力されなくなります。

ユーザー ID とパスワードによるログインレスポンス

概要 ユーザー ID とパスワードによるログインおよび強制ログインリクエストに対して認証モジュールより返信されるレスポンスです。

フォーマット

ログイン成功時
MSG_ID: ResLoginUID_{CRLF}
ERR_NO: ログイン結果_{CRLF}
IW_INFO: セッション ID_{CRLF}
PWD_EXPR: パスワード有効期限_{CRLF}
SID_EXPR: セッション ID 有効期限_{CRLF}
任意の要素名: 任意のデータ_{CRLF}

ログイン失敗時
MSG_ID: ResLoginUID_{CRLF}
ERR_NO: ログイン結果_{CRLF}
任意の要素名: 任意のデータ_{CRLF}

解説 本リクエストを構成する要素は次のとおりです。

MSG_ID
 ResLoginUID が設定されます。この要素は必ず設定されます。

ERR_NO
 ログイン結果が設定されます。設定される値は以下のとおりです。この要素は必ず設定されます。

値	内容
0	ログイン成功
1	ユーザー ID エラー
2	最大ログイン数オーバーエラー
3	二重ログインエラー
4	パスワードエラー
6	ログイン停止エラー
7	所属グループなしエラー
8	並行利用エラー
9	アカウントロックエラー
15	ログイン成功、パスワード有効期限切れによるパスワード変更要求
18	認証 DB ダウンエラー
19	リクエスト実行権限エラー
20	ログイン成功、パスワード有効期限切れ 警告期間内
21	セッション ID 生成エラー
91	ユーザー定義エラー 1

値	内容
92	ユーザー定義エラー 2
93	ユーザー定義エラー 3
99	システムエラー

IW_INFO

セッション ID が設定されます。この要素はログイン結果が「ログイン成功」、「ログイン成功、パスワード有効期限切れによるパスワード変更要求」および「ログイン成功、パスワード有効期限切れ警告期間内」の場合のみ設定されます。

COOKIE 有効期限付加フラグが ON (COOKIEEXP=1) のとき、セッション ID にセミコロン区切りで有効期限が以下の形式で付加されます。

xxx ~ xxx; Expires=Wdy, DD-Mon-YYYY HH:MM:SS GMT;

PWD_EXPR

ログインしたユーザーのパスワード有効期限が設定されます。この要素はログイン結果が「ログイン成功」、「ログイン成功、パスワード有効期限切れによるパスワード変更要求」または「ログイン成功、パスワード有効期限切れ警告期間内」の場合のみ設定されます。

ただし、認証モジュール側でパスワード有効期限の情報を取得していなければ送信されません。

SID_EXPR

ログインしたユーザーのセッション ID のタイムアウト時刻が設定されます。この要素はログイン結果が「ログイン成功」、「ログイン成功、パスワード有効期限切れによるパスワード変更要求」および「ログイン成功、パスワード有効期限切れ警告期間内」の場合のみ設定されます。

ここで設定される有効期限は、ログイン時に算出された時刻となります。認証モジュールの設定によっては、認証モジュール側で保持される有効期限が変化するため、この時間で有効期限切れとならない場合があります。

このデータは認証モジュールの設定に関係なく送信されます。

任意の要素名

認証モジュール側の UserExit ルーチンにて「補足情報」として追加されたものが設定されます。要素の個数には制限はありません。なお、補足情報が存在する場合は、ログインの成功・失敗に関係なく送信されます。

使用例

- (1) ユーザー ID とパスワードによるログインに成功し、COOKIE 有効期限付加フラグが ON の場合

MSG_ID: ResLoginUID

ERR_NO: 0

IW_INFO: xxx ~ xxx; Expires=Sun, 14-Dec-2008 23:59:59 GMT;

PWD_EXPR: 20080930172959

SID_EXPR: 20081214235959

- (2) ユーザー ID とパスワードによるログインに成功したが、パスワードが有

効期限切れ警告期間内の場合

MSG_ID: ResLoginUID

ERR_NO: 20

IW_INFO: xxx ~ xxx

PWD_EXPR: 20080930172959

SID_EXPR: 20081214235959

(3) パスワードエラーによりログインに失敗した場合（補足情報あり）

MSG_ID: ResLoginUID

ERR_NO: 4

REQ_NO: 1073

※ データ区切りの改行コード CRLF は省略しています。

備考

- アクセス制御（ユーザー ID とパスワード）、パスワード変更、ログアウトの各リクエストでは、本リクエストで送信された IW_INFO データの値をセッション ID として使用してください。
- IW_INFO の値に有効期限が付加されている場合はセミコロン以前の文字列をセッション ID として使用してください。

クライアント証明書とパスワードによるログインレスポンス

概要 クライアント証明書とパスワードによるログインおよび強制ログインリクエストに対して認証モジュールより返信されるレスポンスです。

フォーマット

ログイン成功時

MSG_ID: ResLoginCERT_{CRLF}

ERR_NO: ログイン結果_{CRLF}

IW_INFO: セッション ID_{CRLF}

PWD_EXPR: パスワード有効期限_{CRLF}

SID_EXPR: セッション ID 有効期限_{CRLF}

任意の要素名: 任意のデータ_{CRLF}

ログイン失敗時

MSG_ID: ResLoginCERT_{CRLF}

ERR_NO: ログイン結果_{CRLF}

任意の要素名: 任意のデータ_{CRLF}

解説 本リクエストを構成する要素は次のとおりです。

MSG_ID

ResLoginCERT が設定されます。この要素は必ず設定されます。

ERR_NO

ログイン結果が設定されます。設定される値は以下のとおりです。この要素は必ず設定されます。

値	内容
0	ログイン成功
1	ユーザー ID エラー
2	最大ログイン数オーバーエラー
3	二重ログインエラー
4	パスワードエラー
5	証明書シリアル No エラー
6	ログイン停止エラー
7	所属グループなしエラー
9	アカウントロックエラー
15	ログイン成功、パスワード有効期限切れによるパスワード変更要求
18	認証 DB ダウンエラー
19	リクエスト実行権限エラー
20	ログイン成功、パスワード有効期限切れ 警告期間内
21	セッション ID 生成エラー
91	ユーザー定義エラー 1

値	内容
92	ユーザー定義エラー 2
93	ユーザー定義エラー 3
99	システムエラー

IW_INFO

セッション ID が設定されます。この要素はログイン結果が「ログイン成功」、「ログイン成功、パスワード有効期限切れによるパスワード変更要求」および「ログイン成功、パスワード有効期限切れ警告期間内」の場合のみ設定されます。

COOKIE 有効期限付加フラグが ON (COOKIEEXP=1) のとき、セッション ID にセミコロン区切りで有効期限が以下の形式で付加されます。

xxx ~ xxx; Expires=Wdy, DD-Mon-YYYY HH:MM:SS GMT;

PWD_EXPR

ログインしたユーザーのパスワード有効期限が設定されます。この要素はログイン結果が「ログイン成功」、「ログイン成功、パスワード有効期限切れによるパスワード変更要求」または「ログイン成功、パスワード有効期限切れ警告期間内」の場合のみ設定されます。

ただし、認証モジュール側でパスワード有効期限の情報を取得していなければ送信されません。

SID_EXPR

ログインしたユーザーのセッション ID のタイムアウト時刻が設定されます。この要素はログイン結果が「ログイン成功」、「ログイン成功、パスワード有効期限切れによるパスワード変更要求」および「ログイン成功、パスワード有効期限切れ警告期間内」の場合のみ設定されます。

ここで設定される有効期限は、ログイン時に算出された時刻となります。認証モジュールの設定によっては、認証モジュール側で保持される有効期限が変化するため、この時間で有効期限切れとならない場合があります。

このデータは認証モジュールの設定に関係なく送信されます。

任意の要素名

認証モジュール側の UserExit ルーチンにて「補足情報」として追加されたものが設定されます。要素の個数には制限はありません。なお、補足情報が存在する場合は、ログインの成功・失敗に関係なく送信されます。

使用例

- (1) クライアント証明書とパスワードによるログインに成功し、COOKIE 有効期限付加フラグが ON の場合

MSG_ID: ResLoginCERT

ERR_NO: 0

IW_INFO: xxx ~ xxx; Expires=Sun, 14-Dec-2008 23:59:59 GMT;

PWD_EXPR: 20090930172959

SID_EXPR: 20081214235959

- (2) クライアント証明書とパスワードによるログインに成功したが、パス

ワードが有効期限切れの場合
MSG_ID: ResLoginCERT
ERR_NO: 15
IW_INFO: xxx ~ xxx
PWD_EXPR: 20081220172959
SID_EXPR: 20081221235959

- (3) 最大ログイン数オーバーによるエラーのためログインに失敗時した場合
MSG_ID: ResLoginCERT
ERR_NO: 2

※ データ区切りの改行コード CRLF は省略しています。

備考

- アクセス制御（クライアント証明書とパスワード）、パスワード変更、ログアウトの各リクエストでは、本リクエストで送信された IW_INFO データの値をセッション ID として使用してください。
- IW_INFO の値に有効期限が付加されている場合はセミコロン以前の文字列をセッション ID として使用してください。

アクセス制御リクエスト（ユーザー ID とパスワード）

概要 認証モジュールに対して URL のアクセス権限の確認を行う場合に使用するリクエストです。ユーザー ID とパスワードによるログインを行った場合に使用します。

フォーマット

MSG_ID: ReqAccessUID_{CRLF}
IW_INFO: セッション ID_{CRLF}
ACC_URL: アクセス権限チェック URL_{CRLF}
AGENT_ID: リクエスト送信元識別名_{CRLF}
REMOTE_ADDR: クライアント IP アドレス_{CRLF}
METHOD: メソッド名_{CRLF}
USER_AGENT: ユーザーエージェント名_{CRLF}
任意の要素名: 任意のデータ_{CRLF}

解説 本リクエストを構成するデータは次のとおりです。

MSG_ID

ReqAccessUID を設定します。この要素は必須です。

IW_INFO

「ユーザー ID とパスワードによるログインレスポンス」にて送信された

IW_INFO 要素の値を設定します。この要素は必須です。

セッション ID 長は SESSIONIDLEN 項目の設定により、32 バイトもしくは 64 バイトになります。

ACC_URL

アクセス権限チェックを行う URL を設定します。URL は絶対パスで指定します。URL 長は 1024 バイトまでです。この要素は必須です。URL は、URL エンコードしてもしなくてもどちらでも構いません。

AGENT_ID

リクエスト送信元を特定する識別名を設定します。識別名長の制限はありません。この要素は、認証モジュールのアクセスログに出力されるため（ただし、認証モジュール側の設定による）、設定することを推奨いたします。

REMOTE_ADDR

クライアントの IP アドレスを設定します。IPv4 アドレスの場合はドット区切り表記、IPv6 アドレスの場合は 8 つの 16 ビットフィールドで構成される 128 ビットで、16 進数値で設定します。この要素は将来の拡張において使用される場合があるため、設定することを推奨いたします。

METHOD

クライアントから要求されたメソッドを設定します。この要素は将来の拡張において使用される場合があるため、設定することを推奨いたします。

す。

USER_AGENT

クライアントで使用されるブラウザの User-Agent を設定します。この要素は将来の拡張において使用される場合があるため、設定することを推奨いたします。

任意の要素名

リクエスト送信元からリクエスト時のみ有効な付加情報として任意の情報を送信することが可能です。名称も任意に設定可能です。ここで送信された情報は認証モジュールへ通知された後、レスポンス受信と同時に消去されます。

使用例

(1) URL が http://www.hp.com/ のアクセス制御リクエストの場合

MSG_ID: ReqAccessUID
IW_INFO: xxx ~ xxx
ACC_URL: http://www.hp.com/
AGENT_ID: IceWall SSO DFW
REMOTE_ADDR: 192.168.0.1
METHOD: GET
USER_AGENT: Mozilla/x.x

※ データ区切りの改行コード CRLF は省略しています。

備考

- ACC_URL 要素で指定した URL は、そのままアクセス権限チェックに使われます。認証モジュールのアクセスコントロールファイルにて文字の大小を間違えている場合には、想定される結果が返されないことに注意してください。
- 本リクエストで AGENT_ID を送信しない場合、認証モジュールのアクセスログに AGENT_ID が出力されなくなります。

アクセス制御リクエスト（クライアント証明書とパスワード）

概要 認証モジュールに対して URL のアクセス権限の確認を行う場合に使用するリクエストです。クライアント証明書とパスワードによるログインを行った場合に使用します。

フォーマット

MSG_ID: ReqAccessCERT_{CRLF}
IW_INFO: セッション ID_{CRLF}
ACC_URL: アクセス権限チェック URL_{CRLF}
CERT_UID: ユーザー ID_{CRLF}
CERT_NO: シリアル No_{CRLF}
CERT_EXPR: 有効期限_{CRLF}
AGENT_ID: リクエスト送信元識別名_{CRLF}
REMOTE_ADDR: クライアント IP アドレス_{CRLF}
METHOD: メソッド名_{CRLF}
USER_AGENT: ユーザーエージェント名_{CRLF}
任意の要素名: 任意のデータ_{CRLF}

解説

本リクエストを構成するデータは次のとおりです。

MSG_ID
ReqAccessCERT を設定します。この要素は必須です。

IW_INFO
「クライアント証明書とパスワードによるログインレスポンス」にて送信された IW_INFO 要素の値を設定します。この要素は必須です。
セッション ID 長は SESSIONIDLEN 項目の設定により、32 バイトもしくは 64 バイトになります。

ACC_URL
アクセス権限チェックを行う URL を設定します。URL は絶対パスで指定します。URL 長の制限はありません。この要素は必須です。URL は、URL エンコードしてもしなくてもどちらでも構いません。

CERT_UID
アクセス制御を行うユーザーのクライアント証明書に含まれるユーザー ID を設定します。この値はログイン時に設定したユーザー ID と同じ値を指定します。この要素は必須です。
ただしアクセス制御の証明書チェックフラグ (ACCCTRLFLG) が 1 以上を設定されている場合は、この設定がなくともアクセス制御は正常に行われます。

CERT_NO
アクセス制御を行うユーザーのクライアント証明書に含まれるシリアル No を設定します。この値はログイン時に設定したシリアル No と同じ値

を指定します。この要素は将来の拡張において使用される場合があるため、設定することを推奨いたします。

CERT_EXPR

アクセス制御を行うユーザーのクライアント証明書に含まれる有効期限を以下の形式で設定します。

YYYYMMDDhhmmss

この値はログイン時に設定した有効期限と同じ値を指定します。有効期限長は 14 バイトまでです。この要素は将来の拡張において使用される場合があるため、設定することを推奨いたします。

AGENT_ID

リクエスト送信元を特定する識別名を設定します。識別名長の制限はありません。この要素は、認証モジュールのアクセスログに出力されるため（ただし、認証モジュール側の設定による）、設定することを推奨いたします。

REMOTE_ADDR

クライアントの IP アドレスを設定します。IPv4 アドレスの場合はドット区切り表記、IPv6 アドレスの場合は 8 つの 16 ビットフィールドで構成される 128 ビットで、16 進数値で設定します。この要素は将来の拡張において使用される場合があるため、設定することを推奨いたします。

METHOD

クライアントから要求されたメソッドを設定します。この要素は将来の拡張において使用される場合があるため、設定することを推奨いたします。

USER_AGENT

クライアントで使用するブラウザーの User-Agent を設定します。この要素は将来の拡張において使用される場合があるため、設定することを推奨いたします。

任意の要素名

リクエスト送信元からリクエスト時のみ有効な付加情報として任意の情報を送信することが可能です。名称も任意に設定可能です。ここで送信された情報は認証モジュールへ通知された後、レスポンス受信と同時に消去されます。

使用例

(1) URL が http://www.hp.com/ のアクセス制御リクエストの場合

MSG_ID: ReqAccessCERT

IW_INFO: xxx ~ xxx

ACC_URL: http://www.hp.com/

CERT_UID: user01

CERT_NO: 01

CERT_EXPR: 20090930235959

AGENT_ID: IceWall SSO DFW

REMOTE_ADDR: 192.168.0.1

METHOD: GET

USER_AGENT: Mozilla/x.x

※ データ区切りの改行コード CRLF は省略しています。

備考

- ACC_URL 要素で指定した URL は、そのままアクセス権限チェックに使われます。認証モジュールのアクセスコントロールファイルにて文字の大小を間違えている場合には、想定される結果が返されないことに注意してください。
- 本リクエストで AGENT_ID を送信しない場合、認証モジュールのアクセスログに AGENT_ID が出力されなくなります。

アクセス制御レスポンス（ユーザー ID とパスワード）

概要 アクセス制御リクエスト（ユーザー ID とパスワード）に対するレスポンスです。

フォーマット アクセス許可
MSG_ID: ResAccessUID_{CRLF}
ERR_NO: アクセス制御結果_{CRLF}
IW_UID: ユーザー ID_{CRLF}
IW_PWD: パスワード_{CRLF}
認証 DB カラム: ユーザー情報_{CRLF}
環境情報: 環境情報_{CRLF}
任意の要素名: 任意のデータ_{CRLF}

アクセス制御エラー
MSG_ID: ResAccessUID_{CRLF}
ERR_NO: アクセス制御結果_{CRLF}
任意の要素名: 任意のデータ_{CRLF}

解説 本リクエストを構成するデータは次のとおりです。
MSG_ID
ResAccessUID が設定されます。この要素は必ず設定されます。
ERR_NO
アクセス制御結果が設定されます。設定される値は以下のとおりです。この要素は必ず設定されます。

値	内容
0	アクセス許可
10	アクセス権限エラー
11	ログインタイムアウトによる再ログイン要求
15	パスワード有効期限切れによるパスワード変換要求
19	リクエスト実行権限なしエラー
91	ユーザー定義エラー 1
92	ユーザー定義エラー 2
93	ユーザー定義エラー 3
99	システムエラー

IW_UID
ログイン時に指定したユーザー ID が設定されます。この要素はアクセス制御結果が「アクセス許可」かつ、認証モジュール側にて送信する設定の場合のみ設定されます。
IW_PWD

ログイン時に指定したパスワードが設定されます。この要素はアクセス制御結果が「アクセス許可」かつ、認証モジュール側にて送信する設定の場合のみ設定されます。

認証 DB カラム

認証 DB に登録されているユーザー情報で、認証モジュールにて読み込み指定されたカラムが設定されます。この要素はアクセス制御結果が「アクセス許可」かつ、認証モジュール側にて送信する設定の場合のみ設定されます。

環境情報

ユーザーがログインを行う際に環境情報として通知したもののすべてが設定されます。この要素はアクセス制御結果が「アクセス許可」かつ、認証モジュール側にて送信する設定の場合のみ設定されます。

任意の要素名

認証モジュール側の UserExit ルーチンにて「補足情報」として追加されたものが設定されます。要素の個数には制限はありません。なお、補足情報が存在する場合は、アクセスの成功・失敗に関係なく送信されます。

使用例

- (1) アクセス制御結果がアクセス許可の場合

MSG_ID: ResAccessUID
ERR_NO: 0
IW_UID: user01
IW_PWD: password
USERID: user01
PASSWD: {MD5}xxx
LOGINDATE: 20081214091708
REMOTE_ADDR: x.x.x.x

- (2) アクセス制御結果がアクセス許可以外の場合

MSG_ID: ResAccessUID
ERR_NO: 10

※ データ区切りの改行コード CRLF は省略しています。

備考

なし

アクセス制御レスポンス（クライアント証明書とパスワード）

概要 アクセス制御リクエスト（クライアント証明書とパスワード）に対するレスポンスです。

フォーマット アクセス許可
MSG_ID: ResAccessCERT^{CRLF}
ERR_NO: アクセス制御結果^{CRLF}
IW_UID: ユーザー ID^{CRLF}
IW_PWD: パスワード^{CRLF}
認証 DB カラム: ユーザー情報^{CRLF}
環境情報名: 環境情報^{CRLF}
任意のデータ名: 任意のデータ^{CRLF}

アクセス制御エラー
MSG_ID: ResAccessCERT^{CRLF}
ERR_NO: アクセス制御結果^{CRLF}
任意の要素名: 任意のデータ^{CRLF}

解説 本リクエストを構成するデータは次のとおりです。

MSG_ID

ResAccessCERT 設定されます。この要素は必ず設定されます。

ERR_NO

アクセス制御結果が設定されます。設定される値は以下のとおりです。この要素は必ず設定されます。

値	内容
0	アクセス許可
10	アクセス権限エラー
11	ログインタイムアウトによる再ログイン要求
15	パスワード有効期限切れによるパスワード変換要求
19	リクエスト実行権限なしエラー
91	ユーザー定義エラー 1
92	ユーザー定義エラー 2
93	ユーザー定義エラー 3
99	システムエラー

IW_UID

ログイン時に指定した、またはクライアント証明書に格納されているユーザー ID が設定されます。この要素はアクセス制御結果が「アクセス許可」かつ、認証モジュール側にて送信する設定の場合のみ設定されます。

IW_PWD

ログイン時に指定したパスワードが設定されます。この要素はアクセス制御結果が「アクセス許可」かつ、認証モジュール側にて送信する設定の場合のみ設定されます。

認証 DB カラム

認証 DB に登録されているユーザー情報で、認証モジュールにて読み込み指定されたカラムが設定されます。この要素はアクセス制御結果が「アクセス許可」かつ、認証モジュール側にて送信する設定の場合のみ設定されます。

環境情報

ユーザーがログインを行う際に環境情報として通知したものすべてが設定されます。この要素はアクセス制御結果が「アクセス許可」かつ、認証モジュール側にて送信する設定の場合のみ設定されます。

任意の要素名

認証モジュール側の UserExit ルーチンにて「補足情報」として追加されたものが設定されます。要素の個数には制限はありません。なお、補足情報が存在する場合は、アクセスの成功・失敗に関係なく送信されます。

使用例

- (1) アクセス制御結果がアクセス許可の場合

MSG_ID: ResAccessCERT

ERR_NO: 0

IW_UID: user01

IW_PWD: password

USERID: user01

PASSWD: {MD5}xxx

RASERIAL: 123

IWSERIAL: 123

REMOTE_ADDR: x.x.x.x

- (2) アクセス制御結果がアクセス許可以外の場合

MSG_ID: ResAccessCERT

ERR_NO: 10

※ データ区切りの改行コード CRLF は省略しています。

備考

なし

パスワード変更リクエスト

概要 パスワード変更を行う場合に使用するリクエストです。

フォーマット **MSG_ID: ReqPasswdChg**^{CRLF}
 IW_INFO: セッション ID^{CRLF}
 NEW_PWD: パスワード^{CRLF}
 PWD_EXPR: パスワード有効期限^{CRLF}
 CERT_UID: ユーザー ID^{CRLF}
 CERT_NO: シリアル No^{CRLF}
 CERT_EXPR: 有効期限^{CRLF}
 AGENT_ID: リクエスト送信元識別名^{CRLF}
 REMOTE_ADDR: クライアント IP アドレス^{CRLF}
 USER_AGENT: ユーザーエージェント名^{CRLF}
 任意の要素名: 任意のデータ^{CRLF}

解説 本リクエストを構成するデータは次のとおりです。

MSG_ID

ReqPasswdChg を設定します。この要素は必須です。

IW_INFO

「ユーザー ID とパスワードによるログインレスポンス」または「クライアント証明書とパスワードによるログインレスポンス」にて送信された IW_INFO 要素の値を設定します。この要素は必須です。

セッション ID 長は SESSIONIDLEN 項目の設定により、32 バイトもしくは 64 バイトになります。

NEW_PWD

新たに設定するパスワードを設定します。パスワード長は 128 バイトまでです。この要素は必須です。

PWD_EXPR

新たに設定するパスワードの有効期限を設定します。有効期限は日数で設定します。この要素は必須です。

CERT_UID

クライアント証明書を使用時のみ、パスワード変更を行うユーザーのクライアント証明書に含まれるユーザー ID を設定します。この値はログイン時に設定したユーザー ID と同じ値を指定します。ユーザー ID 長は 64 バイトまでです。この要素は将来の拡張において使用される場合があるため、設定することを推奨いたします。

CERT_NO

クライアント証明書を使用時のみ、パスワード変更を行うユーザーのクライアント証明書に含まれるシリアル No を設定します。この値はログイン

ン時に設定したシリアル No と同じ値を指定します。この要素は将来の拡張において使用される場合があるため、設定することを推奨いたします。

CERT_EXPR

クライアント証明書使用時のみ、パスワード変更を行うユーザーのクライアント証明書に含まれる有効期限を以下の形式で設定します。

YYYYMMDDhhmmss

この値はログイン時に設定した有効期限と同じ値を指定します。有効期限長は 14 バイトまでです。この要素は将来の拡張において使用される場合があるため、設定することを推奨いたします。

AGENT_ID

リクエスト送信元を特定する識別名を設定します。識別名長の制限はありません。この要素は、認証モジュールのアクセスログに出力されるため（ただし、認証モジュール側の設定による）、設定することを推奨いたします。

REMOTE_ADDR

クライアントの IP アドレスを設定します。IPv4 アドレスの場合はドット区切り表記、IPv6 アドレスの場合は 8 つの 16 ビットフィールドで構成される 128 ビットで、16 進数値で設定します。この要素は将来の拡張において使用される場合があるため、設定することを推奨いたします。

USER_AGENT

クライアントで使用されるブラウザの User-Agent を設定します。この要素は将来の拡張において使用される場合があるため、設定することを推奨いたします。

任意の要素名

リクエスト送信元からリクエスト時のみ有効な付加情報として任意の情報を送信することが可能です。名称も任意に設定可能です。ここで送信された情報は認証モジュールへ通知された後、レスポンス受信と同時に消去されます。

使用例

(1) 新しいパスワードを「passwd」とし、有効期限を 30 日間とする場合
MSG_ID: ReqPasswdChg
IW_INFO: xxx ~ xxx
NEW_PWD: passwd
PWD_EXPR: 30
AGENT_ID: IceWall SSO DFW
REMOTE_ADDR: 192.168.0.1
USER_AGENT: Mozilla/x.x

※ データ区切りの改行コード CRLF は省略しています。

備考

- 本リクエストで AGENT_ID を送信しない場合、認証モジュールのアクセスログに AGENT_ID が出力されなくなります。

パスワード変更レスポンス

概要 パスワード変更リクエストに対するレスポンスです。

フォーマット パスワード変更成功
MSG_ID: ResPasswdChg^{CRLF}
ERR_NO: パスワード変更結果^{CRLF}
PWD_EXPR: パスワード有効期限^{CRLF}
任意の要素名: 任意のデータ^{CRLF}

パスワード変更失敗
MSG_ID: ResPasswdChg^{CRLF}
ERR_NO: パスワード変更結果^{CRLF}
任意の要素名: 任意のデータ^{CRLF}

解説 本リクエストを構成するデータは次のとおりです。

MSG_ID

ResPasswdChg が設定されます。この要素は必ず設定されます。

ERR_NO

パスワード変更結果が設定されます。設定される値は以下のとおりです。
この要素は必ず設定されます。

値	内容
0	パスワード変更成功
13	パスワード変更エラー
14	パスワード変更権限エラー
16	パスワードポリシーエラー
18	認証 DB ダウンエラー
19	リクエスト実行権限なしエラー
22	パスワード最小長エラー
23	パスワード最大長エラー
24	パスワード文字種エラー
25	ユーザー ID とパスワード同一エラー
91	ユーザー定義エラー 1
92	ユーザー定義エラー 2
93	ユーザー定義エラー 3
99	システムエラー

PWD_EXPR

新たに設定されたパスワードの有効期限が設定されます。この要素は変更結果が成功の場合のみ設定されます。

任意の要素名

認証モジュール側の UserExit ルーチンにて「補足情報」として追加されたものが設定されます。要素の個数には制限はありません。なお、補足情報が存在する場合は、パスワード変更の成功・失敗に関係なく送信されます。

使用例

(1) パスワード変更成功した場合
MSG_ID: ResPasswdChg
ERR_NO: 0
PWD_EXPR: 20090101000000

(2) パスワード変更失敗した場合
MSG_ID: ResPasswdChg
ERR_NO: 16

※ データ区切りの改行コード CRLF は省略しています。

備考

なし

ログアウトリクエスト

概要 ログアウトを行う場合に使用するリクエストです。

フォーマット

MSG_ID: ReqLogout_{CRLF}
IW_INFO: セッション ID_{CRLF}
CERT_UID: ユーザー ID_{CRLF}
CERT_NO: シリアル No_{CRLF}
CERT_EXPR: 有効期限_{CRLF}
AGENT_ID: リクエスト送信元識別名_{CRLF}
REMOTE_ADDR: クライアント IP アドレス_{CRLF}
USER_AGENT: ユーザーエージェント名_{CRLF}
任意の要素名: 任意のデータ_{CRLF}

解説 本リクエストを構成するデータは次のとおりです。

MSG_ID

ReqLogout を設定します。この要素は必須です。

IW_INFO

「ユーザー ID とパスワードによるログインレスポンス」または「クライアント証明書とパスワードによるログインレスポンス」にて送信された IW_INFO 要素の値を設定します。この要素は必須です。

セッション ID 長は SESSIONIDLEN 項目の設定により、32 バイトもしくは 64 バイトになります。

CERT_UID

クライアント証明書を使用時のみ、ログアウトを行うユーザーのクライアント証明書に含まれるユーザー ID を設定します。この値はログイン時に設定したユーザー ID と同じ値を指定します。ユーザー ID 長は 64 バイトまでです。この要素は将来の拡張において使用される場合があるため、設定することを推奨いたします。

CERT_NO

クライアント証明書を使用時のみ、ログアウトを行うユーザーのクライアント証明書に含まれるシリアル No を設定します。この値はログイン時に設定したシリアル No と同じ値を指定します。この要素は将来の拡張において使用される場合があるため、設定することを推奨いたします。

CERT_EXPR

クライアント証明書使用時のみ、ログアウトを行うユーザーのクライアント証明書に含まれる有効期限を以下の形式で設定します。

YYYYMMDDhhmmss

この値はログイン時に設定した有効期限と同じ値を指定します。有効期限長は 14 バイトまでです。この要素は将来の拡張において使用される場合があるため、設定することを推奨いたします。

AGENT_ID

リクエスト送信元を特定する識別名を設定します。識別名長の制限はありません。この要素は、認証モジュールのアクセスログに出力されるため（ただし、認証モジュール側の設定による）、設定することを推奨いたします。

REMOTE_ADDR

クライアントの IP アドレスを設定します。IPv4 アドレスの場合はドット区切り表記、IPv6 アドレスの場合は 8 つの 16 ビットフィールドで構成される 128 ビットで、16 進数値で設定します。この要素は将来の拡張において使用される場合があるため、設定することを推奨いたします。

USER_AGENT

クライアントで使用されるブラウザの User-Agent を設定します。この要素は将来の拡張において使用される場合があるため、設定することを推奨いたします。

任意の要素名

リクエスト送信元からリクエスト時のみ有効な付加情報として任意の情報を送信することが可能です。名称も任意に設定可能です。ここで送信された情報は認証モジュールへ通知された後、レスポンス受信と同時に消去されます。

使用例**(1) ログアウトを行う場合**

MSG_ID: ReqLogout
IW_INFO: xxx ~ xxx
AGENT_ID: IceWall SSO DFW
REMOTE_ADDR: 192.168.50.1
USER_AGENT: Mozilla/x.x

※ データ区切りの改行コード CRLF は省略しています。

備考

- 本リクエストで AGENT_ID を送信しない場合、認証モジュールのアクセスログに AGENT_ID が出力されなくなります。

ログアウトレスポンス

概要 ログアウトリクエストに対するレスポンスです。

フォーマット

ログアウト成功
MSG_ID: ResLogoutCRLF
ERR_NO: ログアウト結果CRLF
IW_INFO: セッション IDCRLF
任意の要素名: 任意のデータCRLF

ログアウト失敗
MSG_ID: ResLogoutCRLF
ERR_NO: ログアウト結果CRLF
任意の要素名: 任意のデータCRLF

解説

本リクエストを構成するデータは次のとおりです。

MSG_ID
レスポンス名が設定されます。「ResLogout」固定となります。この要素は必ず設定されます。

ERR_NO
ログアウト結果が設定します。設定される値は以下のとおりです。この要素は必ず設定されます。

値	内容
0	ログアウト成功
17	ログアウトエラー
19	リクエスト実行権限なしエラー
99	システムエラー

IW_INFO
ログアウトされたセッション ID と過去日付が以下の形式で設定されます。この要素はログアウト結果が「ログアウト成功」の場合のみ設定されます。

xxx ~ xxx; Expires=Thu, 01-Jan-1970 09:00:01 GMT;

任意の要素名
認証モジュール側の UserExit ルーチンにて「補足情報」として追加されたものが設定されます。要素の個数には制限はありません。なお、補足情報が存在する場合は、ログアウトの成功・失敗に関係なく送信されます。

使用例

(1) ログアウトに成功した場合
MSG_ID: ResLogout

ERR_NO: 0

IW_INFO: xxx ~ xxx; Expires=Thu, 01-Jan-1970 09:00:01 GMT;

(2) ログアウトに失敗した場合

MSG_ID: ResPasswdChg

ERR_NO: 17

※ データ区切りの改行コード CRLF は省略しています。

備考

なし

4 通信メッセージ暗号化ライブラリについて (10.0)

本章より通信メッセージ暗号化ライブラリの仕様について記載いたします。

4.1 通信メッセージ暗号化ライブラリとは

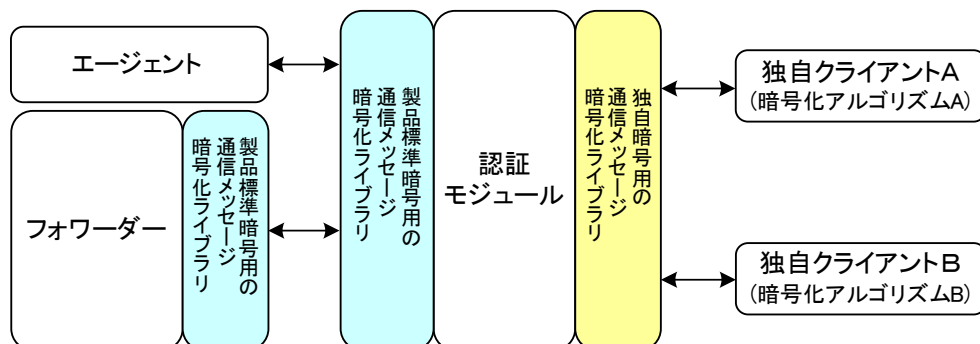
通信メッセージ暗号化ライブラリとは、IceWall SSO の各モジュール間で行われている ICP 2.0 の通信メッセージを暗号化／復号するためのライブラリです。

4.2 通信メッセージ暗号化ライブラリの拡張

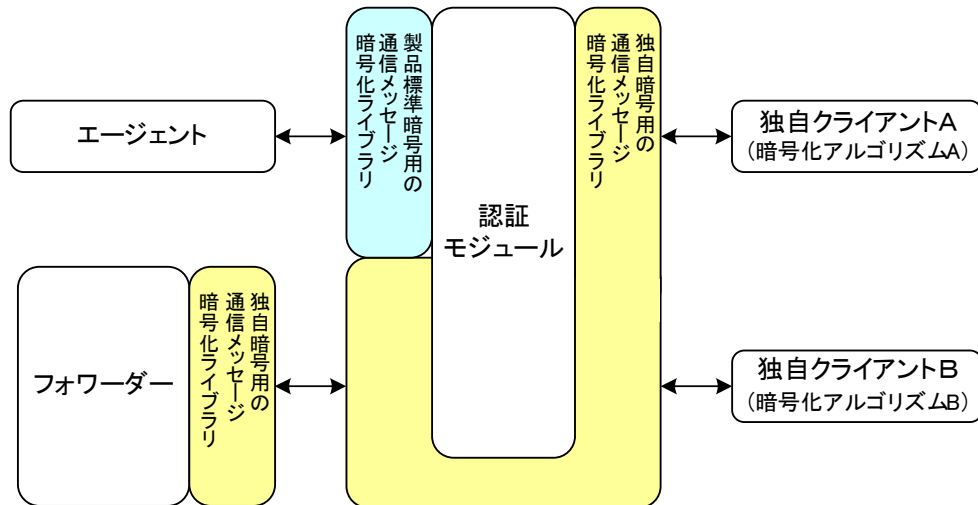
8.0 R3 までの ICP 2.0 の通信メッセージ暗号化ライブラリは、ICP 2.0 通信を行うすべてのモジュール、および独自開発したクライアントで1つの同じ暗号化アルゴリズムである必要がありました。

本バージョンより、製品標準暗号用の通信メッセージ暗号化ライブラリとは別に独自暗号用の通信メッセージ暗号化ライブラリが使用できるようになり、製品標準暗号を使用しつつ独自の暗号化アルゴリズムを使用することが可能になりました。

- (1) 独自クライアントと認証モジュール間を独自開発した暗号化アルゴリズムで通信を行う場合



- (2) フォワーダー、独自クライアントと認証モジュール間を独自開発した暗号化アルゴリズムで通信を行う場合



4.2.1 製品標準暗号用の通信メッセージ暗号化ライブラリ

IceWall SSO の各製品が認証モジュールとの通信時に使用する ICP 2.0 の通信メッセージ暗号化ライブラリです。

8.0 R3 までと同様に、製品標準暗号用の通信メッセージ暗号化ライブラリ自身の開発も従来どおり可能です。

4.2.2 独自暗号用の通信メッセージ暗号化ライブラリ

独自暗号用の ICP 2.0 の通信メッセージ暗号化ライブラリです。

製品標準暗号用の通信メッセージ暗号化ライブラリを使用した ICP2.0 の通信へ影響を与えずに、独自開発した暗号化アルゴリズムで認証モジュールと ICP 2.0 の通信が可能です。

ICP 2.0 のリクエストヘッダーに X-iw-encrypted ヘッダーがあり、その値が icewall 以外の場合、独自の暗号化アルゴリズムを使用したリクエストであると判断し、独自暗号用の通信メッセージ暗号化ライブラリで暗号化／復号が行われます。

独自暗号用の通信メッセージ暗号化ライブラリで暗号化／復号を行うためには、独自暗号用の通信メッセージ暗号化ライブラリで暗号化されたリクエストであることを示す情報を、ICP 2.0 のリクエストヘッダーに追加する必要があります。

- (1) 独自クライアント A からの独自暗号化した ICP 2.0 のユーザーID とパスワードによるログインリクエストの場合

```
REQ / ICP/2.0
X-iw-encrypted: encryptA
Content-length: 55

MSG_ID: ReqLoginUID
IW_UID: UserID
IW_PWD: Password
```

※ Body 部は実際には独自暗号用の通信メッセージ暗号化ライブラリで暗号化されています。

- (2) 独自クライアント B からの独自暗号化した ICP 2.0 のユーザーID とパスワードによるログインリクエストの場合

```
REQ / ICP/2.0
X-iw-encrypted: encryptB
Content-length: 55

MSG_ID: ReqLoginUID
IW_UID: UserID
IW_PWD: Password
```

※ Body 部は実際には独自暗号用の通信メッセージ暗号化ライブラリで暗号化されています。

4.3 通信メッセージ暗号化ライブラリの開発について

製品標準暗号用の通信メッセージ暗号化ライブラリは、暗号化および復号のアルゴリズムを任意のものに変更することができます。

独自暗号用の通信メッセージ暗号化ライブラリは、製品標準暗号用の通信メッセージ暗号化ライブラリとは別に独自の暗号化および復号のアルゴリズムを通信メッセージ暗号化ライブラリとして、使用することができます。

本バージョンより、独自の暗号アルゴリズムで ICP 2.0 の通信メッセージ暗号化ライブラリを開発する場合は、製品標準暗号用の通信メッセージ暗号化ライブラリを使用した通信への影響を与えないようにするため、独自暗号用の通信メッセージ暗号化ライブラリで開発することをおすすめします。

4.3.1 製品標準暗号用の通信メッセージ暗号化ライブラリを開発手順

以下は製品標準暗号用の通信メッセージ暗号化ライブラリを任意の暗号化／復号アルゴリズムに変更するための一般的な開発手順です。

(1) 暗号化／復号の仕様決定

暗号化および復号のアルゴリズムを検討します。

(2) ライブラリの作成

決定したアルゴリズムに基づいてソースコードのコーディングを行います。通信メッセージ暗号化ライブラリは、**Shared Library** として作成します。ライブラリ作成時には API 関数が含まれるアーカイブファイル (libcpt.a) を忘れずにリンクしてください。

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。ビルドはフォワーダー、または認証モジュールの開発キットに含まれる Makefile で行います。

フォワーダー

```
$ cd /opt/icewall-ssso/developkit/dfw/DfwCryptoExit
$ make -f Makefile
```

認証モジュール

```
$ cd /opt/icewall-ssso/developkit/certd/CryptoExit
$ make -f Makefile
```

(3) テスト・デバック

作成した **Shared Library** をテストプログラム等で動作テストおよびデバックを行います。すべてのテストが終了した後にフォワーダー、または認証モジュールへ組み込み、動作確認を行います。

フォワーダーの通信メッセージ暗号化ライブラリのインストールは以下の手順で行います。

(1) 標準でインストールされる製品標準暗号用の通信メッセージ暗号化ライブラリをバックアップします。

```
$ cd /opt/icewall-ssso/lib
$ cp -p libDfwCryptoExit.sl libDfwCryptoExit.sl.bak
```

(2) 作成した製品標準暗号用の通信メッセージ暗号化ライブラリをコピーします。

```
$ cp -p /opt/icewall-ssso/developkit/dfw/DfwCryptoExit/libDfwCryptoExit.sl /opt/icewall-ssso/lib
```

認証モジュールの通信メッセージ暗号化ライブラリのインストールは以下の手順で行います。

(1) 認証モジュールを停止します。

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

- (2) 標準でインストールされる製品標準暗号用の通信メッセージ暗号化ライブラリをバックアップします。

```
$ cd /opt/icewall-ssso/lib
$ cp -p libCryptoExit.sl libCryptoExit.sl.bak
```

- (3) 作成した製品標準暗号用の通信メッセージ暗号化ライブラリをコピーします。

```
$ cp -p /opt/icewall-ssso/developkit/certd/CryptoExit/libCryptoExit.sl /opt/icewall-ssso/lib
```

- (4) 認証モジュールを起動します。

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

4.3.2 独自暗号用の通信メッセージ暗号化ライブラリの開発手順

以下は独自暗号用の通信メッセージ暗号化ライブラリを使用するための一般的な開発手順です。

- (1) 暗号化／復号の仕様決定
暗号化および復号のアルゴリズムを検討します。

- (2) ライブラリの作成
決定したアルゴリズムに基づいてソースコードのコーディングを行います。通信メッセージ暗号化ライブラリは、**Shared Library** として作成します。ライブラリ作成時には API 関数が含まれるアーカイブファイル (libcpt.a) を忘れずにリンクしてください。

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/icewall-ssso/developkit/certd/ExCryptoExit
$ make -f Makefile
```

- (3) テスト・デバック
作成した **Shared Library** をテストプログラム等で動作テストおよびデバックを行います。すべてのテストが終了した後にフォワーダー、または認証モジュールへ組み込み、動作確認を行います。

認証モジュールの通信メッセージ暗号化ライブラリのインストールは以下の手順で行います。

- (1) 認証モジュールを停止します。

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

- (2) 標準でインストールされる独自暗号用の通信メッセージ暗号化ライブラリをバックアップします。

```
$ cd /opt/icewall-ssso/lib
$ cp -p libExCryptoExit.sl libExCryptoExit.sl.bak
```

- (3) 作成した独自暗号用の通信メッセージ暗号化ライブラリをコピーします。

```
$ cp -p /opt/icewall-ssso/developkit/certd/ExCryptoExit/libExCryptoExit.sl /opt/icewall-ssso/lib
```

- (4) 認証モジュールを起動します。

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

4.4 独自暗号用の通信メッセージ暗号化ライブラリを使用する場合の注意点

フォワーダーと認証モジュールとの間で独自暗号方式で ICP 2.0 で通信する場合、以下の手順で行います。

- ・ フォワーダーの標準暗号用の通信メッセージ暗号化ライブラリの開発キットを使用して、独自暗号用のアルゴリズムを実装した通信メッセージ暗号化ライブラリを作成し、インストールします。
- ・ ICP_ENCSTR 項目に「icewall」以外値を設定します。

例) フォワーダーから ICP2.0 で独自暗号化方式「encrypt01」を使用していることを認証モジュールへ通知する場合

```
ICP_ENCSTR=encrypt01
```

5 製品標準暗号用の通信メッセージ暗号化ライブラリ仕様

本章より製品標準暗号用の通信メッセージ暗号化ライブラリの仕様を説明いたします。
公開される関数は、暗号化および復号を行い、コール元ヘデータを返すコールバック関数と、そのコールバック関数内で使用可能な API 関数の 2 種類となります。

5.1 コールバック関数

コールバック関数は次の 2 つとなります。

関数名	処理内容
IW_ExEncrypto()	製品標準暗号用の通信メッセージの暗号化を行う
IW_ExDecrypto()	製品標準暗号用の通信メッセージの復号を行う

各コールバック関数の戻り値は暗号化構造体のアドレスとなります。構造体の仕様は以下のとおりです。

構造体名 : IW_EXCRYPTO

変数名	型	内容
buff	char*	暗号化または復号された通信メッセージのデータが格納された領域のアドレスが格納される。
buflen	int	暗号化または復号された通信メッセージのデータ長が格納される。
result	int	暗号化または復号処理の処理結果が格納される。
errmsg	char*	暗号化または復号処理でエラーが発生した場合、コール元へエラー内容を通知するエラーメッセージが格納される。

各コールバック関数の仕様については次ページより解説いたします。

IW_ExEncrypto

通信メッセージの暗号化を行うコールバック関数の仕様は以下のとおりです。

書式 **IW_EXCRYPTO *IW_ExEncrypto(char* buff, int buflen)**

引数 **buff** :暗号化の対象となるメッセージが格納されている領域のアドレスです。

buflen :メッセージのデータ長です。

戻り値 以下の値を返します。
 NULL 以外 : 暗号化成功または暗号化失敗 (エラーメッセージあり)
 NULL : 暗号化失敗 (エラーメッセージなし)

解説 本コールバック関数では以下の処理を行ってください。

- (1) メッセージの暗号化处理
- (2) 暗号化されたメッセージを暗号化構造体へ格納

戻り値の暗号化構造体のメンバーに設定されている値により認証モジュールの動作が以下のように変化します。

- **buff ≠ NULL** かつ **buflen ≥ 1** かつ **result = 0**
暗号化成功。errmsg は無視されます。
- 成功以外 (**buff = NULL** または **buflen < 1** または **result ≠ 0**)
暗号化失敗とみなし、**result** および **errmsg** の値をメッセージとしてエラーログに出力します。

制限事項 本コールバック関数の戻り値は後述の IW_ExCPTCreateCrypto0 で取得した暗号化構造体のアドレスおよび NULL 以外は設定しないでください。

本コールバック関数の引数は参照専用です。値の変更等を行わないでください。

IW_ExDecrypto

通信メッセージの復号を行うコールバック関数の仕様は以下のとおりです。

書式 **IW_EXCRYPTO *IW_ExDecrypto(char* buff, int buflen)**

引数 **buff** : 復号の対象となるメッセージが格納されている領域のアドレスです。

buflen : メッセージのデータ長です。

戻り値 以下の値を返します。
 NULL 以外 : 復号成功または暗号化失敗 (エラーメッセージあり)
 NULL : 復号失敗 (エラーメッセージなし)

解説 本コールバック関数では以下の処理を行ってください。

- (1) メッセージの復号処理
- (2) 復号されたメッセージを暗号化構造体へ格納

戻り値の暗号化構造体のメンバーに設定されている値により認証モジュールの動作が以下のように変化します。

- **buff ≠ NULL** かつ **buflen ≥ 1** かつ **result = 0**
復号成功。errmsg は無視されます。
- 成功以外 (**buff = NULL** または **buflen < 1** または **result ≠ 0**)
復号失敗とみなし、**result** および **errmsg** の値をメッセージとしてエラーログに出力します。

制限事項 本コールバック関数の戻り値は後述の IW_ExCPTCreateCrypto0 で取得した暗号化構造体のアドレスおよび NULL 以外は設定しないでください。

本コールバック関数の引数は参照専用です。値の変更等を行わないでください。

5.2 API 関数

コールバック関数内でコール可能な API 関数は次のとおりです。

API 名称	処理内容
IW_ExCPTCreateCrypto0	暗号化構造体の作成を行う
IW_ExCPTReleaseCrypto0	暗号化構造体の解放を行う

各 API の関数仕様については次ページより解説いたします。

IW_ExCPTCreateCrypto

コールバック関数の戻り値となる暗号化構造体を作成するための API の仕様は以下のとおりです。

書式 **IW_EXCRYPTO *IW_ExCPTCreateCrypto(char* buff, int len, int result, char* errmsg)**

引数 **buff** : コールバック関数の戻り値として返す暗号化（または復号）の対象となるメッセージが格納されている領域のアドレスを指定します。

len : メッセージのデータ長を指定します。

result : コールバック関数を成功終了する際は 0 を指定します。エラーで終了する際はエラーコードを指定します。エラーコードは 0 以外の任意な値を指定可能です。

errmsg : コールバック関数をエラーで終了する際のエラーメッセージが格納されている領域のアドレスを指定します。

戻り値 以下の値を返します。
 NULL 以外 : 暗号化構造体のアドレス
 NULL : 内部エラーにより値を取得できなかった

制限事項 引数 len が引数 buff に格納されているデータより少ない値の場合、len で指定されたデータ長までが有効となります。

引数 len が 0 以下の場合、引数 buff にデータが存在していても格納されません。

引数 buff と len が NULL と 0 以下でも、引数 result や errmsg が指定されている場合は格納されます。

引数 errmsg が 200 文字を超える場合、認証モジュールではエラーログにメッセージを出力する際に末尾がカットされることがあります。

本 API を実行して取得した領域をコールバック関数の戻り値として使用しない場合には、IW_ExCPTReleaseCrypto() をコールして領域の破棄を行ってください。

IW_ExCPTReleaseCrypto

IW_ExCPTCreateCrypto() で作成された暗号化構造体を解放する API の仕様は以下のとおりです。

書式 **int IW_ExCPTReleaseCrypto (IW_EXCRYPTO* excrypto)**

引数 **excrypto** : 暗号化構造体のアドレスです。

戻り値 以下の値を返します。
 0 : 解放成功
 0 以外: 解放失敗

制限事項 本 API をコールする際の引数には IW_ExCPTCreateCrypto() により取得した暗号化構造体のアドレス以外を指定しないでください。

6 独自暗号用の通信メッセージ暗号化ライブラリ仕様 [10.0]

本章より独自暗号用の通信メッセージ暗号化ライブラリの仕様を説明いたします。
公開される関数は、暗号化および復号を行い、コール元ヘデータを返すコールバック関数と、そのコールバック関数内で使用可能な API 関数の 2 種類となります。

6.1 コールバック関数

コールバック関数は次の 2 つとなります。

関数名	処理内容
CS_ExEncrypto()	独自暗号用の通信メッセージの暗号化を行う
CS_ExDecrypto()	独自暗号用の通信メッセージの復号を行う

各コールバック関数の戻り値は暗号化構造体のアドレスとなります。構造体の仕様は以下のとおりです。

構造体名 : IW_EXCRYPTO

変数名	型	内容
req	CS_REQ_DATA*	リクエスト情報の通信メッセージのデータが格納された領域のアドレスが格納される。
buff	char*	暗号化または復号された通信メッセージのデータが格納された領域のアドレスが格納される。
buflen	int	暗号化または復号された通信メッセージのデータ長が格納される。

各コールバック関数の仕様については次ページより解説いたします。

CS_ExEncrypto

独自暗号用の通信メッセージの暗号化を行うコールバック関数の仕様は以下のとおりです。

書式 **IW_EXCRYPTO *CS_ExEncrypto(CS_REQ_DATA *req, char *buff, int buflen)**

引数 **req** : リクエスト情報が格納されている領域のアドレスです。

buff : 暗号化の対象となるメッセージが格納されている領域のアドレスです。

buflen : メッセージのデータ長です。

戻り値 以下の値を返します。
 NULL 以外 : 暗号化成功または暗号化失敗 (エラーメッセージあり)
 NULL : 暗号化失敗 (エラーメッセージなし)

解説 本コールバック関数では以下の処理を行ってください。

- (1) 引数 **req** が NULL または異常な場合は異常終了
- (2) メッセージの暗号化処理
- (3) 暗号化されたメッセージを暗号化構造体へ格納

戻り値の暗号化構造体のメンバーに設定されている値により認証モジュールの動作が以下のように変化します。

- **buff ≠ NULL** かつ **buflen ≥ 1** かつ **result = 0**
暗号化成功。**errmsg** は無視されます。
- 成功以外 (**buff = NULL** または **buflen < 1** または **result ≠ 0**)
暗号化失敗とみなし、**result** および **errmsg** の値をメッセージとしてエラーログに出力します。

制限事項 本コールバック関数の戻り値は後述の **IW_ExCPTCreateCrypto0** で取得した暗号化構造体のアドレスおよび NULL 以外は設定しないでください。

本コールバック関数の引数は参照専用です。値の変更等は行わないでください。

CS_ExDecrypto

独自暗号用の通信メッセージの復号を行うコールバック関数の仕様は以下のとおりです。

書式 **IW_EXCRYPTO *CS_ExDecrypto(CS_REQ_DATA *req, char *buff, int buflen)**

引数 **req** : リクエスト情報が格納されている領域のアドレスです。

buff : 復号の対象となるメッセージが格納されている領域のアドレスです。

buflen : メッセージのデータ長です。

戻り値 以下の値を返します。
 NULL 以外 : 復号成功または復号失敗 (エラーメッセージあり)
 NULL : 復号失敗 (エラーメッセージなし)

解説 本コールバック関数では以下の処理を行ってください。

- (1) 引数 **req** が NULL または異常な場合は異常終了
- (2) メッセージの復号処理
- (3) 復号されたメッセージを暗号化構造体へ格納

戻り値の暗号化構造体のメンバーに設定されている値により認証モジュールの動作が以下のように変化します。

- **buff ≠ NULL** かつ **buflen ≥ 1** かつ **result = 0**
復号成功。errmsg は無視されます。
- 成功以外 (**buff = NULL** または **buflen < 1** または **result ≠ 0**)
復号失敗とみなし、**result** および **errmsg** の値をメッセージとしてエラーログに出力します。

制限事項 本コールバック関数の戻り値は後述の IW_ExCPTCreateCrypto0 で取得した暗号化構造体のアドレスおよび NULL 以外は設定しないでください。

本コールバック関数の引数は参照専用です。値の変更等は行わないでください。

6.2 API 関数

コールバック関数内でコール可能な API 関数は次のとおりです。

API 名称	処理内容
IW_ExCPTCreateCrypto()	暗号化構造体の作成を行う
IW_ExCPTReleaseCrypto()	暗号化構造体の解放を行う
CS_ExGetIWEncryptedType()	ICP2.0 のリクエストに含まれる X-iw-encrypted ヘッダーの値を取得する。

各 API の関数仕様については次ページより解説いたします。

IW_ExCPTCreateCrypto

コールバック関数の戻り値となる暗号化構造体を作成するための API の仕様は以下のとおりです。

書式 **IW_EXCRYPTO *IW_ExCPTCreateCrypto(char* buff, int len, int result, char* errmsg)**

引数 **buff** : コールバック関数の戻り値として返す暗号化（または復号）の対象となるメッセージが格納されている領域のアドレスを指定します。

len : メッセージのデータ長を指定します。

result : コールバック関数を成功終了する際は 0 を指定します。エラーで終了する際はエラーコードを指定します。エラーコードは 0 以外の任意な値を指定可能です。

errmsg : コールバック関数をエラーで終了する際のエラーメッセージが格納されている領域のアドレスを指定します。

戻り値 以下の値を返します。
 NULL 以外 : 暗号化構造体のアドレス
 NULL : 内部エラーにより値を取得できなかった

制限事項 引数 len が引数 buff に格納されているデータより少ない値の場合、len で指定されたデータ長までが有効となります。

引数 len が 0 以下の場合、引数 buff にデータが存在していても格納されません。

引数 buff と len が NULL と 0 以下でも、引数 result や errmsg が指定されている場合は格納されます。

引数 errmsg が 200 文字を超える場合、認証モジュールではエラーログにメッセージを出力する際に末尾がカットされることがあります。

本 API を実行して取得した領域をコールバック関数の戻り値として使用しない場合には、IW_ExCPTReleaseCrypto() をコールして領域の破棄を行ってください。

IW_ExCPTReleaseCrypto

IW_ExCPTCreateCrypto() で作成された暗号化構造体を解放する API の仕様は以下のとおりです。

書式 **int IW_ExCPTReleaseCrypto (IW_EXCRYPTO* excrypto)**

引数 **excrypto** : 暗号化構造体のアドレスです。

戻り値 以下の値を返します。
 0 : 解放成功
 0 以外: 解放失敗

制限事項 本API をコールする際の引数には IW_ExCPTCreateCrypto() により取得した暗号化構造体のアドレス以外を指定しないでください。

CS_ExGetIWEncryptedType

独自暗号の暗号化方式を取得する API の仕様は以下のとおりです。

書式 `char *CS_ExGetIWEncryptedType( CS_REQ_DATA *req )`

引数 **req** : リクエスト情報が格納されている領域のアドレスを指定します。

戻り値 以下の値を返します。
 NULL 以外 : X-iw-encrypted ヘッダーの文字列
 X-iw-encrypted ヘッダーがない場合は、icewall が返されます。
 NULL : 内部エラーにより値を取得できなかった

制限事項 本 API は X-iw-encrypted ヘッダー値の参照専用です。値の変更等を行わないでください。

7 ICP を実装する際の注意および制限事項

本章では ICP 2.0 を実装する場合の注意事項および制限事項を記載します。

7.1 注意事項

ICP は暗号化されたメッセージで通信することが前提となっています。暗号化アルゴリズムについては、認証モジュールで使用する通信メッセージ暗号化ライブラリと同じものを使用してください。

7.2 制限事項

本仕様は予告なく変更される場合があります。

8 通信メッセージ暗号化ライブラリを開発する際の注意および制限事項

本章では通信メッセージ暗号化ライブラリを開発する際の注意事項および制限事項を記載します。

8.1 注意事項

(1) ライブラリの影響範囲

ライブラリ内で実行される処理は、認証モジュールの内部処理と密接な関係にあるため、ライブラリ内でシステム的なエラーが発生した場合は、そのプロセスがダウンする可能性があります。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、ライブラリ内で `exit()` をコールしますと、そのプロセスが終了しますので、記述しないでください。

(2) パフォーマンスの低下

暗号化および復号処理で高負荷の処理を行った場合、認証モジュール全体のパフォーマンスが低下することがあります。

(3) メモリリーク

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

(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暗号化ロジック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保存しておいてください。

8.2 制限事項

本ライブラリを作成するにあたり、以下の制限事項が存在す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、認証モジュール自体が実行できない場合や、動作が不安定になる場合があります。

(1) 認証モジュールと、リンクするライブラリのビット数を合わせること。

認証モジュールはビット数が一致するライブラリを利用する前提で作成されているため、必ずライブラリのビット数を合わせてください。

例えば、64 ビット版の認証モジュールと 32 ビットで開発されたライブラリとの組み合わせでは使用できません。

(2) 標準ライブラリ関数およびユーザー関数はスレッドセーフ版を使用すること。

認証モジュールは内部処理にスレッドを使用しています。通信メッセージ暗号化ライブラリは、スレッド内のファンクションとして、認証モジュールよりコールされます。

このため、通信メッセージ暗号化ライブラリ内で使用する標準ライブラリ関数およびユーザー関数は、スレッドセーフ版を使用してください。

- (3) バージョンの異なる標準ライブラリをアーカイブでリンクしないこと。
認証モジュールは、ダイナミックにライブラリを使用して動作するように作成されています。
ただし、通信メッセージ暗号化ライブラリにアーカイブライブラリをリンクしている場合には、それにリンクされているライブラリが利用されることになるため、バージョンの違いによる不整合を招く可能性があります。(OS への Patch を適用しても解決しない、等のトラブルになる可能性があります)

注) 特に Oracle 等、インストール環境によって生成されるライブラリ構成が変わる (libclntsh.sl の構成が変わる) ものを通信メッセージ暗号化ライブラリで利用する場合にはリンクされているライブラリの状態について十分注意する必要があります。

9 ICP2.0 クライアントサンプル

本章では、ICP 2.0 を使用する場合のクライアントのサンプルソースコードを記載します。サンプルは C 言語版と Java 版でそれぞれ 2 種類となります。

なお、各サンプルの仕様は次のようになります。

- ・ ログインに失敗した場合、アクセス制御は行いません。
- ・ メッセージ暗号化／復号アルゴリズムは、「10 通信メッセージ暗号化ライブラリサンプル」に掲載しているサンプルを使用するものとします。
- ・ ログインを行うユーザーは「ユーザー ID=01」「パスワード=01」とする。
- ・ 認証モジュールにてログインを行うユーザーのグループ設定およびアクセスコントロール設定が行われているものとします。

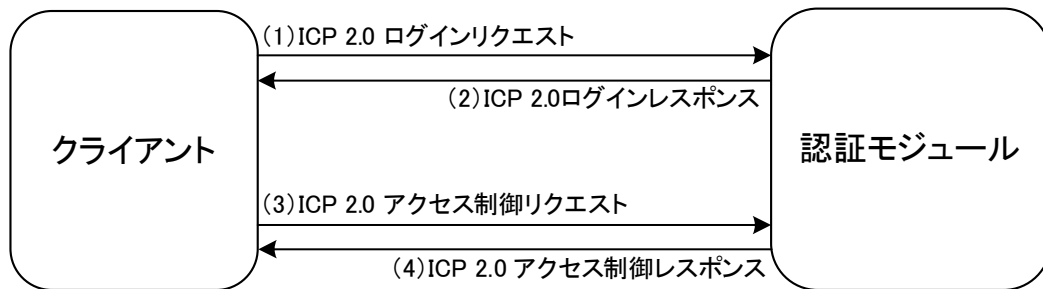
なお、IceWall SSO Version 8.0 Agent Option 2007 Update Release 2 でリリースされた Java Agent Library (ICP 2.0 対応版) を使用することで、本書の Java 用サンプルのようなソースコードを作成せずに、ICP 2.0 で認証モジュールと通信することができます。

サンプルソースの構成は以下のディレクトリおよびファイルです。

ディレクトリ			内容
/opt/icewall-ss0/sample_src/ICP20	/sample1	Makefile	メイクファイル
		Sample1.java	クライアントサンプル 1 (Java 言語用)
		sample1.c	クライアントサンプル 1 (C 言語用)
	/sample2	Makefile	メイクファイル
		Sample2.java	クライアントサンプル 2 (Java 言語用)
		sample2.c	クライアントサンプル 2 (C 言語用)
	/agent	Makefile	メイクファイル
		mod_sample.c	エージェントサンプル (C 言語用)

9.1 クライアントサンプル 1

クライアントより ICP 2.0 を使用して認証モジュールへのログインリクエスト、アクセス制御リクエストを送信するサンプルです。

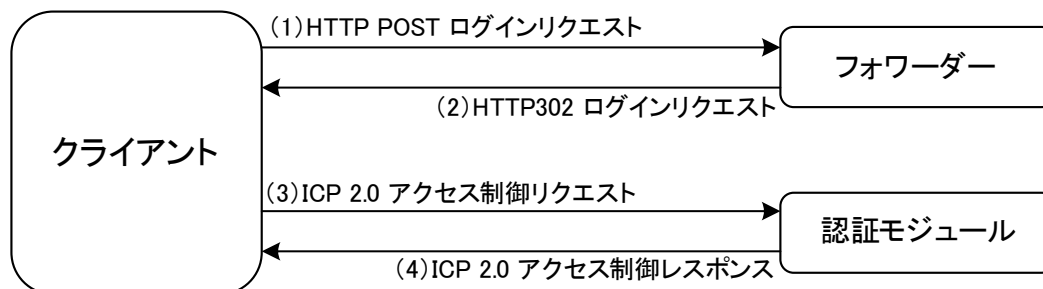


このサンプルでは以下の処理を行います。

- (1) ログインリクエスト電文を作成する。
- (2) 認証モジュールへ ICP 2.0 でログインリクエストを送信する。
- (3) 認証モジュールより ICP 2.0 でログインレスポンスを受信する。
- (4) ログインレスポンスよりセッション ID を取得する。
- (5) 取得したセッション ID を使用してアクセス制御リクエスト電文を作成する。
- (6) 認証モジュールへ ICP 2.0 でアクセス制御リクエストを送信する。
- (7) 認証モジュールより ICP 2.0 でアクセス制御レスポンスを受信する。
- (8) アクセス制御レスポンスよりアクセス制御結果を取得する。

9.2 クライアントサンプル 2

クライアントより HTTP 通信でフォワーダーへログインを行い、その後 ICP 2.0 を使用してアクセス制御リクエストを認証モジュールへ送信するサンプルです。



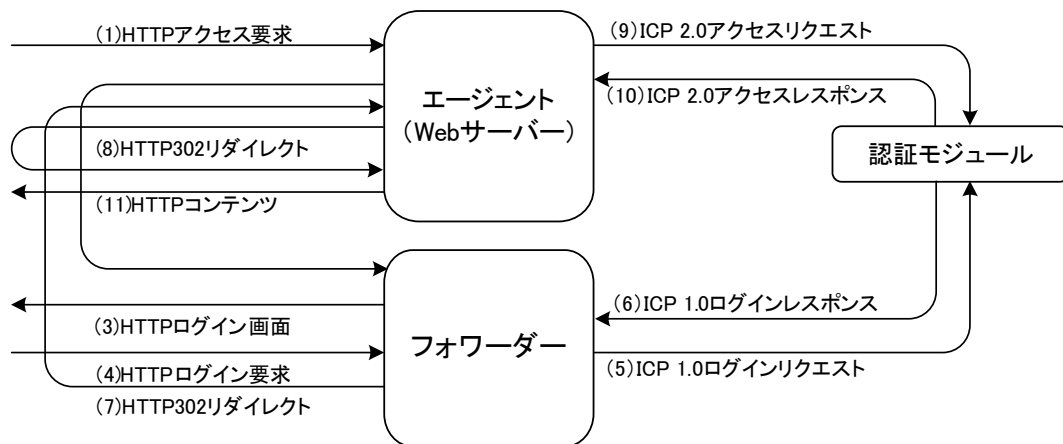
このサンプルでは以下の処理を行います。

- (1) ユーザー ID とパスワードによるログインリクエスト電文を作成する。
- (2) フォワーダーへ HTTP で POST メソッドにて送信する。
- (3) フォワーダーよりログインレスポンスを受信する。
- (4) ログインレスポンスよりセッション ID を取得する。
- (5) 取得したセッション ID で、認証モジュールへ送信するアクセス制御リクエスト電文を作成する。
- (6) 認証モジュールへ ICP 2.0 でアクセス制御リクエストを送信する。
- (7) 認証モジュールより ICP 2.0 でアクセス制御レスポンスを受信する。

(8) アクセス制御結果を取得する。

9.3 エージェントサンプル (Apache HTTP Server 1.3.x 用)

Web サーバーに組み込むエージェントより ICP 2.0 を使用して認証モジュールへアクセス制御リクエストを送信するサンプルです。なお、ログイン要求は Agent Option 同様にフォワーダーにて行い、ログイン後に制御がエージェントに戻るものとなります。



このサンプルでは以下の処理を行います。

- (1) クライアントから HTTP でアクセス要求をエージェント (Web サーバー) に送信する。
- (2) エージェントは IceWall にログインしているかどうかを判断し、未ログインの場合はフォワーダーへリダイレクトを行う。
- (3) フォワーダーはクライアントへログイン画面を送信する。
- (4) クライアントから入力されたユーザーID とパスワードで IceWall へログイン要求をフォワーダーへ送信する。
- (5) フォワーダーは認証モジュールへ ICP 1.0 でログインリクエストを送信する。
- (6) フォワーダーは認証モジュールより ICP 1.0 でログインレスポンスを受信する。
- (7) フォワーダーはログイン結果が OK の場合、エージェントへリダイレクトすると同時にセッション ID をクライアントへ送信する。
- (8) エージェントはフォワーダーより送信されたセッションIDをクライアントへ通知するためにリダイレクトを行う。リダイレクト先は (1) で指定された URL となる。
- (9) エージェントは認証モジュールへ ICP 2.0 でアクセス制御リクエストを送信する。
- (10) エージェントは認証モジュールより ICP 2.0 でアクセス制御レスポンスを受信する。
- (11) アクセス制御結果が OK の場合、リクエストされたコンテンツをクライアントへ送信する。

この処理では Agent Option と同じシーケンスを使用します。このため、決められたフォーマットにて必要な情報を送信する必要があります。各フォーマットを以下に示します。

処理番号	フォーマット
(2)	HTTP/1.1 302 Moved Temporary Location: [フォワーダーの URL]?[識別キーワード]=[リクエストされた URL のサーバー名部分][識別ダミーパス]&path=[リクエストされた URL のパス部分]&query=[リクエストされた引数]
(7)	HTTP/1.1 302 Moved Temporary Location: [リクエストされた URL のサーバー名部分][識別ダミーパス]?[識別キーワード]=[セッション ID]& path=[リクエストされた URL のパス部分]&query=[リクエストされた引数]
(8)	HTTP/1.1 302 Moved Temporary Location: [リクエストされた URL]? [リクエストされた引数] Set-Cookie: [認証 Cookie 名]=[セッション ID]

各パラメーター（斜体部分）は以下のように取得します。

パラメーター名	取得先
フォワーダーの URL	Web サーバーの設定ファイル (httpd.conf) の DFW_PATH より取得します。
識別キーワード	固定値「AGENT_KEY」を使用します。
リクエストされた URL	(1) でクライアントから指定された URL を取得します。
リクエストされた引数	(1) でクライアントから指定された URL の引数部分を取得します。
識別ダミーパス	固定値「/sample」を使用します。
認証 Cookie 名	固定値「IW_INFO」を使用します。

このエージェントサンプルは次の仕様となります。

- ・ 接続する認証モジュール情報は固定値を使用します。
- ・ リダイレクト時に各パラメーターは URL エンコードを行います。
- ・ ユーザー ID およびセッション ID のみ HTTP ヘッダー情報として追加します。
- ・ 何らかのエラーが発生、またはアクセス制御で NG となった場合はステータス 403 で終了します。
- ・ ログは出力しません。
- ・ 本サンプルは IceWall サーバーと同じ Web サーバー上では動作しません。

このエージェントサンプルを仕様する場合は Web サーバーの設定ファイル (httpd.conf) に次の設定を行う必要があります。

```

~
LoadModule sample_module /opt/icewall-ss0/sample_src/ICP20/agent/mod_sample.so
AddModule mod_sample.c
~
<VirtualHost _default_:80>
~
  <Location />
    DFW_PATH http://dfw_host/fw/dfw

```

```
ErrorDocument 403 "mod_sample error"
</Location>
~
</VirtualHost>
```

LoadModule 行ではサンプルエージェントのライブラリをフルパスで指定します。
 AddModule 行ではモジュールのメイン関数が存在するソースファイルを指定します。
 DFW_PATH 行ではログインを行うフォワーダーの URL を設定します。
 これらの設定を行った後、Web サーバーの再起動を行ってください。

9.4 サンプルソース (C 言語)

以下は C 言語によるサンプルソースです。

9.4.1 クライアントサンプル 1 (sample1.c)

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <netdb.h>
#include <netinet/in.h>
#include <sys/ioctl.h>
#include <sys/socket.h>

#define LOGIN_MODE 1
#define ACCESS_MODE 2
#define CRYPTOKEY "ygHxXW7Y+LiQDz7YUyIh4I9Ig28X6VaYAHMmFIXGEnJXEAXx7hGQgY"

int CreateRequest( int mode, char *request, char *sessionid );
void CheckResponse( int mode, char *response, char *data );
void SocketCertd( int sendsize, char *request, char *response );
void ChangeEOF( char *buff, int len );

/*-----*/
int main( int argc, char **argv )
{
    char request[4096];
    char response[4096];
    char sessionid[33];
    char errcode[3];
    int sendsize;

    /* Login */
    sendsize = CreateRequest( LOGIN_MODE, request, NULL );
    SocketCertd( sendsize, request, response );
    CheckResponse( LOGIN_MODE, response, sessionid );
```



```
/* Login Error */
if( strlen( sessionid ) == 0 ) exit( -1 );

/* Access */
sendsize = CreateRequest( ACCESS_MODE, request, sessionid );
SocketCertd( sendsize, request, response );
CheckResponse( ACCESS_MODE, response, errcode );

exit( atoi( errcode ) );
}

void SocketCertd( int sendsize, char *request, char *response )
{
    int          sock;
    struct sockaddr_in addr;
    struct hostent *hent;

    /* Certd ( localhost:14142 ) */
    memset( &addr, 0x00, sizeof(struct sockaddr_in) );
    addr.sin_family = AF_INET;
    addr.sin_port   = htons( 14142 );
    hent = gethostbyname( "localhost" );
    memcpy( &addr.sin_addr, hent->h_addr, hent->h_length );

    sock = socket( AF_INET, SOCK_STREAM, 0 );

    connect( sock, (struct sockaddr*)&addr, sizeof( addr ) );

    send( sock, request, sendsize, 0 );

    recv( sock, response, 4096, 0 );

    shutdown( sock, SHUT_RDWR );
    close( sock );
}

int CreateRequest( int mode, char *request, char *sessionid )
{
    /* Messeage -> Encrypto */
    switch( mode ) {
    case LOGIN_MODE:
        strcpy( request,
            "REQ / ICP/2.0\r\nContent-length: 43\r\n\r\nMSG_ID: ReqLoginUID\r\nIW_UID: 01\r\nIW_PWD: 01" );
        ChangeEOF( ( request + 37 ), 43 );
        break;

    case ACCESS_MODE:
        sprintf( request,
            "REQ / ICP/2.0\r\nContent-length: 78\r\n\r\nMSG_ID: ReqAccessUID\r\nIW_INFO: %s\r\nACC_URL: http",
            sessionid );
    }
```

```
        ChangeEOF( ( request + 37 ), 78 );
        break;
    }

    return( strlen( request ) );
}

void CheckResponse( int mode, char *response, char *data )
{
    char    *ptr;
    char    *buff;
    int     len;

    /* Decrypto */
    buff = strstr( response, "\r\n\r\n" ) + 4;
    len = atoi( strstr( response, "Content-length: " ) + 16 );
    ChangeEOF( buff, len );

    switch( mode ) {
    case LOGIN_MODE:
        /* Get SessionID */
        ptr = strstr( buff, "IW_INFO: " ) + 9;
        if( ptr != NULL ) strncpy( data, ptr, 32 );
        break;

    case ACCESS_MODE:
        /* Get ErrNo */
        ptr = strstr( buff, "ERR_NO: " ) + 8;
        if( ptr != NULL ) strncpy( data, ptr, 2 );
        break;
    }
}

void ChangeEOF( char *buff, int len )
{
    char ascii;
    int count;
    int index;

    index = 0;
    for( count = 0; count < len; count++ ){

        /* ASCII 4bit */
        ascii = CRYPTOKEY[index++] & 0x0F;
        if( index >= 54 ) index = 0;

        /* EOR */
        buff[count] ^= ascii;

        /* 0x7f */
        if( buff[count] == 0x7F ) buff[count] ^= ascii;
    }
}
```

}

9.4.2 クライアントサンプル2 (sample2.c)

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <netdb.h>
#include <netinet/in.h>
#include <sys/ioctl.h>
#include <sys/socket.h>

#define LOGIN_MODE 1
#define ACCESS_MODE 2
#define CRYPTOKEY "ygHxXW7Y+LiQDz7YUyIh4I9Ig28X6VaYAHMmFIXGEn
JXEAXx7hGQgY"

int CreateRequest( int mode, char *request, char *sessionid );
void CheckResponse( int mode, char *response, char *data );
void SocketMessage( int port, int sendsize, char *request, char *response );
void ChangeEOF( char *buff, int len );

/*-----*/
int main( int argc, char **argv )
{
    char request[4096];
    char response[4096];
    char sessionid[33];
    char errcode[3];
    int sendsize;

    /* Login : dfw(localhost:80) */
    sendsize = CreateRequest( LOGIN_MODE, request, NULL );
    SocketMessage( 80, sendsize, request, response );
    CheckResponse( LOGIN_MODE, response, sessionid );

    /* Login Error */
    if( strlen( sessionid ) == 0 ) exit( -1 );

    /* Access : certd(localhost:14142) */
    sendsize = CreateRequest( ACCESS_MODE, request, sessionid );
    SocketMessage( 14142, sendsize, request, response );
    CheckResponse( ACCESS_MODE, response, errcode );

    exit( atoi( errcode ) );
}

void SocketMessage( int port, int sendsize, char *request, char *response )
{
    int          sock;
```

```
struct sockaddr_in addr;
struct hostent *hent;

memset( &addr, 0x00, sizeof(struct sockaddr_in) );
addr.sin_family = AF_INET;
addr.sin_port = htons( port );
hent = gethostbyname( "localhost" );
memcpy( &addr.sin_addr, hent->h_addr, hent->h_length );

sock = socket( AF_INET, SOCK_STREAM, 0 );

connect( sock, (struct sockaddr*)&addr, sizeof( addr ) );

send( sock, request, sendsize, 0 );

recv( sock, response, 4096, 0 );

shutdown( sock, SHUT_RDWR );
close( sock );
}

int CreateRequest( int mode, char *request, char *sessionid )
{
    switch( mode ) {
        case LOGIN_MODE:
            /* Login PostData */
            strcpy( request,
                "POST /fw/dfw HTTP/1.0\r\nContent-length: 84\r\nHost: localhost:80\r\n\r\nACCOUNTUID=01&PASSWORD=01&HIDEURL=%2Fhttp%2F&LOGIN=ICEWALL_LOGIN&SUBMIT=+%91%97%90M+" );
            break;

        case ACCESS_MODE:
            /* Icp20 Access */
            sprintf( request,
                "REQ / ICP/2.0\r\nContent-length: 78\r\n\r\nMSG_ID: ReqAccessUID\r\nIW_INFO: %s\r\nACC_URL: http",
                sessionid );

            /* Encrypto */
            ChangeEOF( ( request + 37 ), 78 );
            break;
    }

    return( strlen( request ) );
}

void CheckResponse( int mode, char *response, char *data )
{
    char *ptr;
    char *buff;
    int len;
```

```
switch( mode ) {
case LOGIN_MODE:
    /* Get SessionID */
    ptr = strstr( response, "Set-Cookie: IW_INFO=" ) + 20;
    if( ptr != NULL ) strncpy( data, ptr, 32 );
    break;

case ACCESS_MODE:
    /* Decrypto */
    buff = strstr( response, "\r\n\r\n" ) + 4;
    len = atoi( strstr( response, "Content-length: " ) + 16 );
    ChangeEOF( buff, len );

    /* Get ErrNo */
    ptr = strstr( buff, "ERR_NO: " ) + 8;
    if( ptr != NULL ) strncpy( data, ptr, 2 );
    break;
}

void ChangeEOF( char *buff, int len )
{
    char ascii;
    int count;
    int index;

    index = 0;
    for( count = 0; count < len; count++ ){

        /* ASCII 4bit */
        ascii = CRYPTOKEY[index++] & 0x0F;
        if( index >= 54 ) index = 0;

        /* EOR */
        buff[count] ^= ascii;

        /* 0x7f */
        if( buff[count] == 0x7F ) buff[count] ^= ascii;
    }
}
```

9.5 サンプルソース (Java 言語)

以下は Java 言語によるサンプルソースです。

9.5.1 クライアントサンプル 1 (Sample1.java)

```
import java.io.*;
```

```
import java.net.Socket;
import java.net.SocketException;
import java.net.UnknownHostException;
import java.util.StringTokenizer;

/**
 * SampleA
 *
 */
public class SampleA {

    private final String CRYPTOKEY = "ygHxXW7Y+LiQDz7YUyIh4I9Ig28X6VaYAHMmFIXGEnJXEAXx7hGQgY";
    private final int LOGIN_MODE = 0;
    private final int ACCESS_MODE = 1;

    /** certd ホスト名 */
    private String i_host = "localhost";
    /** certd ポート番号 */
    private int i_port = 14142;
    /** ユーザ ID */
    private String i_uid = "user01";
    /** パスワード */
    private String i_pwd = "user01";

    /**
     * コンストラクタ。
     *
     */
    SampleA(String[] args) {

        // ログイン処理
        String loginRequest = createRequest(LOGIN_MODE, null);
        String response = socketCertd(loginRequest);
        String sessionId = checkResponse(LOGIN_MODE, response);

        // ログインエラー
        if(sessionId == null || sessionId.length() == 0)
            System.exit(-1);

        //Access
        String accessRequest = createRequest(ACCESS_MODE, sessionId);
        response = socketCertd(accessRequest);
        String errCode = checkResponse(ACCESS_MODE, response);
        try{
            System.exit( Integer.parseInt(errCode) );
        }catch(NumberFormatException e){
            System.exit(-1);
        }
    }

    /**
```

```
* certd に接続して、リクエストを送信し、結果を返す。
*
* @param request 送信リクエスト
* @return レスポンス。失敗の場合は null を返す
*/
private String socketCertd(String request) {

    String resData = null;
    try {
        Socket socket = new Socket(i_host, i_port);

        // メッセージ送信
        DataOutputStream out = new DataOutputStream(socket.getOutputStream());
        out.write(request.getBytes());
        out.flush();

        // メッセージ受信
        byte[] recvbuff = new byte[0];
        int buffsize = 0;
        byte[] workbuff = new byte[1024];

        DataInputStream in = new DataInputStream(socket.getInputStream());
        while(true) {
            int recvsiz = in.read(workbuff, 0, workbuff.length);
            if (recvsiz == -1) {
                break;
            }

            if (buffsize == 0) {
                recvbuff = new byte[recvsiz];
                System.arraycopy(workbuff, 0, recvbuff, 0, recvsiz);
                buffsize = recvsiz;
            } else {
                byte[] tempbuff = new byte[buffsize + recvsiz];
                System.arraycopy(recvbuff, 0, tempbuff, 0, buffsize);
                System.arraycopy(workbuff, 0, tempbuff, buffsize, recvsiz);
                recvbuff = tempbuff;
                buffsize = buffsize + recvsiz;
            }
        }
        resData = new String(recvbuff);

        out.close();
        in.close();
        socket.close();
    }
    catch( ArrayIndexOutOfBoundsException e ) {
        System.err.println("Usage:java MessageClient hostname");
        System.exit(-1);
    }
    catch( UnknownHostException e ) {
        System.err.println( "Host not found." );
    }
}
```

```
        System.exit(-1);
    }
    catch( SocketException e ) {
        System.err.println("Socket Error!" + e);
        System.exit(-1);
    }
    catch( IOException e ) {
        System.err.println("IO Error!");
        System.exit(-1);
    }
    return resData;
}

/**
 * リクエスト文字列を生成する。
 *
 * @param mode モード
 * @param sessionId セッション ID
 * @return リクエスト文字列
 */
private String createRequest(int mode, String sessionId) {
    String request = null;
    StringBuffer header = new StringBuffer();
    StringBuffer body = new StringBuffer();
    switch(mode){
        case LOGIN_MODE:
            body.append("MSG_ID: ReqLoginUID\r\n");
            body.append("IW_UID: " + i_uid + "\r\n");
            body.append("IW_PWD: " + i_pwd);
            header.append("REQ / ICP/2.0\r\n");
            header.append("Content-length: " + body.length() + "\r\n");
            header.append("\r\n");
            request = header.toString() + changeEOF(body.toString());
            break;
        case ACCESS_MODE:
            body.append("MSG_ID: ReqAccessUID\r\n");
            body.append("IW_INFO: " + sessionId + "\r\n");
            body.append("ACC_URL: http");
            header.append("REQ / ICP/2.0\r\n");
            header.append("Content-length: " + body.length() + "\r\n");
            header.append("\r\n");
            request = header.toString() + changeEOF(body.toString());
            break;
    }
    return request;
}

/**
 * レスポンスのチェックを行なう。
 *
 * @param mode モード
 * @param resData レスポンス
 */
```



```
* @return レスポンスを解析して取得した文字列
*/
private String checkResponse(int mode, String resData) {
    String result = null;
    if(resData == null || resData.length() == 0)
        return null;

    // ヘッダ部取出し
    int wkIdx = 0;
    String resLine = resData.substring(wkIdx, resData.indexOf("\r\n"));
    wkIdx += resLine.length() + 2;
    String dataLenLine = resData.substring(wkIdx, resData.indexOf("\r\n", wkIdx
));
    // ボディ取出し
    wkIdx += dataLenLine.length() + 4;
    String body = changeEOF(resData.substring(wkIdx));

    String key = null;
    switch(mode){
    case LOGIN_MODE:
        key = "IW_INFO: ";
        if(body.indexOf(key) >= 0){
            int idx = body.indexOf(key) + key.length();
            result = body.substring(idx, idx + 32);
        }
        break;
    case ACCESS_MODE:
        StringTokenizer st = new StringTokenizer(body, "\r\n");
        String line = null;
        key = "ERR_NO: ";
        for( ; st.hasMoreElements(); ) {
            line = st.nextToken();
            if(line.indexOf(key) >= 0){
                result = line.substring(line.indexOf(key) + key.length());
                break;
            }
        }
        break;
    default:
        return null;
    }

    return result;
}

/**
 * 文字列の EOF 変換を行う。
 *
 * @param base 変換元文字列
 * @return 変換後文字列
 */
private String changeEOF(String base) {
```

```
byte[] buff = base.getBytes();
int len = buff.length;
byte ascii;
int count;
int index;

index = 0;
for( count = 0; count < len; count++){

    /* ASCII 4bit */
    ascii = (byte)((int)CRYPTOKEY.charAt(index) & 0x0F);
    index++;
    if( index >= 54 ) index = 0;

    /* EOR */
    buff[count] ^= ascii;

    /* 0x7f */
    if( buff[count] == 0x7F ) buff[count] ^= ascii;

}
return new String(buff);
}

public static void main(String[] args) {
    new SampleA(args);
}
}
```

9.5.2 クライアントサンプル2 (Sample2.java)

```
import java.io.*;
import java.net.Socket;
import java.net.SocketException;
import java.net.UnknownHostException;
import java.util.StringTokenizer;

/**
 * SampleB
 *
 */
public class SampleB {

    private final String CRYPTOKEY = "ygHxXW7Y+LiQDz7YUyIh4I9Ig28X6VaYAHMmFIXGEnJXEAXx7hGQgY";
    private final int LOGIN_MODE = 0;
    private final int ACCESS_MODE = 1;

    /** dfw ホスト名 */
```

```
private String i_dfwHost = "localhost";
/** dfw ポート番号 */
private int i_dfwPort = 80;
/** certd ホスト名 */
private String i_certdHost = "localhost";
/** certd ポート番号 */
private int i_certdPort = 14142;
/** ユーザ ID */
private String i_uid = "user01";
/** パスワード */
private String i_pwd = "user01";

/**
 * コンストラクタ。
 *
 */
SampleB(String[] args) {

    if(args.length > 0)
        i_dfwHost = args[0];
    if(args.length > 1)
        i_dfwPort = Integer.parseInt(args[1]);
    if(args.length > 2)
        i_certdHost = args[2];
    if(args.length > 3)
        i_certdPort = Integer.parseInt(args[3]);

    // ログイン処理
    String loginRequest = createRequest(LOGIN_MODE, null);
    String response = socketMessage(i_dfwHost, i_dfwPort, loginRequest);
    String sessionId = checkResponse(LOGIN_MODE, response);

    // ログインエラー
    if(sessionId == null || sessionId.length() == 0)
        System.exit(-1);

    //Access
    String accessRequest = createRequest(ACCESS_MODE, sessionId);
    response = socketMessage(i_certdHost, i_certdPort, accessRequest);
    String errorCode = checkResponse(ACCESS_MODE, response);
    try{
        System.exit( Integer.parseInt(errorCode) );
    }catch(NumberFormatException e){
        System.exit(-1);
    }
}

/**
 * ソケット通信を行う。リクエストを送信し、結果を返す。
 *
 * @param host 接続先ホスト名
 * @param port 接続先ポート

```

```
* @param request 送信リクエスト
* @return レスポンス。失敗の場合は null を返す
*/
private String socketMessage(String host, int port, String request) {
    String resData = null;
    try {
        Socket socket = new Socket( host, port );

        // メッセージ送信
        DataOutputStream out = new DataOutputStream(socket.getOutputStream
());
        out.write(request.getBytes());
        out.flush();

        // メッセージ受信
        byte[] recvbuff = new byte[0];
        int buffsize = 0;
        byte[] workbuff = new byte[1024];

        DataInputStream in = new DataInputStream(socket.getInputStream());
        while(true) {
            int recvsize = in.read(workbuff, 0, workbuff.length);
            if (recvsize == -1) {
                break;
            }

            if (buffsize == 0) {
                recvbuff = new byte[recvsize];
                System.arraycopy(workbuff, 0, recvbuff, 0, recvsize);
                buffsize = recvsize;
            } else {
                byte[] tempbuff = new byte[buffsize + recvsize];
                System.arraycopy(recvbuff, 0, tempbuff, 0, buffsize);
                System.arraycopy(workbuff, 0, tempbuff, buffsize, recvsize);
                recvbuff = tempbuff;
                buffsize = buffsize + recvsize;
            }
        }
        resData = new String(recvbuff);

        out.close();
        in.close();
        socket.close();
    }
    catch( ArrayIndexOutOfBoundsException e ) {
        System.err.println("Usage:java MessageClient hostname");
        System.exit(-1);
    }
    catch( UnknownHostException e ) {
        System.err.println( "Host not found." );
        System.exit(-1);
    }
}
```

```
        catch( SocketException e ) {
            System.err.println("Socket Error!");
            System.exit(-1);
        }
        catch( IOException e ) {
            System.err.println("IO Error!");
            System.exit(-1);
        }
        return resData;
    }

    /**
     * リクエスト文字列を生成する。
     *
     * @param mode モード
     * @param sessionId セッション ID
     * @return リクエスト文字列
     */
    private String createRequest(int mode, String sessionId) {
        String request = null;
        StringBuffer header = new StringBuffer();
        StringBuffer body = new StringBuffer();
        switch(mode){
            case LOGIN_MODE:
                body.append("ACCOUNTUID=");
                body.append(i_uid);
                body.append("&PASSWORD=");
                body.append(i_pwd);

                body.append("&HIDEURL=%2Fhttp&LOGIN=ICEWALL_LOGIN&SUBMIT=+%91%97%90M+");
                header.append("POST /fw70sp2/dfw HTTP/1.0\r\n");
                header.append("Content-length: " + body.length() + "\r\n");
                header.append("Host: ");
                header.append(i_dfwHost);
                header.append(":");
                header.append(i_dfwPort);
                header.append("\r\n\r\n");
                request = header.toString() + body.toString();
                break;
            case ACCESS_MODE:
                body.append("MSG_ID: ReqAccessUID\r\n");
                body.append("IW_INFO: " + sessionId + "\r\n");
                body.append("ACC_URL: http");
                header.append("REQ / ICP/2.0\r\n");
                header.append("Content-length: " + body.length() + "\r\n");
                header.append("\r\n");
                request = header.toString() + changeEOF(body.toString());
                break;
        }
        return request;
    }
}
```

```
/**
 * レスポンスのチェックを行なう。
 *
 * @param mode モード
 * @param resData レスポンス
 * @return レスポンスを解析して取得した文字列
 */
private String checkResponse(int mode, String resData) {
    String result = null;
    if(resData == null || resData.length() == 0)
        return null;

    String key = null;
    switch(mode){
    case LOGIN_MODE:
        key = "Set-Cookie: IW_INFO=";
        if(resData.indexOf(key) >= 0){
            int idx = resData.indexOf(key)+key.length();
            result = resData.substring(idx, idx+32);
        }
        break;
    case ACCESS_MODE:
        // ヘッダ部取出し
        int wkIdx = 0;
        String resLine = resData.substring(wkIdx, resData.indexOf("\r\n"));
        wkIdx += resLine.length() + 2;
        String dataLenLine = resData.substring(wkIdx, resData.indexOf("\r\n", wkI
dx));
        // ボディ取出し
        wkIdx += dataLenLine.length() + 4;
        String body = changeEOF(resData.substring(wkIdx));
        StringTokenizer st = new StringTokenizer(body, "\r\n");
        String line = null;
        key = "ERR_NO: ";
        for( ; st.hasMoreElements(); ) {
            line = st.nextToken();
            if(line.indexOf(key) >= 0){
                result = line.substring(line.indexOf(key) + key.length());
                break;
            }
        }
        break;
    default:
        return null;
    }

    return result;
}

/**
 * 文字列の EOF 変換を行う。

```

```

*
* @param base 変換元文字列
* @return 変換後文字列
*/
private String changeEOF(String base) {
    byte[] buff = base.getBytes();
    int len = buff.length;
    byte ascii;
    int count;
    int index;

    index = 0;
    for( count = 0; count < len; count++){

        /* ASCII 4bit */
        ascii = (byte)((int)CRYPTOKEY.charAt(index) & 0x0F);
        index++;
        if( index >= 54 ) index = 0;

        /* EOR */
        buff[count] ^= ascii;

        /* 0x7f */
        if( buff[count] == 0x7F ) buff[count] ^= ascii;

    }
    return new String(buff);
}

public static void main(String[] args) {
    new SampleB(args);
}
}

```

9.6 サンプルソース（エージェントサンプル：mod_sample.c）

エージェントサンプルは Apache1.3 用の C 言語のみとなります。

```

#include "httpd.h"
#include "http_config.h"
#include "http_protocol.h"
#include "ap_config.h"

/* define */
#define CERT          "localhost"
#define PORT          14142
#define AGENT_KEY      "AGENT_KEY"
#define AGENT_PATH     "/sample"
#define COOKIENAME     "IW_INFO"
#define AGENT_KEY_EQ   "AGENT_KEY="

```

```

#define COOKIENAME_EQ "IW_INFO="
#define CRYPTOKEY      "ygHxXW7Y+LiQDz7YUyIh4I9Ig28X6VaYAHMmFIXG
EnJXEAXx7hGQgY"

module MODULE_VAR_EXPORT sample_module;

typedef struct{
    char *dfw_path;
} sample_dir_config;

/* prototype */
char *EncodeData( request_rec *r, char *old_buf );
void LoginRedirect( request_rec *r, sample_dir_config *conf, char *path, char *query );
void SetCookieRedirect( request_rec *r, char *query );
char *CreateRequest( request_rec *r, char *path, char *cookie );
int  GetErrNo( request_rec *r, char *res_body );
void SetHTTPHeader( request_rec *r, char *res_body, char *cookie );
void SocketCertd( int sendsize, char *request, char *response );
void ChangeEOF( char *buff, int len );

/* check access by host address */
/*-----*/
/*      メイン処理                                */
/*-----*/
/*      1. httpd.conf 設定値構造体のポインタを取得          */
/*      2. 各種情報取得                                */
/*      3. リクエスト判断                                */
/*      4. リクエストによりステータス等を変更し、サーバへ返す */
/*-----*/
static int SAMPLE_DFW_Main( r )
request_rec *r; /* Apache リクエスト構造体ポインタ */
{
    sample_dir_config *conf; /* httpd.conf 設定値構造体ポインタ */
    char *w_cookie;          /* Cookie */
    char *w_path;             /* リクエスト URL */
    char *w_query;            /* リクエスト QUERY_STRING */
    char *request;            /* ICP2.0 リクエストメッセージ */
    char *response;           /* ICP2.0 レスポンスメッセージ */
    char *res_body;           /* ICP2.0 レスポンス ボディ部 */
    int w_err_no;             /* ICP2.0 レスポンス err_no( 応答コード ) */
    int w_ret = DECLINED; /* 戻り値 */
    int sendsize;            /* 通信サイズ */

    /* 情報取得 */
    conf = ap_get_module_config( r->per_dir_config, &sample_module );
    w_cookie = (char *)ap_table_get( r->headers_in, "COOKIE" );
    w_path = r->uri;
    w_query = r->args;

    /* Cookie 有無判断 */

```



```
if( w_cookie == NULL ){
    /* リクエスト判断 */
    if( ( strcmp( w_path, AGENT_PATH, strlen( AGENT_PATH ) ) == 0 ) &&
        ( w_query != NULL && strcmp( w_query, AGENT_KEY_EQ, strlen(
AGENT_KEY_EQ ) ) == 0 ) ){
        /* Redirect for Set-Cookie */
        SetCookieRedirect( r, w_query );
    }else{
        /* Redirect for Login */
        LoginRedirect( r, conf, w_path, w_query );
    }
    w_ret = HTTP_MOVED_TEMPORARILY;
}else{
    /* IceWall の Cookie 有無判断 */
    if( ( w_cookie = strstr( w_cookie, COOKIENAME_EQ ) ) != NULL ){
        /* リクエスト判断 */
        if( ( strcmp( w_path, AGENT_PATH, strlen( AGENT_PATH ) ) == 0 ) &&
            ( w_query != NULL && strcmp( w_query, AGENT_KEY_EQ, strlen(
AGENT_KEY_EQ ) ) == 0 ) ){
            /* Redirect for Set-Cookie */
            SetCookieRedirect( r, w_query );
            w_ret = HTTP_MOVED_TEMPORARILY;
        }else{
            /* アクセス制御処理 */
            /* ICP2.0 アクセス制御リクエストメッセージ作成 */
            request = CreateRequest( r, w_path, w_cookie );

            /* アクセス制御通信 */
            sendsize = strlen( request );
            response = ap_palloc( r->pool, 4096 );
            memset( response, '\0', 4096 );
            SocketCertd( sendsize, request, response );

            /* ICP2.0 アクセス制御レスポンスメッセージ取得 */
            res_body = strstr( response, "\r\n\r\n" ) + 4;
            ChangeEOF( res_body, strlen( res_body ) );

            /* ICP2.0 アクセス制御レスポンスメッセージ判断 */
            w_err_no = GetErrNo( r, res_body );
            if( w_err_no != 0 ){
                if( w_err_no == 11 ){
                    /* 最ログイン要求 */
                    LoginRedirect( r, conf, w_path, w_query );
                    w_ret = HTTP_MOVED_TEMPORARILY;
                }else{
                    /* アクセス制御エラー */
                    w_ret = FORBIDDEN;
                }
            }else{
                /* 成功 */
                SetHTTPHeader( r, res_body, w_cookie );
                w_ret = DECLINED;
            }
        }
    }
}
```

```

    }
    }
    }else{
        if( ( strcmp( w_path, AGENT_PATH, strlen( AGENT_PATH ) ) == 0 ) &&
            ( w_query != NULL && strcmp( w_query, AGENT_KEY_EQ, strlen(
AGENT_KEY_EQ ) ) == 0 ) ){
            /* Redirect for Set-Cookie */
            SetCookieRedirect( r, w_query );
        }else{
            /* Redirect for Set-Cookie */
            LoginRedirect( r, conf, w_path, w_query );
        }
        w_ret = HTTP_MOVED_TEMPORARILY;
    }
}

return( w_ret );
}

/*-----*/
/* URL エンコード処理 */
/* */
/* 1. 引数 old_buf に渡されたバッファを一文字ずつ検索し、以下の文字の時に */
/* 1 6 進に変換する */
/* 「space " # % { } | \ ^ ~ [ ] ` ; ? : @ = &」 */
/* 2. 変換後の文字列を返す */
/*-----*/
char *EncodeData( r, old_buf )
request_rec *r; /* Apache リクエスト構造体ポインタ */
char *old_buf; /* エンコード前文字列 */
{
    char *w_ptr; /* エンコード前文字列 */
    char *w_new_ptr; /* エンコード後文字列 */
    char *w_en_chr; /* エンコード文字一時退避領域 */

    if( old_buf == NULL || strlen( old_buf ) == 0 ){
        return( old_buf );
    }

    w_new_ptr = ap_pstrdup( r->pool, "" );
    for ( w_ptr = old_buf; *w_ptr != '\0'; w_ptr++ ){
        switch( *w_ptr ){
            case ' ':
            case '"':
            case '#':
            case '%':
            case '{':
            case '}':
            case '|':
            case '\\':
            case '^':

```

```

        case '~':
        case '[':
        case ']':
        case '\\':
        case ';':
        case '?':
        case ':':
        case '@':
        case '=':
        case '&':
            w_en_chr = ap_psprintf( r->pool, "%%%02x", *w_ptr );
            w_new_ptr = ap_pstrcat( r->pool, w_new_ptr, w_en_chr, NULL );
            break;
        default:
            w_en_chr = ap_psprintf( r->pool, "%c", *w_ptr );
            w_new_ptr = ap_pstrcat( r->pool, w_new_ptr, w_en_chr, NULL );
            break;
    }
}

return( w_new_ptr );
}

/*-----*/
/* ログインリダイレクト処理 */
/* */
/* 1. HTTP_HOST 値を取得 */
/* 2. QUERY_STRING 部分へ設定する各パラメータを URL エンコード */
/* 3. DFW へのリダイレクト用の Location ヘッダを作成 */
/* 4. Apache のリクエスト構造体へ Location ヘッダをセットする */
/*-----*/
void LoginRedirect( r, conf, path, query )
request_rec *r; /* Apache リクエスト構造体ポインタ */
sample_dir_config *conf; /* httpd.conf 設定値構造体ポインタ */
char *path; /* &path= 部分 */
char *query; /* &query= 部分 */
{
    char *w_host; /* HTTP_HOST 値 */
    char *w_content_url; /* コンテンツ URL */
    char *w_en_content_url; /* エンコード済みコンテンツ URL */
    char *w_en_path; /* エンコード済み &path= 部分 */
    char *w_en_content_query; /* エンコード済み &query= 部分 */
    char *w_redirect; /* Location ヘッダ設定文字列 */

    /* Redirect for Login */
    w_host = (char *)ap_table_get( r->headers_in, "HOST" );
    w_content_url = ap_psprintf( r->pool,
                                "%s://%s%s",
                                ap_http_method( r ),
                                w_host,
                                AGENT_PATH );

```

```

/* Encode */
w_en_content_url = EncodeData( r, w_content_url );
w_en_path = EncodeData( r, path );

/* Location Header */
if( query == NULL || strlen( query ) == 0 ){
    w_redirect = ap_psprintf( r->pool,
        "%s?%s=%s&path=%s&query=",
        conf->dfw_path,
        AGENT_KEY,
        w_en_content_url,
        w_en_path );
}
else{
    w_en_content_query = EncodeData( r, query );
    w_redirect = ap_psprintf( r->pool,
        "%s?%s=%s&path=%s&query=%s",
        conf->dfw_path,
        AGENT_KEY,
        w_en_content_url,
        w_en_path,
        w_en_content_query );
}
ap_table_set( r->err_headers_out, "Location", w_redirect );

return;
}

/*-----*/
/*      Set-Cookie リダイレクト処理                                */
/*-----*/
/*      1. HTTP_HOST 値を取得                                    */
/*      2. QUERY_STRING 部分から取得した情報を URL デコード      */
/*      3. Set-Cookie リダイレクト用の Location ヘッダを作成      */
/*      4. Apache のリクエスト構造体へ Location ヘッダをセットする */
/*      5. セッション情報を取得                                    */
/*      6. セッション情報を元に Set-Cookie ヘッダを作成          */
/*      7. Apache のリクエスト構造体へ Set-Cookie ヘッダをセットする */
/*-----*/
void SetCookieRedirect( r, query )
request_rec *r; /* Apache リクエスト構造体ポインタ */
char *query; /* QUERY_STRING */
{
    char *w_host; /* HTTP_HOST 値 */
    char *w_path; /* &path= 部分 */
    char *w_content_url; /* コンテンツ URL */
    char *w_content_query; /* コンテンツ QUERY_STRING */
    char *w_redirect; /* Location ヘッダ設定文字列 */
    char *w_session; /* セッション情報 */
    char *w_query; /* QUERY_STRING */
    char *w_setcookie; /* Set-Cookie ヘッダ設定文字列 */

```

```

/* Redirect for Set-Cookie */
/* host */
w_host = (char *)ap_table_get( r->headers_in, "HOST" );
/* path */
w_path = strstr( query, "path=" );
if( w_path != NULL ){
    w_path = w_path + 5;
    w_path = ap_getword( r->pool, (const char **)&w_path, '&' );
    ap_unescape_url( w_path );
}

/* URL */
w_content_url = ap_psprintf( r->pool,
                             "%s://%s%s",
                             ap_http_method( r ),
                             w_host,
                             w_path );

/* query */
w_content_query = strstr( query, "query=" );
ap_unescape_url( w_content_query );

/* Location Header */
if( w_content_query != NULL && strlen( w_content_query ) > 6 ){
    w_content_query = w_content_query + 6;
    w_redirect = ap_psprintf( r->pool,
                             "%s?%s",
                             w_content_url,
                             w_content_query );
}else{
    w_redirect = ap_psprintf( r->pool,
                             "%s",
                             w_content_url );
}
ap_table_set( r->err_headers_out, "Location", w_redirect );

/* SessionID */
w_query = ap_pstrdup( r->pool, query + strlen( AGENT_KEY_EQ ) );
w_session = ap_getword( r->pool, (const char **)&w_query, '&' );

/* Set-Cookie Header */
w_setcookie = ap_psprintf( r->pool, "%s=%s; path=/", COOKIENAME, w_session );
ap_table_set( r->err_headers_out, "Set-Cookie", w_setcookie );

return;
}

/*-----*/
/*   アクセス制御リクエストメッセージ作成処理   */
/*   */
/*   1. HTTP_HOST 値を取得   */
/*   2. アクセス制御を行う URL を作成   */

```

```

/* 3. セッション情報を取得 */
/* 4. リクエストメッセージのボディ部を作成し、暗号化する */
/* 5. リクエストメッセージのヘッダ部を作成する */
/* 6. ヘッダ部とボディ部を結合してリクエストメッセージを作成する */
/* 7. リクエストメッセージ（ボディ部暗号化済み）を返す */
/*-----*/
char *CreateRequest( r, path, cookie )
request_rec *r; /* Apache リクエスト構造体ポインタ */
char *path; /* アクセス制御対象 PATH */
char *cookie; /* IceWall の Cookie 情報 */
{
    char *w_host; /* HTTP_HOST 値 */
    char *w_content_url; /* コンテンツ URL */
    char *w_session; /* セッション情報 */
    char *w_request; /* ICP2.0 アクセス制御リクエストメッセージ */
    char *w_req_header; /* ICP2.0 アクセス制御リクエスト ヘッダ部 */
    char *w_req_body; /* ICP2.0 アクセス制御リクエスト ボディ部 */

    /* Access Control */
    /* host */
    w_host = (char *)ap_table_get( r->headers_in, "HOST" );
    /* url */
    w_content_url = ap_psprintf( r->pool,
                                "%s://%s%s",
                                ap_http_method( r ),
                                w_host,
                                path );

    /* SessionID */
    w_session = cookie + strlen( COOKIENAME_EQ );

    /* Request Message */
    w_req_body = ap_psprintf( r->pool,
                              "MSG_ID: ReqAccessUID\r\nIW_INFO: %s\r\nACC_URL: %s",
                              w_session,
                              w_content_url );
    ChangeEOF( w_req_body, strlen( w_req_body ) );
    w_req_header = ap_psprintf( r->pool,
                                "REQ / ICP/2.0\r\nContent-length: %d\r\n\r\n",
                                strlen( w_req_body ) );
    w_request = ap_pstrcat( r->pool, w_req_header, w_req_body, NULL );

    return( w_request );
}

/*-----*/
/* アクセス制御レスポンス ERR_NO 取得処理 */
/*-----*/
/* 1. レスポンスのボディ部から、ERR_NO の値を取得（スペースは無視） */
/* 2. 取得した ERR_NO を数値化し、返す */
/*-----*/

```

```

int GetErrNo( r, res_body )
request_rec *r; /* Apache リクエスト構造体ポインタ */
char *res_body; /* ICP2.0 レスポンス 復号化済みボディ部 */
{
    char *w_err_no; /* ICP2.0 レスポンス err_no( 応答コード ) */
    char *w_ptr; /* 作業用ポインタ */
    char *w_ptr_chr; /* 作業用文字退避領域 */

    /* err_no */
    w_err_no = ap_pstrdup( r->pool, "" );
    w_ptr = ap_pstrdup( r->pool, strstr( res_body, "ERR_NO:" ) + 7 );
    while( *w_ptr != '\0' && *w_ptr != '\r' ){
        if( *w_ptr == ' ' ){
            w_ptr++;
            continue;
        }
        w_ptr_chr = ap_psprintf( r->pool, "%c", *w_ptr );
        w_err_no = ap_pstrcat( r->pool, w_err_no, w_ptr_chr, NULL );
        w_ptr++;
    }
    return( atoi( w_err_no ) );
}

/*-----*/
/* HTTP ヘッダ設定処理 */
/*
/* 1. ユーザ ID を取得 */
/* 2. セッション情報を取得 */
/* 3. HTTP_UID ヘッダを設定 */
/* 4. HTTP_SESSION ヘッダを設定 */
/*-----*/
void SetHttpHeader( r, res_body, cookie )
request_rec *r; /* Apache リクエスト構造体ポインタ */
char *res_body; /* ICP2.0 レスポンス 復号化済みボディ部 */
char *cookie; /* IceWall の Cookie 情報 */
{
    char *w_uid; /* HTTP_UID 値設定文字列 */
    char *w_ptr; /* 作業用ポインタ */
    char *w_ptr_chr; /* 作業用文字退避領域 */
    char *w_session; /* HTTP_SESSION 値設定文字列 */

    /* UID */
    w_uid = ap_pstrdup( r->pool, "" );
    w_ptr = ap_pstrdup( r->pool, strstr( res_body, "IW_UID:" ) + 7 );
    while( *w_ptr != '\0' && *w_ptr != '\r' ){
        if( *w_ptr == ' ' ){
            w_ptr++;
            continue;
        }
        w_ptr_chr = ap_psprintf( r->pool, "%c", *w_ptr );
        w_uid = ap_pstrcat( r->pool, w_uid, w_ptr_chr, NULL );

```

```

    w_ptr++;
}
/* SessionID */
w_session = cookie + strlen( COOKIENAME_EQ );

/* Set Header */
ap_table_set( r->headers_in, "Uid", w_uid );
ap_table_set( r->headers_in, "SESSION", w_session );

return;
}

/*-----*/
/*   Certd ソケット通信処理                               */
/*                               */
/*   1. ソケット作成                               */
/*   2. Certd 接続                               */
/*   3. リクエストメッセージ送信                               */
/*   4. レスポンスメッセージ取得                               */
/*   5. ソケットクローズ                               */
/*                               */
/*-----*/
void SocketCertd( sendsize, request, response )
int sendsize; /* 通信サイズ */
char *request; /* ICP2.0 リクエストメッセージ */
char *response; /* ICP2.0 レスポンスメッセージ */
{
    int sock; /* ソケット */
    struct sockaddr_in addr; /* ソケット情報 */
    struct hostent *hent; /* ホスト情報 */

    /* Certd ( localhost:14142 ) */
    memset( &addr, 0x00, sizeof(struct sockaddr_in) );
    addr.sin_family = AF_INET;
    addr.sin_port = htons( PORT );
    hent = gethostbyname( CERT );
    memcpy( &addr.sin_addr, hent->h_addr, hent->h_length );

    sock = socket( AF_INET, SOCK_STREAM, 0 );

    connect( sock, (struct sockaddr*)&addr, sizeof( addr ) );

    send( sock, request, sendsize, 0 );

    recv( sock, response, 4096, 0 );

    shutdown( sock, SHUT_RDWR );
    close( sock );
}

/*-----*/

```



```

/* 暗号化 / 復号化处理 */
/* */
/* 1. EOF 暗号化 / 復号化 */
/* */
/*-----*/
void ChangeEOF( buff, len )
char *buff; /* 暗号化 ( 復号化 ) 対象文字列 */
int len; /* 文字列長 */
{
    char ascii; /* ASCII 下位 4 ビット */
    int count; /* カウンタ */
    int index; /* インデックス */

    index = 0;
    for( count = 0; count < len; count++){

        /* ASCII 4bit */
        ascii = CRYPTOKEY[index++] & 0x0F;
        if( index >= 54 ) index = 0;

        /* EOR */
        buff[count] ^= ascii;

        /* 0x7f */
        if( buff[count] == 0x7F ) buff[count] ^= ascii;
    }
}

/* create per-dir config structures */
/*-----*/
/* httpd.conf 設定値構造体初期化处理 */
/* */
/* 1. 構造体領域確保 */
/* */
/* 2. 初期値設定 */
/* */
/*-----*/
static void *create_per_dir_config( p, arg )
pool *p; /* プール */
char *arg; /* 引数 */
{
    sample_dir_config *conf; /* httpd.conf 設定値構造体 */

    conf=ap_palloc( p,sizeof( sample_dir_config ) );
    conf->dfw_path = ap_pstrdup( p, "" );
    return( (void *)conf );
}

/*-----*/
/* httpd.conf 設定値取得処理 */
/* */
/* 1. httpd.conf の DFW_PATH に設定した値を、conf->dfw_path へ設定 */

```

```

/*
/*-----*/
static const char *set_dfw_path( cmd , dummy , arg )
cmd_parms *cmd; /* コマンドパラメータ */
void *dummy; /* ダミー void ポインタ */
char *arg; /* 引数 */
{
    sample_dir_config *conf; /* httpd.conf 設定値構造体 */

    conf = dummy;
    conf->dfw_path = ap_pstrdup( cmd->pool, arg );

    return( NULL );
}

/* table of config file commands */
static const command_rec cmds[]=
{
    { "DFW_PATH", set_dfw_path, NULL, OR_ALL, TAKE1, "DFW_PATH" },
    { NULL }
};

/* Dispatch list for API hooks */
module MODULE_VAR_EXPORT sample_module = {
    STANDARD_MODULE_STUFF,
    NULL, /* module initializer */
    create_per_dir_config, /* create per-dir config structures */
    NULL, /* merge per-dir config structures */
    NULL, /* create per-server config structures */
    NULL, /* merge per-server config structures */
    cmds, /* table of config file commands */
    NULL, /* [#8] MIME-typed-dispatched handlers */
    NULL, /* [#1] URI to filename translation */
    NULL, /* [#4] validate user id from request */
    NULL, /* [#5] check if the user is ok _here_ */
    SAMPLE_DFW_Main, /* [#3] check access by host address */
    NULL, /* [#6] determine MIME type */
    NULL, /* [#7] pre-run fixups */
    NULL, /* [#9] log a transaction */
    NULL, /* [#2] header parser */
    NULL, /* child_init */
    NULL, /* child_exit */
    NULL /* [#0] post read-request */
#ifdef EAPI
    ,NULL, /* EAPI: add_module */
    NULL, /* EAPI: remove_module */
    NULL, /* EAPI: rewrite_command */
    NULL /* EAPI: new_connection */
#endif
};

```

10 通信メッセージ暗号化ライブラリサンプル

フォワーダーおよび認証モジュールをインストールすることで、標準でインストールされる通信メッセージ暗号化ライブラリ開発キットの内容です。

フォワーダーの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です。

ディレクトリ	内容
/opt/icewall-ss0/developkit/dfw/DfwCryptoExit	Makefile
	iwcrypto.c
	iwcrypto.h
	libcpt.a

認証モジュールの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です。

ディレクトリ	内容
/opt/icewall-ss0/developkit/certd/CryptoExit	Makefile
	iwcrypto.c
	iwcrypto.h
	libcpt.a

10.1 サンプルソースファイル (iwcrypto.c)

サンプルソースファイルは、フォワーダーと認証モジュールで共通です。

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

#include "iwcrypto.h"

#define CRYPTOKEY "ygHxXW7Y+LiQDz7YUyIh4I9Ig28X6VaYAHMmFIXGEnJ
XEAXx7hGQgY"

void EXCPT_EOF( char *buff, int buflen );

IW_EXCRYPTO *IW_ExEncrypto( char *buff, int buflen )
{
    IW_EXCRYPTO *excrypto;

    excrypto = IW_ExCPTCreateCrypto( buff, buflen, 0, NULL );

    EXCPT_EOF( excrypto->buff, excrypto->buflen );

    return( excrypto );
}

IW_EXCRYPTO *IW_ExDecrypto( char *buff, int buflen )
```

```

{
    IW_EXCRYPTO *excrypto;

    excrypto = IW_ExCPTCreateCrypto( buff, buflen, 0, NULL );

    EXCPT_EOF( excrypto->buff, excrypto->buflen );

    return( excrypto );
}

void EXCPT_EOF( char *buff, int buflen )
{
    char ascii;
    int count;
    int index;

    index = 0;
    for( count = 0; count < buflen; count++ ){

        /* ASCII 4bit */
        ascii = CRYPTOKEY[index++] & 0x0F;
        if( index >= 54 ) index = 0;

        /* EOR */
        buff[count] ^= ascii;

        /* 0x7f */
        if( buff[count] == 0x7F ) buff[count] ^= ascii;
    }
}

```

※ サンプルの暗号化処理では暗号化後の文字列にヌル (0x00) が含まれる場合があります。そのような場合にはヌルを意識しないコピー関数に変更してください。

10.2 ヘッダーファイル (iwcrypto.h)

ヘッダーファイルは、フォワーダーと認証モジュールで共通です。

```

#ifndef IWCRYPTO_H
#define IWCRYPTO_H

typedef struct _IW_EXCRYPTO {
    char *buff;
    int buflen;
    int result;
    char *errmsg;
} IW_EXCRYPTO;

IW_EXCRYPTO *IW_ExEncrypt( char *buff, int buflen );
IW_EXCRYPTO *IW_ExDecrypt( char *buff, int buflen );

```

```
IW_EXCRYPTO *IW_ExCPTCreateCrypto( char *buff, int buflen,  
                                   int result, char *errmsg );  
int IW_ExCPTReleaseCrypto( IW_EXCRYPTO *excrypto );  
  
#endif /* #ifndef IWCRYPTO_H */
```

10.3 フォワーダー用メイクファイル (HP-UX 版 : Makefile) **10.0**

```
CC                = cc  
  
CCFLAGS           = -D_UNIX -D_HPUX_SOURCE -D_POSIX_C_SOURCE=19  
9506L -Aa +e -D_FILE_OFFSET_BITS=64 -z +DD64 +z  
  
LIBS              = ./libcpt.a  
  
INCLUDE           = -I./  
  
MAKEFILE          = Makefile  
  
OBJS              = iwcrypto.o  
  
PROGRAM           = DfwCryptoExit  
  
SRCS              = iwcrypto.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
```

10.4 フォワーダー用メイクファイル (Linux 版 : Makefile) **10.0**

```
CC                = gcc  
  
CCFLAGS           = -m64 -fPIC -Dlinux -D_FILE_OFFSET_BITS=64 -D_LAR  
GEFILE_SOURCE -D_GNU_SOURCE  
  
LDFLAGS           = -m64 -fPIC  
  
LIBS              = ./libcpt.a  
  
INCLUDE           = -I./
```

MAKEFILE	= Makefile
OBJS	= iwcrypto.o
PROGRAM	= DfwCryptoExit
SRCS	= iwcrypto.c
all:	\$(PROGRAM)
\$(PROGRAM):	\$(OBJS)
\$(LIBS)	gcc -shared \$(LDFLAGS) -o lib\$(PROGRAM).sl \$(OBJS)
.c.o:	\$(CC) \$(CCFLAGS) \$(INCLUDE) -c \$(SRCS)
clean:	rm -f \$(OBJS) lib\$(PROGRAM).sl core

10.5 認証モジュール用メイクファイル (HP-UX 版 : 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	= ./libcpt.a
INCLUDE	= -I./
MAKEFILE	= Makefile
OBJS	= iwcrypto.o
PROGRAM	= CryptoExit
SRCS	= iwcrypto.c
all:	\$(PROGRAM)
\$(PROGRAM):	\$(OBJS)
.c.o:	ld -b -z -o lib\$(PROGRAM).sl \$(OBJS) \$(LIBS)
clean:	\$(CC) \$(CCFLAGS) \$(INCLUDE) -c \$(SRCS)
	rm -f \$(OBJS) lib\$(PROGRAM).sl core

10.6 認証モジュール用メイクファイル (Linux 版 : Makefile)

```
CC                = gcc
CCFLAGS           = -m64 -fPIC -DLinux -D_FILE_OFFSET_BITS=64 -D_LARGEFILE_SOURCE -D_GNU_SOURCE
LDFLAGS           = -m64 -fPIC
LIBS              = ./libcpt.a
INCLUDE           = -I./
MAKEFILE          = Makefile
OBJS              = iwcrypto.o
PROGRAM           = CryptoExit
SRCS              = iwcrypto.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
```