Biz Browser Mobile

追補マニュアル

1. はじめに

Biz/Browser Mobile は PC 版 Biz/Browser V3.0 をベースにモバイル端末に必要な機能を抽出し、かつ一部 PC 版の最新版である Biz/BrowserXE の機能を取り込んでいます。

このマニュアルでは Biz/Browser V3.0 のマニュアルに対し Biz/Browser Mobile での機能制限や動作変更、機能拡張されている部分に関し追補するものです。
このマニュアルと併せて PC 版のマニュアルもご参照くださるようお願いいたします。

2. セットアップ

Biz/Browser Mobile のセットアップ方法について解説いたします。
WindoswsCE.NET，WindowsMobile5 の操作を理解していることが前提になります。

2.1 ActiveSync を利用して PC からファイルをコピーする方法

1. PC と端末を ActiveSync で接続します。
2. PC 側ののエクスプローラを使って BizBrowserMobile.xx.cab(xx は機種により異なります)
3. コピーした BizBrowserMobile.xx.cab を端末側のエクスプローラでタップし実行します
4. インストーラが起動しますので指示に従いインストールを行います。デフォルトでは¥Program Files¥BizBrowserMobile へインストールを行います
5. スタートメニューのプログラム中に BizBrowserMobile のアイコンが表示されていることを確認してください。

2.2 ネットワークから Web ブラウザにてダウンロードする方法

1. BizBrowserMobile.xx.cab ファイル(xx は機種により異なります)を端末よりアクセス可能な Web サイトへ配置します
2. 端末の Web ブラウザを使って、サイトから cab ファイルを端末へダウンロードします
3. ダウンロードした BizBrowserMobile.xx.cab を端末側のエクスプローラでタップし実行します
4. インストーラが起動しますので指示に従いインストールを行います。デフォルトでは¥Program Files¥BizBrowserMobile へインストールを行います
5. スタートメニューのプログラム中に BizBrowserMobile のアイコンが表示されていることを確認してください。

備考

※ 初回実行時、自動的にインストールフォルダ以下にフォルダが作成されます。これらはキャッシュの保存などに使用されます。

※ 既に Biz/Browser Mobile がインストールされている場合上書きインストールされます(旧バージョンである Biz/Browser for PDA は上書きされません)。

3. 起動方法

アプリケーションを起動する方法は、下記の方法があります。

3.1 スタートメニューから起動

[スタートメニュー]-[プログラム]から[BizBrowserMobile]を選択します

3.2 デスクトップにショートカットを登録して、アイコンを指定する方法

特定ユーザーおよび定型業務の起動に適しています

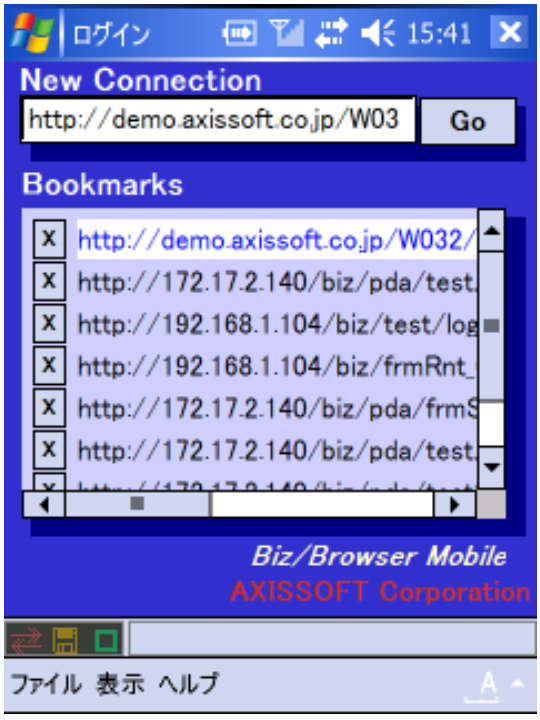
1. BizBrowserMobile.exe のショートカットをデスクトップ(または¥Windows¥スタートメニュー)に作成する。
2. ショートカットの引数にアプリケーションの URL を指定する。

※ ショートカットはプロパティにて設定します。

例)"¥Program Files¥BizBrowserMobile¥BizBrowserMobile.exe" "http://myapp.myserver/login.crs"

3.3 ログイン画面で URL を入力または履歴リストから選択する方法

プログラム引数なしに起動した時は、ログイン画面が表示されます。



New Connection ボックスに URL を入力し、[Go]ボタンでアプリケーションを起動します。

一度記入した URL は、次回起動時に一覧で表示されます。この一覧の URL をタップすると New Connection ボックスに URL がコピーされます。
また一覧をダブルタップすると直接 URL へアクセスを行います。

URL 一覧を削除したい場合は URL 左の[X]をタップすると削除確認ダイアログが表示され、OK すると一覧から削除されます。

4. 前バージョンからの主な変更点

Biz/Browser Moble は前バージョンである Biz/Browser for PDA V1.1 に対し PC 版 Biz/Browser よりフィードバックされた大幅なバグフィックスを施し、さらに一部機能拡張が行われています。

この章では主な変更点について解説いたします、また併せて「機能差異表.pdf」もご参照ください。

4.1 接続ライセンス管理を導入

- PC 版 Biz/Browser と同様の接続ライセンスを導入しました。

接続先 URL に対しライセンスの導入が必要になります。詳しくは 5 章にて解説いたします。

4.2 画面の縦横画面動的変更に対応

- 動的に画面を縦横に切り替える操作に対応しました。

4.3 コマンドバー、ステータスバーの表示切替機能

- 追加メソッドによりコマンドバーおよびステータスバーの表示・非表示の切り替えに対応しました。

4.4 エディット系オブジェクトの操作性の改善

- 前バージョンで使用できなかった AutoTab、InputMode などのサポートによりキー入力重視の操作性が実現可能になりました
- ペンのタップ＆ホールドによるコンテキストメニューの表示および編集に対応しました

4.5 NumberEdit、DateEdit の改善

- Biz/Browser XE 互換のプロパティ、メソッドを追加しました

4.6 フォーカス制御命令の拡張

- Biz/Browser XE 互換のフォーカス処理(SkipTabFocus、AutoTabFocus プロパティ FocusOperation イベント、MoveFocus メソッド)を追加しました

4.7 セキュリティ機能の搭載

- Biz/Browser XE 互換のセキュリティパッケージを搭載
  1. 暗号化ファイルによりローカルファイルのセキュア化が可能
  2. ハッシュによりファイルの改編チェックなどが可能
- ユニーク ID を作成する GetUUID 関数を実装

4.8 一部使用不可だった関数、メソッドの復活

- round 系関数、cachedate 等 V1.1 で使用できなかった関数が使用可能になりました

4.9 フォントおよび描画ロジックの改善

描画方法の改良、およびアンチエイリアスフォントの採用(一部機種)しました

5. 接続ライセンス

Biz/Browser Mobile より新しく接続単位のライセンス管理に対応しました。
従来は、インストール単位にライセンスが必要となっていたため、Biz/Browser をASP サービスで利用する場合や、Biz/Browser 上に構築されたパッケージ製品のデモなどに対応する事ができませんでした。

新しいライセンス管理では、Biz/Browser とWEB サーバとの接続単位にライセンスを確認する事が可能となったため、デモや試用などの一時的な利用や、特定のサーバへの接続だけに限定した利用形態に柔軟に対応する事ができるようになりました。

5.1 接続ライセンスの仕組み

ここでは、接続ライセンスの技術的な仕組みと利用方法について説明します。

5.1.1 接続ライセンスの有無で制限される機能

Biz/Browser には、多くの通信機能が搭載されていますが、そのうち GET メソッドを利用した http または https 通信だけが接続ライセンスの有無により制限されます。
その他の Login メソッド、httpSession クラスの GET、POST や import 命令などは、接続ライセンスの有無に関わらず従来どおり実行する事ができます。

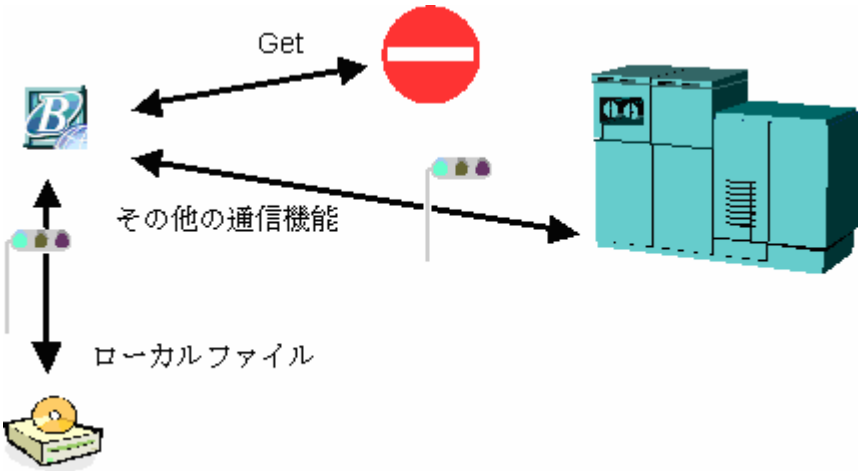
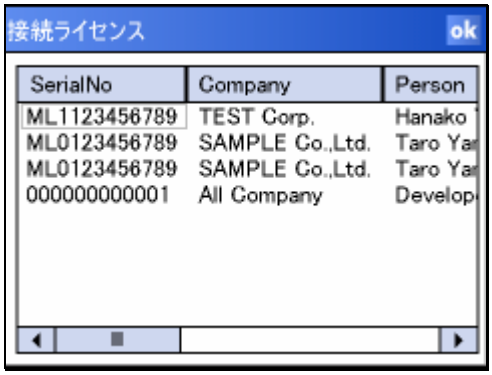


Fig.1 接続ライセンスがない場合

接続ライセンスには、有効期限、許可された接続先サーバの URL などが記載されています。つまり、接続ライセンスは接続先のサーバ単位に必要となります。サーバ A に対する接続ライセンスではサーバ B に対して GET を利用する事はできません。また、有効期限が明示されている場合（無期限の場合もあります）、示された期日以降はその接続ライセンスは無効となります。

5.1.2 接続ライセンスの確認

現在有効な接続ライセンスは、メニューの「表示」→「接続ライセンスの確認」で表示されるダイアログで確認する事ができます。



また、CRS プログラム上では、新しいグローバル関数 HasConnectionLicense により確認する事ができます。

5.1.3 接続ライセンスのロード

接続ライセンスは、接続ライセンス証明書をインストールすることにより Biz/Browser に認識させる事ができます。接続ライセンス証明書は、次のような形式の XML ファイルです。

```
<?xml version="1.0" encoding="utf-8" ?>
<license xmlns="http://www.axissoft.co.jp/CRS/2004/license">
  <serial_no>ML0123456789</serial_no>
  <product>Biz/Browser Mobile ver2.0</product>
  <company>SAMPLE Co.,Ltd.</company>
  <person>Taro Yamada</person>
  <usage>public</usage>
  <date>Wed, 11 Oct 2006 11:31:24 +0900</date>
  <expire>Sun, 31 Dec 2006 00:00:00 +0900</expire>
  <url>http://app.sample.co.jp</url>
  <url>http://app2.sample.co.jp</url>
  <signature algorithm="3DES">jN51jyoFIC9ZypEV0s2M2i+CGSFgh4R
</license>
```

通常、このファイルは Biz/Browser の購入時に同梱されます。

接続ライセンス証明書は、Biz/Browser のインストールディレクトリ内の、settings.v2¥license ディレクトリに置くと、Biz/Browser の起動時にロードされます。

また、CRS プログラム上では、新しいグローバル関数 ImportConnectionLicense により指定した URL からダウンロードしてロードする事ができます。ImportConnectionLicense は、指定された URL から接続ライセンス証明書をダウンロードし、記載内容を確認します。もし、記載内容に問題がなければ上記のディレクトリに格納し、指定された接続先への接続を許可します。

接続ライセンスを使用した例

menu.crs

```
/* 接続ライセンスの確認とインポート */
if( sys.CLIENT == “Biz/Browser Mobile”  && sys.CLIENT_VERSION >= 2.0 ) {
    if( !HasConnectionLicense("http://server") ) {
        try {
            importConnectionLicense("http://server/license.xml");
        }
        catch(exp) {
            MessageBox(exp);
            login();
        }
    }
}

/* キャッシュのバージョン確認 */
string curVersion="2.0.0";
GET("version.crs");
if( curVersion != version ) {
    deleteCache();
    GET("version.crs");
}
GET("app_main.crs"); /* アプリケーションのメインプログラム */
```

5.2 接続ライセンス証明書

接続ライセンス証明書は、次のような形式の XML ファイルです

```
<?xml version="1.0" encoding="utf-8" ?>
<license xmlns="http://www.axissoft.co.jp/CRS/2004/license">
  <serial_no>ML0123456789</serial_no>
  <product>Biz/Browser Mobile ver2.0</product>
  <company>SAMPLE Co.,Ltd.</company>
  <person>Taro Yamada</person>
  <usage>public</usage>
  <date>Wed, 11 Oct 2006 11:31:24 +0900</date>
  <expire>Sun, 31 Dec 2006 00:00:00 +0900</expire>
  <url>http://app.sample.co.jp</url>
  <url>http://app2.sample.co.jp</url>
  <signature algorithm="3DES">jN51jyoFIC9ZypEV0s2M2i+CGSFgh4R
</license>
```

接続ライセンス証明書の記載内容

接続ライセンス証明書には以下の情報が記載されています。

serial_no
シリアル番号で、個々の接続ライセンス証明書に対してユニークな番号が振られます。

product
ライセンスする製品名とバージョンです。

company
ライセンスを受けた会社名です。

person
ライセンスを受けた担当者名です。

usage
利用目的です。public と private が定義されています。

date
この接続ライセンスを発行した日時です。

expire
この接続ライセンスが無効となる満了日です。無期限のライセンスの場合、このフィールドはありません。

url
ライセンスされた接続先の URL です。NetObject.Get メソッドで通信をを行う際、この URL のプロトコル、ホスト名、ポート番号と一致する必要があります。ホスト名は、ドメインの有無や IP アドレス表記など同一の接続先を示すさまざまな表記方法がありますが、このフィールドと一致する表記の場合だけ有効となります。
このフィールドには、接続先無制限として*(アスタリスク)が記載される事があります。またこのフィールドは、複数併記されることがあります。

signature
デジタル署名です。

5.3 接続ライセンス証明書の取り扱いについて

接続ライセンス証明書および、その記載内容は、Biz/Browser の表示メニューからいつでも確認する事ができますが、インターネットやその他の通信経路を通じて、Biz/Browser から外部に送出されることはありません。

5.4 接続ライセンス証明書の入手方法について

通常、接続ライセンス証明書は Biz/Browser の購入時に同梱されます。入手方法の詳細は、アクシスソフト(株)の営業窓口にお問い合わせください。

6. 前バージョンからの移行

Biz/Browser Mobile においても基本的に前バージョンである Biz/Browser for PDA V1.1 用に開発された CRS をそのまま動作させることが可能です。

しかしながら一部動作仕様の変更や、バグフィックスによる動作の違いがありアプリケーションの動作が V1.1 の挙動に依存している場合問題が発生する場合があります。

この章では Biz/Browser Mobile への移行に際し問題になりそうなポイントについて解説いたします。

6.1 配置フォルダの変更

V1.1 では設定ファイルを¥biz 以下に保存するように固定されていました。今バージョンよりは exe のあるフォルダを起点とし設定ファイル等を保存するように変更されています。

配置フォルダに依存したプログラミングを行っている場合修正を行う必要があります。

以下に主なフォルダ配置を示します

	Biz/Browser for PDA V1.1	Biz/Browser Mobile
exe ファイル	任意の場所(¥biz、ルート)	任意の場所(インストーラデフォルト ¥Program Files¥BizBrowserMobile)
設定ファイルフォルダ	¥biz¥defaultSettings 固定	[exe のあるフォルダ]¥settings.v2
ユーザーファイルフォルダ	¥biz¥users¥default 固定 または ¥Windows¥Profiles¥guest¥Application Data¥AXIS Soft¥BizBrowser(WindowsMobile 版)	[exe のあるフォルダ]¥users または 同左(WindowsMobile 版)
キャッシュ格納フォルダ	[ユーザーファイルフォルダ]¥cache	[ユーザーファイルフォルダ]¥cache.v2
拡張 DLL フォルダ	[exe のあるフォルダ]¥dll	同左

6.2 UserAgent の変更

Web サーバへのリクエスト時の UserAgent が変更されました。V1.1 では PC 版と区別がありませんでしたが、今バージョンでは PC 版と区別するために変更されています。

UserAgent にて処理を分岐している場合修正の必要があります。

V1.1	Biz/Browser
Mobile	Biz/Browser Mobile

6.3 SYS オブジェクト

SYS オブジェクトの以下のプロパティ値が変更されました。

V1.1 では PC 版 V3.0 と同じ値でしたが、Mobile では独自の値を取ります。

	Biz/Browser for PDA V1.1	Biz/Browser Mobile
SYS_CLIENT	Biz/Browser	Biz/Browser Mobile
SYS_CLIENT_VERSION	3.042	2.000

6.4 使用できないオブジェクト

V1.1 では Spread、TabFrame などの使用できないオブジェクトを定義してもエラーにはなりませんでしたが、Mobile では「オブジェクトは作成できません」というエラーになります。

6.5 使用できないメソッド

PC 版 V3.0 でサポートされていて Mobile でサポートされていないメソッドは V1.1 ではエラーにはなりませんでしたが Mobile ではエラーとなります。

6.6 使用できないプロパティ

PC 版 V3.0 でサポートされていて Mobile でサポートされていないプロパティは互換性維持のためエラーにはなりま

せんが、変更しても効果はありません。

6.7 定数「$MOBILE」の利用

Biz/Browser Mobile では実行時に定数$MOBILE が 1 となります。これを利用して CRS の実行の際前バージョンと処理を分岐したり PC 版 Biz/Browser 実行時には無視し Mobile 実行時のみ実行させるハイブリッドな CRS の記述が可能です。

6.8 追加されたオブジェクト、メソッド、プロパティ

V1.1 では使用できなかった機能が使用できる可能性があります。詳しくは「機能差異表.pdf」および後述のリファレンスをご参照ください。

6.9 接続ライセンスの管理

Biz/Browser Mobile では URL への接続時に接続ライセンスが必要になります。

アプリケーション作成者はモバイル端末にあらかじめライセンスをインストールしておくか、またはシステムログイン時にライセンスをチェックし、ライセンスが無い場合ライセンスをインポートするようなコードを追加する必要があります。

詳細に関しては第 5 章をご参照ください。

6.10 バグフィックスによる影響

以下特に影響が問題となりそうなバグフィックスについて記載いたします。

6.10.1 最上位フォームサイズがリサイズされない問題の修正

PC 版では最上位フォームのサイズはルートウィンドウのサイズに自動的に合わせられるのが仕様ですが、V1.1 ではバグのため自動的にフォームがリサイズされません。このためフォームを大きく設定した上で画面サイズ以上の位置にオブジェクトを配置してもスクロールバーが表示されませんでした。

今回問題が修正されたため、フォームはルートウィンドウサイズに自動的にリサイズされ、画面サイズ以上の位置にオブジェクトを配置すると今まで出現しなかったスクロールバーが表示される場合があります。

この場合画面サイズ内に収まるようオブジェクトを配置するか、または Form の Scroll プロパティに$NONE を指定してください。

6.10.2 URL エンコーディング方式の変更

URL エンコーディング・デコーディングが RFC に準拠した方式に変更されています。このためサーバへのリクエストパラメータに影響が出る場合があります。

6.10.3 フォーカス処理の問題の修正

SetFocus 時指定のないオブジェクトへフォーカスが移動する問題が修正されています。

このため GetFocus のイベント処理に影響がでる場合があります。

6.10.4 3 項演算子の処理の問題の修正

return 時の戻り値に直接3項演算子を用いた場合値が不正になる障害、および3項演算子を多重ネストした場合の値の不正が修正されています。

このため3項演算子を使用した処理で結果が変化する場合があります。

6.10.5 オブジェクトスコープの変更の問題の修正

[オブジェクト名] { ... }構文におけるオブジェクトスコープの一時変更時にループ処理の break、continue などで動作不良となる障害、および try-catch でスコープが元に戻らない障害が修正されています。

この修正によりオブジェクトスコープの一時変更での処理結果が変化する場合があります。

6.10.6 メソッドの引数型チェックの厳密化

メソッドの引数の型チェックがより厳密化されています。このため V1.1 では動作していたケースでエラーになる場合

があります。

6.10.7 Edit 系オブジェクトの MaxLength

V1.1 で問題の MaxLength の動作が修正されています。
また Mobile では MaxLength は PC 版と異なり文字数(Unicode 単位) で統一されました。PC 版で開発する場合差異を考慮する必要があります。

6.10.8 キー操作の優先順位の統一

オブジェクト毎に動作に差異があったキー押下による処理の優先度が統一されました。
ひとつのキー操作に対して複数の意味が割り当てられている状態で、実際にそのキー操作を行った場合、実行の優先順序は以下のようになっています。

1. Tab キー、ShiftTab キー、または NextTab プロパティ、PrevTab プロパティに設定されているキー操作は、優先的にフォーカス移動が実行されます
2. オブジェクト固有のキー操作が実行されます
3. Button 系オブジェクトの AltKey に割り当てられているキー操作が実行されます
4. KeyDown イベントが発生します

※ 1 回のキー操作で起こる動作は、フォーカス移動、オブジェクト固有操作、AltKey 動作、KeyDown イベント発行のいずれか1つです。

6.10.9 Unicode 変換時の問題

Biz/Browser Mobile では CRS 実行エンジン自体はデータを ShiftJIS で保持しており、画面描画時に Unicode へ変換を行い OS の API にて描画を行います。
内部的に保持しているデータが ShiftJIS 文字列として正常でない場合、画面描画時に”(N/A)”と表示されます。

7. 追加リファレンス

7.1 オブジェクト共通

7.1.1 プロパティの差異

プロパティ名	アクセス	型	デフォルト	説明
VerticalAlign	CR	Integer	$STD	Label を除いて無効です
IMEMode	CR	Integer	$STD	$OPEN、$CLOSE は機種により無視されます。
ToolTip	CRW	String		動作しません
FontKind	CR	Integer	$STD	$BOLD は$STD と同じになります
MaxLength	CR	Integer		PC 版と異なりバイト数ではなく文字数で指定します。
NextTabKey	CRW	Integer	$STD	$END、$S_END が追加されています
PrevTabKey	CRW	Integer	$STD	$END、$S_END が追加されています
SkipTabFocus	CRW	boolean	false	キー操作によるフォーカス受入れの抑止
AutoTabFocus	CRW	boolean	true	キー操作によるフォーカス移動の制御

アクセス	R	読み込み可能
	W	書き込み可能
	C	初期化可能

7.1.1.1 SkipTabFocus プロパティ

true にセットすると、キーボードによるフォーカス移動の対象外となります。false をセットするとキーボードによるフォーカス移動の対象となります。

補助的な機能を持つボタンなどに SkipTabFocus=true をセットすることにより、入力の中心となるオブジェクトだけにキーボードによるフォーカス移動を制限することができ、入力操作の効率を向上させることができます。

7.1.1.2 AutoTabFocus プロパティ

true にセットすると、TAB キーまたは NextTabKey、PrevTabKey に設定したキーの押下、AutoTab プロパティがあるクラスについては AutoTab が実行される操作で自動的に次のタブ順のオブジェクトにフォーカスを移します。

デフォルト値は true です。

false にセットすると、フォーカスを自動的に移動することはなくなり、ユーザのマウスオペレーションまたは、スクリプトからの SetFocus メソッド、MoveFocus メソッドでのみフォーカスが移動します。

ただし、ウィンドウの切り替え（ダイアログやメッセージボックスなど、Biz/Browser 自身からのポップアップ表示を含みます）による LostFocus は発生します。また、フォーカスを受けているオブジェクトを削除するなど、フォーカスを維持できない場合、フォーカスは別のオブジェクトに移動します。

false に設定した状態で TAB キーまたは NextTabKey、PrevTabKey に設定したキーの押下、AutoTab プロパティがあるクラスについては AutoTab が実行される操作を行ったとき、FocusOperation イベントが発生します。

7.1.2 メソッドの差異

メソッド名	説明
MoveFocus	フォーカス移動順序に従ったフォーカス移動

7.1.2.1 MoveFocus メソッド

説明　フォーカス移動順序に従ったフォーカス移動を行います。

direction を省略するか、$NEXTFOCUS を指定した場合、TAB キーを押下したときと同じ順序でフォーカスが移動します。***direction*** に$PREVFOCUS を指定した場合、SHFT+TAB と同じ順序でフォーカスが移動します。それ以外の値を指定したときはフォーカスは移動しません。

以下の場合、正しくフォーカスが移動しない場合があります。
- MoveFocus を呼び出したオブジェクト以外にキーボードフォーカスが設定されている場合
- 選択されていないオプションボタンなど本来フォーカスを受け取らない状態にあるオブジェクトの場合
- ダイアログに隠れている非活性の Window 内のオブジェクトの場合

標準的な利用方法は、AutoTabFocus＝False に設定したオブジェクトにおいて、FocusOperation イベントのイベントハンドラ内で利用します。典型的には、OnFocusOperation 内で入力値のエラーチェックなどを行い、フォーカス移動を許可する場合だけ、MoveFocus メソッドを実行するようにします。

呼出形式　obj.MoveFocus([***direction***])

戻り値　なし

引数　integer ***direction***

指定できるシンボル	意味
$NEXTFOCUS	次のオブジェクト
$PREVFOCUS	前のオブジェクト

例外　なし

使用例

```
if( MessageBox("確認してください","確認",$OkCancel)==OkSelected) {
        TextBox1.MoveFocus($PREVFOCUS);
}
```

関連項目

7.1.3 イベントの差異

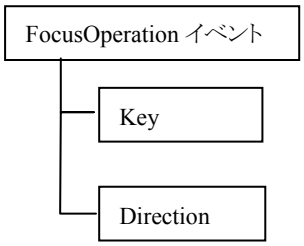
イベント名	説明
RClicked	発生しません
FocusOperation	キーボードフォーカスを受けている際の、フォーカス移動を起こす操作によって発生します

7.1.3.1 FocusOperation イベント

オブジェクトがキーボードフォーカスを受けている際に、フォーカスの移動を起こす操作を行ったときに発生します。フォーカスの移動を起こす操作とは、TABキーまたはNextTabKey、PrevTabKeyに設定したキーの押下、AutoTabプロパティがあるオブジェクトについては AutoTab が実行される操作です。これら以外の操作では FocusOperation イベントは発生しません。

FocusOperation イベントは、自動的なフォーカス移動を指示する AutoTabFocus プロパティの指定に従い、フォーカス移動とは排他的に発生します。AutoTabFocus が true の場合には、フォーカス移動を起こす操作でフォーカスが移動し、FocusOperation イベントは発生しません。逆に、FocusObject.AutoTabFocus プロパティが false の場合、フォーカスは移動せずに FocusOperation イベントが発生します。

FocusOperation イベントは以下のような構造をもちます。



Key
FocusOperation イベントの発生要因となったキーを示す文字列が設定されます。設定される文字列は、KeyDown イベントで Key に設定される文字列と同じです。SHFT キー、CTRL キー、ALT キーの状態は設定されません。

AutoTab プロパティの指定により FocusOperation イベントが発生した場合、このプロパティは"TAB"となります

Direction
フォーカスをどの方向に移動する操作を行ったかを示す以下の値が格納されます。

値	意味
$NEXTFOCUS	次のオブジェクト
$PREVFOCUS	前のオブジェクト

Direction の値は、FocusObject クラスの MoveFocus メソッドに渡すことで、本来のフォーカス移動と同じようにフォーカスを移す事ができます。

AutoTabFocus プロパティ、FocusOperation イベント、MoveFocus メソッド、および FocusOperation イベントを組み合わせて利用することにより、タブキーや Enter キーによるフォーカス移動動作に、入力値のチェックなどアプリケーション固有の動作を介入させる事ができます。以下の例のように、入力値が空の場合はフォーカス移動せずにエラーメッセージを表示し、正当な入力の場合には、次のオブジェクトにフォーカスを移すような動作です。

例

```
TextBox tx {
    :
    AutoTabFocus = false;
    :
    Function OnFocusOperation( e ) {
              if( value == "" ) {
                        BgColor = $red;
                        MessageBox("入力してください");
                        return;
              }
              BgColor = $std;
              MoveFocus( e.Direction );
     }
}
```

8. Standard GUI Package

8.1 Root

8.1.1 プロパティの差異

プロパティ名	アクセス	型	デフォルト	説明
ShellId	CRW	String		動作しません
Embedded	R	integer		動作しません
Resize	CRW	integer	$TRUE	動作しません

アクセス	R	読み込み可能
	W	書き込み可能
	C	初期化可能

8.1.2 メソッドの差異

メソッド名	説明
PostShellEvent	使用できません
ShowMenu	コマンドバーやステータスバーの表示を指定します

8.1.2.1 Root.ShowMenu メソッド

説明　コマンドバーやステータスバーの表示を指定します。
Mode に値を指定することで表示・非表示状態を変更します。
起動直後のデフォルト状態では表示状態です。

呼出形式　ShowMenu(***Mode***)

戻り値　なし

引数　Integer ***Mode***　以下の値の組み合わせで指定します。

Mode	説明
Root.Hide	非表示
Root.CommandBar	コマンドバーを表示
Root.StatusBar	ステータスバーを表示

例外　なし

使用例
```
ShowMenu(Root.CommandBar + Root.StatusBar);
```

関連項目

8.1.3 イベントの差異

イベント名	説明
WindowStateChanged	ウィンドウ状態の変化時に発生します

8.1.3.1 Root.WindowStateChanged イベント

ウィンドウに状態の変化が起きたときに発生します。たとえば縦横で画面を切り替えできる端末などでは縦横切り替え時に発生します。

イベントオブジェクトは次の構造を持ちます。

```
Event {
   Number   state;
}
```

state は参考情報でどのタイミングでどのような値が得られるかは保証されていません。

8.2 Label

8.2.1 プロパティの差異

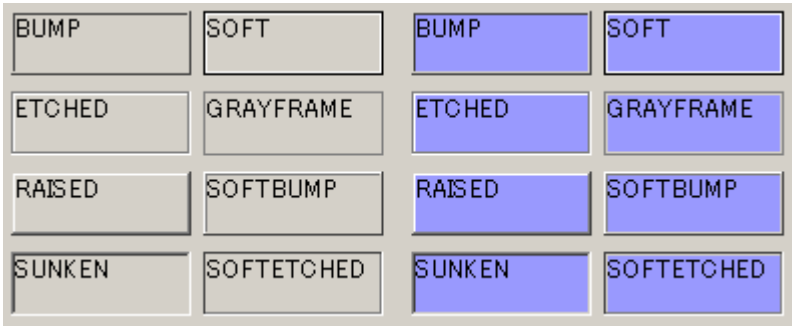
プロパティ名	アクセス	型	デフォルト	説明
BorderStyle	CRW	integer	$SUNKEN	枠のスタイル
VerticalAlign	CR	integer	$STD	縦方向割付け PC 版と異なり画面表示で動作します

8.2.1.1 Label.borderStyle プロパティ

枠のスタイルを以下の値で指定します。

シンボル	値	説明
$SUNKEN	0	Label 全体を凹形状で表示します
$RISED	1	Labe 全体を凸形状で表示します
$ETCHED	2	枠の線を凹形状で表示します
$BUMP	3	枠の線を凸形状で表示します
$GRAYFRAME	4	2 色の灰色で枠を表示します
$SOFT	5	白と黒で枠を表示します
$SOFTETCHED	6	枠の線をやわらかい凹形状で表示します
$SOFTBUMP	7	枠の線をやわらかい凸形状で表示します

表示例



Border プロパティを true に指定すると borderStyle で指定されたスタイルの枠を表示します。
false の場合、枠は表示されません。

8.3 EditBox

8.3.1 プロパティの差異

プロパティ名	アクセス	型	デフォルト	説明
ScrollBarPosition	CR	integer	$STD	HorizontalAlign=$RIGHT または Number 型の場合横スクロールバーは使用できません

アクセス	R	読み込み可能
	W	書き込み可能
	C	初期化可能

8.4 TextBox

8.4.1 プロパティの差異

プロパティ名	アクセス	型	デフォルト	説明
HorizontalAlign	CR	integer	$STD	動作しません

アクセス	R	読み込み可能
	W	書き込み可能
	C	初期化可能

8.5 CheckBox

8.5.1 プロパティの差異

プロパティ名	アクセス	型	デフォルト	説明
Scroll	CR	integer	$AUTO	$STATIC は無効です

アクセス	R	読み込み可能
	W	書き込み可能
	C	初期化可能

8.6 OptionButton

8.6.1 プロパティの差異

プロパティ名	アクセス	型	デフォルト	説明
Scroll	CR	integer	$AUTO	$STATIC は無効です

アクセス	R	読み込み可能
	W	書き込み可能
	C	初期化可能

8.7 Form

8.7.1 プロパティの差異

プロパティ名	アクセス	型	デフォルト	説明
BgHighQuality	CRW	Number	$FALSE	動作しません
BgPattern	CRW	Reference		動作しません
Scroll	CR	integer	$AUTO	$STATIC は無効です

アクセス	R	読み込み可能
	W	書き込み可能
	C	初期化可能

8.7.2 メソッドの差異

メソッド名	説明
FindSVGElement	使用できません
GetSVGElementPosition	使用できません

8.8 Spread

作成できません

8.9 SpreadRow

作成できません

8.10 SpreadColumn

作成できません

8.11 Graph

作成できません

8.12 GraphAxis

作成できません

8.13 GraphSeries

作成できません

8.14 Doc

作成できません

9. Extended GUI Package

9.1 NumberEdit

PC 版と異なり電卓入力のサポート機能はありません

9.1.1 プロパティの差異

プロパティ名	アクセス	型	デフォルト	説明
isNULL	CRW	boolean	false	Value が NULL かどうかを調べる

アクセス	R	読み込み可能
	W	書き込み可能
	C	初期化可能

9.1.1.1 NumberEdit.isNull プロパティ

シンボル	値	説明
$TRUE	ture	Delete キー押下で表示をクリアした場合です。Format に N がある場合は 0 の表示はされていません。
$FALSE	false	デフォルト値、0 が表示されています

Format プロパティに空のデータは表示しないフォーマットを指定されていて value が 0 の場合、このプロパティを参照すると delete キー押下による 0 なのか、0 を入力したのか、が判断できます。このとき値を false にすると 0 を表示し、true にすると 0 を非表示(空白)にします。

例)

```
NumberEdit num1 {
    Format="99ON";
    Function OnTouch(e) {
        if (value==0 && isNull == true) {
           MessageBox("何か入力してください");
        }
    }
}
Button btn {
    title="クリア";
    Function OnTouch(e) {
        ^.num1.isNull = true;
    }
}
```

Format プロパティに空のデータは表示しないフォーマットが指定されていない場合は、このプロパティを true にしても 0 を非表示(空白)になることはありません。また画面操作があったときに false に変わります。

9.2 DateEdit

PC 版と異なりカレンダー表示のサポート機能はありません

9.2.1 プロパティの差異

プロパティ名	アクセス	型	デフォルト	説明
AutoTab	CRW	boolean	false	全桁入力で自動的にフォーカス移動
InvalidDate	R	boolean	false	不正日付の確認
ShowFormat	CR	boolean	false	Format を表示します

アクセス	R	読み込み可能
	W	書き込み可能
	C	初期化可能

9.2.1.1 DateEdit.Autotab プロパティ

キーボードからの入力が最大桁数に達したときに、つぎの Tab 順のオブジェクトにフォーカスをセットします。

シンボル	値	説明
$FALSE	0	自動移動しない
$TRUE	0 以外	自動移動する

9.2.1.2 DateEdit.InvalidDate プロパティ

画面から入力した日付が不正な場合 true となります。正しい日付または未入力の場合 false となります。このプロパティと value プロパティを確認することにより、どのような入力が行われたか知ることができます。

value プロパティ	InvalidDate プロパティ	説明
0 より大きい	false	正しく日付が入力されている
0 より大きい	true	不正な日付が入力されている
0	false	なにも入力されていない
0	true	不正な日付が入力されている

このプロパティは、画面より入力した値が確定した時点でその値により設定されます。
スクリプトから Value プロパティの書き換えを行ったのみでは、反映されません。

9.2.1.3 DateEdit.ShowFormat プロパティ

Format を表示して何を入力すればよいかわかりやすくします。日付要素（YYYY、MM など）を_で表示します。

例
Format = "WRWYY\"年\"MM\"月\"DD\"日\"" は____年__月__日と表示します。

9.3 SVGButton

作成できません

9.4 TabFrame

作成できません

9.5 TabForm

作成できません

9.6 TreeView

作成できません

9.7 TreeItem

作成できません

10. CSV Package

10.1 CSVDocument

10.1.1 イベントの差異

イベント名	説明
BeginBreak	発生しません
KeyBreak	発生しません
EndBreak	発生しません

11. XML Package

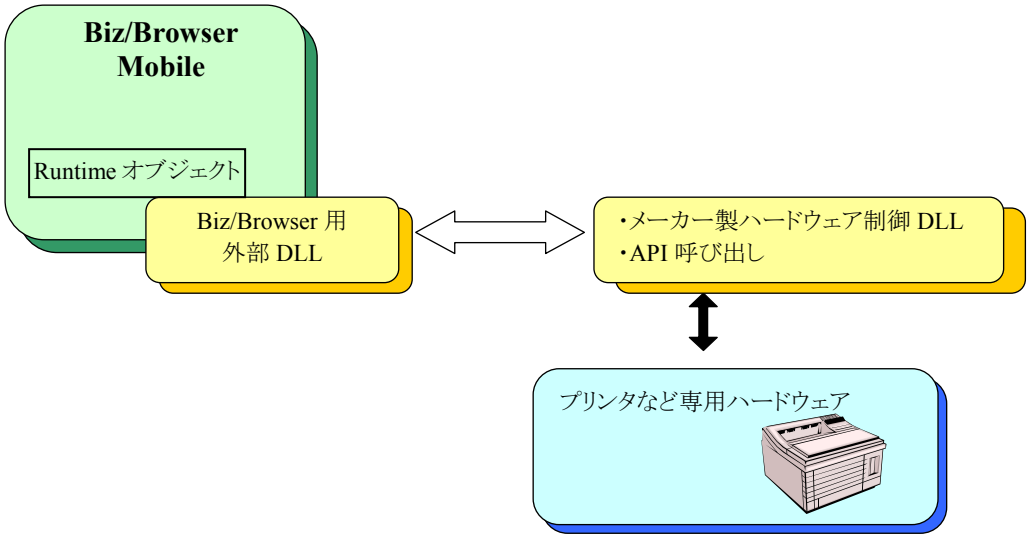
使用できません

12. Runtime Package

12.1 Runtime

PC 版と異なる Biz/Browser Mobile 独自の機能として外部 DLL のローディングおよび関数の呼び出しの機構を実装しています。

この機構によりモバイル端末特有のハードウェアのコントロールおよび値の取得が外部 DLL を介して Biz/Browser から行うことが可能になっています。



12.1.1 メソッドの差異

(コンストラクタ)	Runtime オブジェクトを初期化します。 CallDll メソッドで使用する DLL をローディングします。

メソッド名	説明
GetEnv	使用できません
ShellPrint	使用できません
ShellExplore	シェルによりディレクトリを開きます。 WindowsMobile では使用できません。
GetUUID	UUID を取得します
CallDll	DLL 内の関数を呼び出します

12.1.1.1 Runtime.コンストラクタ

説明　Runtime オブジェクトを初期化します。
引数を指定した場合、指定された DLL をローディングします。
DLL 内の関数は CallDll メソッドで呼び出すことができます。
Runtime オブジェクト消滅時に DLL も解放されます。

呼出形式　var dll = new Runtime([***dllPath***])

戻り値　Runtime オブジェクト

引数		
	String ***dllPath***	ロードするDLLを[Biz/Browserのインストールフォルダ]¥dll以下の相対パスで指定します。 省略した場合通常の Runtime オブジェクトとして初期化されます。

例外		
	RTM-15	DLL ロードエラー

使用例

```
var dll = new Runtime("Sample.dll");
```

関連項目　Runtime.CallDll()

12.1.1.2 Runtime.GetUUID メソッド

説明 UUID を取得します。

UUID(ユニバーサルユニーク ID)は汎用的な一意識別子で、世界中で重複することがなく、ユニークであることを目的に設計された 128bit のランダムな数値を文字列で表現したものです。例えば次のような文字列となります。

"DE07FFE8-A2FF-47CD-BA46-A993160F9A92"

ユニーク性を確保するために、UUID の作成には、作成した時間、作成したコンピュータの MAC アドレスなどの情報を参照することがあります。UUID が重複する可能性は極めて低い確率ですが、完全に重複しないことを保障するものではありません。

Windows 環境では OLE オブジェクト、ActiveX オブジェクトの識別子や製品固有の識別子として利用されています。GUID(グローバルユニーク ID)と呼ばれることもあります。

GetUUID メソッドの呼び出しで返される UUID は、***key*** とペアで記憶され、すでに ***key*** に対応する UUID が生成されている場合はその UUID を返します。***key*** に対応する UUID が生成されていない場合には、新しく生成した UUID を記録して返します。

記録された UUID は、Biz/Browser を再起動したり、端末を再起動しても有効ですが、端末のトラブルなどで失われる可能性があります。一度失われた UUID を復元する手段はありませんので、回復が不可能なデータを UUID の記録に依存して処理しない事をお勧めします。

key と UUID のペアは、オプションのパラメータ ***pass-phrase*** を指定しない場合は、すべてのアプリケーションで共有される名前空間に記録されます。***pass-phrase*** を指定した場合には、GetUUID を実行した CRS をダウンロードしたサーバ固有の名前空間に記録されます。つまり、Server-A からダウンロードした CRS と Server-B からダウンロードした CRS で同じキーを指定して GetUUID メソッドを呼び出した場合、***pass-phrase*** を指定しないときには同じ UUID が返され、***pass-phrase*** を指定すると、それぞれは別のキーとして扱われ、別の UUID が返されます。

オプションのパラメータ ***pass-phrase*** を指定しない場合は、UUID はクリアテキストで記録されますが、***pass-phrase*** を指定すると暗号化されて記録されます。一度 ***pass-phrase*** を指定して記録された UUID は、同じ ***pass-phrase*** を指定して GetUUID を呼び出さなければ取得できず、異なる ***pass-phrase*** や ***pass-phrase*** を省略した場合 RTM-23 例外が throw されます。

呼出形式 var result = rt.GetUUID(***key*** [, ***pass-phrase***])

戻り値 UUID を文字列で返します。

引数		
	String ***key***	UUID に対応する識別子。16 文字までの英数字が指定できます。
	String ***pass-phrase***	UUID を暗号化して記録する場合のパスフレーズ。

例外		
	RTM-22	識別子が不正です
	RTM-21	レジストリアクセスエラー
	RTM-23	UUID が生成できません

使用例

```
var rt = new Runtime;
var uuid = rt.GetUUID("appkey", "pass");
```

関連項目

12.1.1.3 Runtime.CallDll メソッド

説明　Runtime コンストラクタでローディングされた DLL 内の関数を呼び出します。

呼出形式　var ret = dll.CallDll(***funcName,*** [, ***param1*** [,***param2*** [, ...]])

戻り値　DLL 関数から返される値

引数	String ***funcName***	呼び出す関数名を指定します。 関数名は DLL でエクスポートされた名前を正確に記述する必要があります。
	paramN	DLL 関数に渡すパラメータを指定します
例外	RTM-18	外部モジュールでエラーが発生しました

使用例

```
var dll = new Runtime("Sample.dll");
var ret = dll.CallDll("GetValue", "Param1", 2);
```

関連項目　Runtime.コンストラクタ

12.2 FileSystem

12.2.1 メソッドの差異

メソッド名	説明
Rename	ファイル名を変更します 進捗ダイアログは表示されません。 またモード指定で FileSystem.SILENT を指定しても無効です。
Copy	ファイルを複製します 進捗ダイアログは表示されません。 またモード指定で FileSystem.SILENT を指定しても無効です。

12.3 Security Package

Security Package はセキュリティ関連機能を持つオブジェクトを集めたパッケージです。

Security パッケージでサポートされる暗号化アルゴリズムは以下のとおりで、すべてブロック暗号方式です。ブロック暗号化モードは、暗号ブロック連鎖モード（CBC）方式を採用しています。また、DES,トリプル DES、RC2 のアルゴリズムについては、Windows の持つセキュリティ API 機能に依存しており、Microsoft Enhanced Cryptographic Provider がインストールされていない環境では利用できません。Microsoft Enhanced Cryptographic Provider は、米国からのセキュリティ製品輸出が制限された、一部の地域や国では利用できない場合があります。

なお、PC 版に搭載されている BlowFish 暗号化アルゴリズムは搭載されていません。

サポートされる暗号化アルゴリズム

DES アルゴリズム

56 ビットの固定長キーを持つ、64 ビットのブロック暗号方式です。鍵のビット長が短いため、鍵の総当りが現実的な時間内で実行可能という脆弱性を持っています。

トリプル DES アルゴリズム

DES の脆弱性を解決するために考案された方式で、DES アルゴリズムを 3 回通す方式です。3 つの異なるキーを利用する E-E-E 方式と、2 個の異なるキーを利用する E-D-E 方式があります。

RC2 アルゴリズム

RSA 社で開発されたアルゴリズムで、可変長のキーを持つ 64 ビットのブロック暗号方式です。Security パッケージでは、Microsoft Enhanced Cryptographic Provider の標準値である 128 ビットのキー長を使用しています。

サポートされるハッシュアルゴリズム

SHA アルゴリズム

米国標準のハッシュアルゴリズムです。160 ビットのダイジェストを作成します。

MD4 アルゴリズム

RSA 社の Rives 氏により考案されたアルゴリズムで、128 ビットのダイジェストを作成します。

MD5 アルゴリズム

RSA 社の Rives 氏により考案されたアルゴリズムで、128 ビットのダイジェストを作成します。異なるデータから同一のダイジェストを生成する確立が MD4 よりも低く、良質なダイジェストを生成できます。

オブジェクト

クラス名	説明
Encryptor	暗号化クラス
Decryptor	暗号復号クラス
Hash	ハッシュ生成クラス

12.3.1 Encryptor

Encryptor は、元となるデータに対して暗号化処理を行うクラスです。暗号化したデータを元に戻すためには Decryptor を使用します。

暗号化するデータに応じて 2 種類の方法で暗号化することができます。
ひとつは、File オブジェクトや HTTPResponse オブジェクトなどの入出力を暗号化する方法で、Encryptor オブジェクトを生成するときのコンストラクタにデータの入出力に利用するオブジェクトを指定します。このオブジェクトを Read オブジェクトまたは Write オブジェクトといいます。

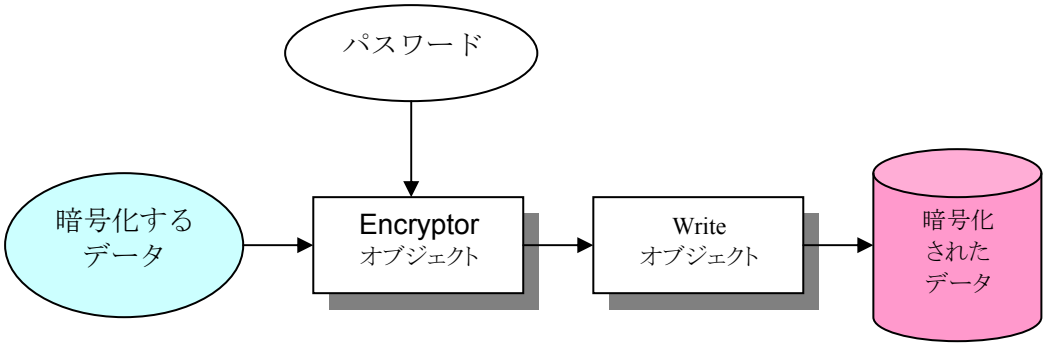
Read オブジェクトは Read メソッドを持つオブジェクトで、読み込みモードで作成した File オブジェクトや HTTPResponse オブジェクトが該当します。コンストラクタに Read オブジェクトを指定した場合、Encryptor オブジェクトの出力系メソッド、Write と WriteString を使用することはできません。

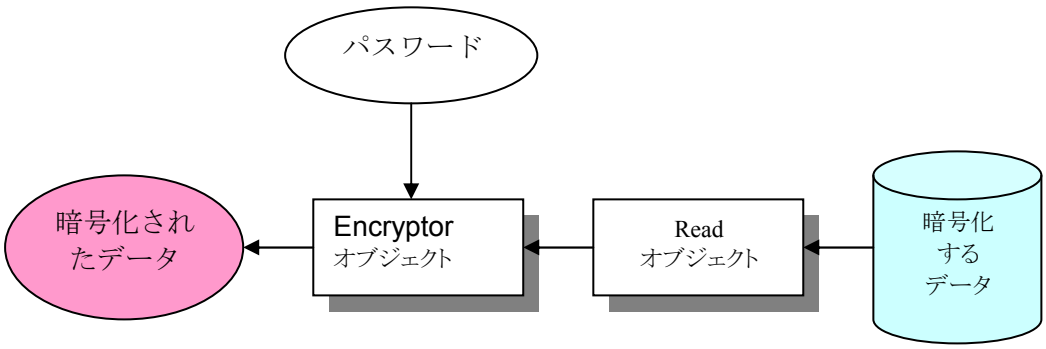
Write オブジェクトは Write メソッドを持つオブジェクトで、書き込みモードで作成した File オブジェクトや HTTPRequest オブジェクトが該当します。コンストラクタに Write オブジェクトを指定した場合、Encryptor オブジェクトの入力系メソッド、Read を使用することはできません。

使用例

```
var fp = fs.Open("data.txt", FileSystem.OPEN_WRITE);
var enc = new Encryptor( fp, "password", Encryptor.CALG_3DES_EEE);
enc.writeString("暗号化したいデータ");
enc.close(true);
```

この例では、Encryptor オブジェクト enc を File オブジェクト fp を引数に初期化しているので、enc.write で暗号化したデータが fp の指すファイルに書き込まれます。このように Encryptor オブジェクトは、暗号化を行うフィルターのような動作をします。





もうひとつの方法は、文字列を直接暗号化する方法です。

```
var enc_data = Encryptor.EncryptString("暗号化する文字列","password", Encryptor.CALG_3DES_EEE);
```

この例では、"暗号化する文字列"を暗号化して enc_data 変数に格納しています。

12.3.1.1 プロパティ

Encryptor で定義されるプロパティは以下のとおりです。

プロパティ名	アクセス	型	デフォルト	説明
Algorithm	R	integer		アルゴリズム
FileName	R	String		ファイル名
PathName	R	String		パス名

アクセス	R	読み込み可能
	W	書き込み可能
	C	初期化可能

12.3.1.2 Encryptor.Algorithm プロパティ

暗号化に利用されるアルゴリズムを番号で示します。
現在以下のアルゴリズムが定義されています。

アルゴリズム	シンボル	値
DES	Encryptor.CALG_DES	1
トリプル DES E-E-E	Encryptor.CALG_3DES_EEE	2
トリプル DES E-D-E	Encryptor.CALG_3DES_EDE	3
RC2	Encryptor.CALG_RC2	4

12.3.1.3 Encryptor.FileName プロパティ

暗号化してデータを入出力するファイル名を示します。
このプロパティは、コンストラクタで指定した Read/Write オブジェクトの FileName プロパティがコピーされています。Read/Write オブジェクトに FileName プロパティが存在しない場合、空文字列となります。

12.3.1.4 Encryptor.PathName プロパティ

暗号化してデータを入出力するファイル名を示します。
このプロパティは、コンストラクタで指定した Read/Write オブジェクトの PathName プロパティがコピーされています。Read/Write オブジェクトに PathName プロパティが存在しない場合、空文字列となります。

12.3.1.5 メソッド

Encryptor で定義されるメソッドは以下のとおりです。

(コンストラクタ)	Encryptor オブジェクトを初期化します

スタティックメソッド

メソッド名	説明
EncryptString	文字列を暗号化します

メソッド

メソッド名	説明
Close	クローズします
Read	暗号化データを読み取ります
Write	暗号化データを出力します
WriteString	暗号化した文字列を出力します

12.3.1.6 Encryptor.コンストラクタ

説明　Encryptor オブジェクトを初期化します。

呼出形式　var enc = new Encryptor(***rw_obj***, ***pass***, ***calg***);

戻り値　初期化された Encryptor オブジェクト

引数

Object ***rw_obj***　暗号化データの出力先となる Write オブジェクトまたは、暗号化するデータの入力元となる Read オブジェクトを指定します。

Write オブジェクトは、Write メソッドを持つオブジェクトで、書き込みモードで作成した File オブジェクトなどが相当します。

Read オブジェクトは、Read メソッドを持つオブジェクトで、読み込みモードで作成した File オブジェクトなどが相当します。

String ***pass***　4 文字以上56文字以下のパスワードを指定します。

Integer ***calg***　以下から暗号化アルゴリズムを 1 つ指定します。

アルゴリズム	シンボル	値
DES	Encryptor.CALG_DES	1
トリプル DES E-E-E	Encryptor.CALG_3DES_EEE	2
トリプル DES E-D-E	Encryptor.CALG_3DES_EDE	3
RC2	Encryptor.CALG_RC2	4

例外

CRS-16	コンストラクタ引数が不正です
SEC-1	有効な入出力オブジェクトが指定されていません
SEC-2	暗号化アルゴリズムが不明です
SEC-3	パスフレーズが不正です

使用例

```
var fs = new FileSystem();
var fp = fs.open("data.txt",FileSystem.OPEN_WRITE);
var enc = new Encryptor(fp, "passwd", Encryptor. CALG_3DES_EEE);
enc.WriteString("暗号化するデータ");
enc.Close(true);
```

関連項目

12.3.1.7 Encryptor. EncryptString メソッド

説明　スタティックメソッド。
文字列を暗号化して Base64 形式の文字列で返します。

EncryptString メソッドはスタティックなメソッドで Encryptor オブジェクトを生成することなく呼び出すことができます。

呼出形式　var enc = Encryptor. EncryptString (***data***, ***pass***, ***calg***);

戻り値　暗号化されたデータを Base64 形式の文字列で返します。

引数

String ***data***	暗号化する文字列
String ***pass***	4 文字以上56文字以下のパスワードを指定します。
Integer ***calg***	以下から暗号化アルゴリズムを 1 つ指定します。

アルゴリズム	シンボル	値
DES	Encryptor.CALG_DES	1
トリプル DES E-E-E	Encryptor.CALG_3DES_EEE	2
トリプル DES E-D-E	Encryptor.CALG_3DES_EDE	3
RC2	Encryptor.CALG_RC2	4

例外

CRS-34	メモリ不足です
SEC-2	暗号化アルゴリズムが不明です
SEC-3	パスフレーズが不正です

使用例

```
var enc = Encryptor.EncryptString("暗号化するデータ","pass",Encryptor.CALG_3DES_EEE);
```

関連項目

12.3.1.8 Encryptor.Close メソッド

説明　出力または入力を終了します。

Encryptor オブジェクトを出力用途に使用する場合、Write または WriteString メソッドによる出力内容は一時的に内部のバッファに保存されます。Close メソッドを呼び出したときにバッファに残っている出力内容をエンコードして Write オブジェクトに渡されます。Close メソッドを実行しない場合、出力内容の一部またはすべてが Write オブジェクトに渡されることなく失われることがありますので、必ず Close メソッドを呼び出してください。

呼出形式　enc.Close([***deep***]);

戻り値　なし

引数

boolean ***deep***	true を指定した場合、Read/Write オブジェクトの Close メソッドを呼び出します。
	それ以外の場合、Encryptor オブジェクトをクローズしますが、Read/Write オブジェクトの Close メソッドは実行されません。

例外

CRS-34	メモリ不足です
SEC-1	有効な入出力オブジェクトが指定されていません
SEC-7	オペレーションシーケンスが不正です
SEC-8	Write メソッドがありません

使用例

```
var fs = new FileSystem();
var fp = fs.open("data.txt",FileSystem.OPEN_WRITE);
var enc = new Encryptor(fp, "passwd", Encryptor. CALG_3DES_EEE);
enc.WriteString("暗号化するデータ");
enc.Close(true);
```

関連項目

12.3.1.9 Encryptor.Write メソッド

説明　暗号化されたデータを書き込みます。

Write メソッドに渡したデータは暗号化された後、Write オブジェクトの Write メソッドに引き渡されます。

暗号化アルゴリズムはブロック単位で処理するため、Write メソッドを実行しても即座に Write オブジェクトに渡されず、内部のバッファに記録された状態で復帰することがあります。必ず最後に Close メソッドを呼び出してバッファをフラッシュする必要があります。

呼出形式　enc.Write(***data***);

戻り値　なし

引数

data	出力するデータを指定します。 通常、String オブジェクトを指定します。 Number や Date など。その他のオブジェクトを指定することも可能ですが、オブジェクト内部で保持されるデータのビットパターンがそのまま出力されるため、意味のある結果は得られません。

例外

CRS-34	メモリ不足です
SEC-1	有効な入出力オブジェクトが指定されていません
SEC-5	出力データの型が不明です
SEC-7	オペレーションシーケンスが不正です

使用例

```
var fs = new FileSystem();
var fp = fs.open("data.txt",FileSystem.OPEN_WRITE);
var enc = new Encryptor(fp, "passwd", Encryptor. CALG_3DES_EEE);
enc.Write("暗号化するデータ");
enc.Close(true);
```

関連項目

12.3.1.10 Encryptor.WriteString メソッド

説明　暗号化された文字列を書き込みます。

WriteString メソッドに渡したデータは文字列に変換された後、暗号化され、Write オブジェクトの Write メソッドに引き渡されます。

暗号化アルゴリズムはブロック単位で処理するため、WriteString メソッドを実行しても即座に Write オブジェクトに渡されず、内部のバッファに記録された状態で復帰することがあります。必ず最後に Close メソッドを呼び出してバッファをフラッシュする必要があります。

呼出形式　enc.WriteString(***data***);

戻り値　なし

引数

String ***data***	出力するデータを指定します。

例外

CRS-34	メモリ不足です
SEC-1	有効な入出力オブジェクトが指定されていません
SEC-7	オペレーションシーケンスが不正です

使用例

```
var fs = new FileSystem();
var fp = fs.open("data.txt",FileSystem.OPEN_WRITE);
var enc = new Encryptor(fp, "passwd", Encryptor. CALG_3DES_EEE);
enc.WriteString("暗号化するデータ");
enc.Close(true);
```

関連項目

12.3.1.11 Encryptor.Read メソッド

説明　暗号化されたデータを読み込みます。

Read オブジェクトの Read メソッドを呼び出し、得られたデータを暗号化して返します。

呼出形式　var d = enc.Read([***size***]);

戻り値　String オブジェクトで暗号化されたデータを返します。

引数		
	Integer ***size***	Read オブジェクトから読み込む最大のバイト数を指定します。省略するか -1 を指定するとファイルの最後まで読み取ります。

例外		
	CRS-34	メモリ不足です
	SEC-1	有効な入出力オブジェクトが指定されていません
	SEC-7	オペレーションシーケンスが不正です
	SEC-9	Read メソッドがありません
	SEC-10	読み込み操作を完了できませんでした

使用例

```
var fs = new FileSystem();
var fp = fs.open("data.txt",FileSystem.OPEN_READ);
var enc = new Encryptor(fp, "passwd", Encryptor. CALG_3DES_EEE);
var d = enc.Read();
enc.Close(true);
```

関連項目

12.3.1.12 イベント

Encryptor で定義されるイベントはありません。

12.3.2 Decryptor

Decryptor は、暗号化されたデータに対して復号化処理を行うクラスです。データを暗号化するためには Encryptor を使用します。

暗号化されたデータに応じて 2 種類の方法で復号化することができます。
ひとつは、File オブジェクトや HTTPResponse オブジェクトなどの入出力を暗号化する方法で、Decryptor オブジェクトを生成するときのコンストラクタにデータの入出力に利用するオブジェクトを指定します。このオブジェクトを Read オブジェクトまたは Write オブジェクトといいます。

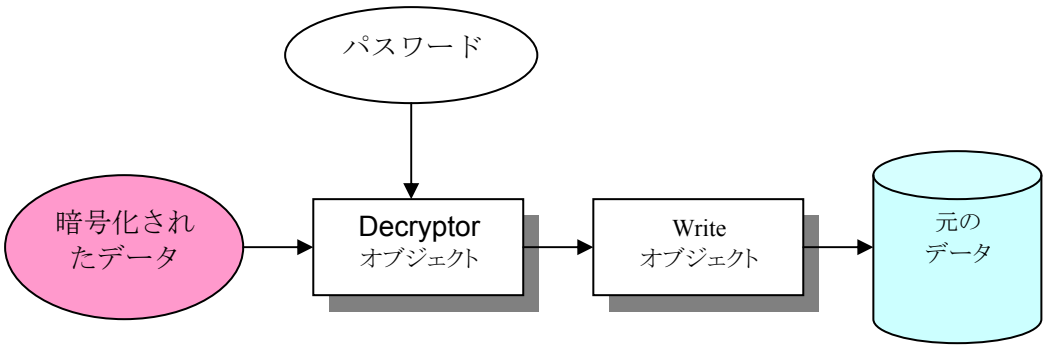
Read オブジェクトは Read メソッドを持つオブジェクトで、読み込みモードで作成した File オブジェクトや HTTPResponse オブジェクトが該当します。コンストラクタに Read オブジェクトを指定した場合、Decryptor オブジェクトの出力系メソッド、Write と WriteString を使用することはできません。

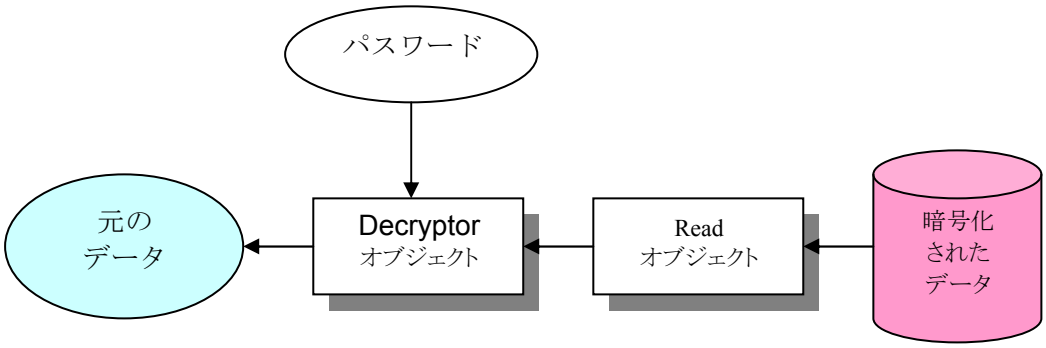
Write オブジェクトは Write メソッドを持つオブジェクトで、書き込みモードで作成した File オブジェクトや HTTPRequest オブジェクトが該当します。コンストラクタに Write オブジェクトを指定した場合、Decryptor オブジェクトの入力系メソッド、Read を使用することはできません。

使用例

```
var fp = fs.Open("data.txt", FileSystem.OPEN_READ);
var dec = new Decryptor( fp, "password", Decryptor.CALG_3DES_EEE);
var data = dec.Read();
dec.close(true);
```

この例では、Decryptor オブジェクト dec を File オブジェクト fp を引数に初期化しているので、dec.read で暗号化したデータが fp の指すファイルから読み込まれ復号化した後で戻り値として返されます。このように Decryptor オブジェクトは、復号化を行うフィルターのような動作をします。





もうひとつの方法は、文字列を直接復号化する方法です。

```
  var dec_data = Decryptor.DecryptString("暗号化された文字列","password",
Encryptor.CALG_3DES_EEE);
```

この例では、"暗号化された文字列"を復号化して dec_data 変数に格納しています

12.3.2.1 プロパティ

Decryptor で定義されるプロパティは以下のとおりです。

プロパティ名	アクセス	型	デフォルト	説明
Algorithm	R	integer		アルゴリズム
FileName	R	String		ファイル名
PathName	R	String		パス名

アクセス	R	読み込み可能
	W	書き込み可能
	C	初期化可能

12.3.2.2 Decryptor.Algorithm プロパティ

暗号化に利用されるアルゴリズムを番号で示します。
現在以下のアルゴリズムが定義されています。

アルゴリズム	シンボル	値
DES	Decryptor.CALG_DES	1
トリプル DES E-E-E	Decryptor.CALG_3DES_EEE	2
トリプル DES E-D-E	Decryptor.CALG_3DES_EDE	3
RC2	Decryptor.CALG_RC2	4

12.3.2.3 Decryptor.FileName プロパティ

暗号化してデータを入出力するファイル名を示します。
このプロパティは、コンストラクタで指定した Read/Write オブジェクトの FileName プロパティがコピーされています。Read/Write オブジェクトに FileName プロパティが存在しない場合、空文字列となります。

12.3.2.4 Decryptor.PathName プロパティ

暗号化してデータを入出力するファイル名を示します。
このプロパティは、コンストラクタで指定した Read/Write オブジェクトの PathName プロパティがコピーされています。Read/Write オブジェクトに PathName プロパティが存在しない場合、空文字列となります。

12.3.2.5 メソッド

Decryptor で定義されるメソッドは以下のとおりです。

(コンストラクタ)	Decryptor オブジェクトを初期化します

スタティックメソッド

メソッド名	説明
DecryptString	文字列を暗号化します

メソッド

メソッド名	説明
Close	クローズします
Read	暗号化データを読み取ります
Write	暗号化データを出力します
WriteString	暗号化した文字列を出力します

12.3.2.6 Decryptor.コンストラクタ

説明　Decryptor オブジェクトを初期化します。

呼出形式　var enc = new Decryptor(***rw_obj***, ***pass***, ***calg***);

戻り値　初期化された Decryptor オブジェクト

引数

Object ***rw_obj***　復号したデータの出力先となる Write オブジェクトまたは、復号するデータの入力元となる Read オブジェクトを指定します。

Write オブジェクトは、Write メソッドを持つオブジェクトで、書き込みモードで作成した File オブジェクトなどが相当します。

Read オブジェクトは、Read メソッドを持つオブジェクトで、読み込みモードで作成した File オブジェクトなどが相当します。

String ***pass***　4 文字以上56文字以下のパスワードを指定します。

Integer ***calg***　以下から暗号化アルゴリズムを 1 つ指定します。

アルゴリズム	シンボル	値
DES	Decryptor.CALG_DES	1
トリプル DES E-E-E	Decryptor.CALG_3DES_EEE	2
トリプル DES E-D-E	Decryptor.CALG_3DES_EDE	3
RC2	Decryptor.CALG_RC2	4

例外

CRS-16	コンストラクタ引数が不正です
SEC-1	有効な入出力オブジェクトが指定されていません
SEC-2	暗号化アルゴリズムが不明です
SEC-3	パスフレーズが不正です

使用例

```
var fs = new FileSystem();
var fp = fs.open("data.txt",FileSystem.OPEN_READ);
var dec = new Decryptor(fp, "passwd", Decryptor. CALG_3DES_EEE);
var data = dec.Read();
dec.Close(true);
```

関連項目

12.3.2.7 Decryptor. DecryptString メソッド

説明　スタティックメソッド。
Encryptor.EncryptString で暗号化された文字列を復号します。

pass,***calg*** パラメータは、EncryptString メソッドで暗号化したときと同じものを指定します。

DecryptString メソッドはスタティックなメソッドで Decryptor オブジェクトを生成することなく呼び出すことができます。

呼出形式　var enc = Decryptor. DecryptString (***data***, ***pass***, ***calg***);

戻り値　復号化した文字列を返します。

引数

String ***data***	復号化する文字列
String ***pass***	4 文字以上56文字以下のパスワードを指定します。
Integer ***calg***	以下から暗号化アルゴリズムを 1 つ指定します。

アルゴリズム	シンボル	値
DES	Decryptor.CALG_DES	1
トリプル DES E-E-E	Encriptor.CALG_3DES_EEE	2
トリプル DES E-D-E	Encriptor.CALG_3DES_EDE	3
RC2	Encriptor.CALG_RC2	4

例外

CRS-34	メモリ不足です
SEC-2	暗号化アルゴリズムが不明です
SEC-3	パスフレーズが不正です
SEC-11	復号化操作を完了できませんでした

使用例

```
var enc = Decryptor.DecryptString("複合化するデータ","pass",Decryptor.CALG_3DES_EEE);
```

関連項目

12.3.2.8 Decryptor.Close メソッド

説明　出力または入力を終了します。

Decryptor オブジェクトを出力用途に使用する場合、Write または WriteString メソッドによる出力内容は一時的に内部のバッファに保存されます。Close メソッドを呼び出したときにバッファに残っている出力内容をエンコードして Write オブジェクトに渡されます。Close メソッドを実行しない場合、出力内容の一部またはすべてが Write オブジェクトに渡されることなく失われることがありますので、必ず Close メソッドを呼び出してください。

呼出形式　dnc.Close([***deep***]);

戻り値　なし

引数

boolean ***deep***	true を指定した場合、Read/Write オブジェクトの Close メソッドを呼び出します。 それ以外の場合、Decryptor オブジェクトをクローズしますが、Read/Write オブジェクトの Close メソッドは実行されません。

例外

CRS-34	メモリ不足です
SEC-1	有効な入出力オブジェクトが指定されていません
SEC-7	オペレーションシーケンスが不正です

使用例

```
var fs = new FileSystem();
var fp = fs.open("data.txt",FileSystem.OPEN_READ);
var dec = new Decryptor(fp, "passwd", Decryptor. CALG_3DES_EEE);
var data = dec.Read();
dec.Close(true);
```

関連項目

12.3.2.9 Decryptor.Read メソッド

説明　復号化されたデータを読み込みます。

Read オブジェクトの Read メソッドを呼び出し、得られたデータを復号化して返します。

呼出形式　var data = dec.Read([***size***]);

戻り値　String オブジェクトで復号化されたデータを返します。

引数		
	Integer ***size***	Read オブジェクトから読み込む最大のバイト数を指定します。省略するか -1 を指定するとファイルの最後まで読み取ります。

例外		
	CRS-34	メモリ不足です
	SEC-1	有効な入出力オブジェクトが指定されていません
	SEC-7	オペレーションシーケンスが不正です
	SEC-9	Read メソッドがありません
	SEC-10	読み込み操作を完了できませんでした

使用例

```
var fs = new FileSystem();
var fp = fs.open("data.txt",FileSystem.OPEN_READ);
var dnc = new Decryptor(fp, "passwd", Decryptor. CALG_3DES_EEE);
var d = dnc.Read();
dec.Close(true);
```

関連項目

12.3.2.10 Decryptor.ReadLine メソッド

説明　復号化されたデータを 1 行読み込みます。

Read オブジェクトの Read メソッドを呼び出し、得られたデータを復号化して 1 行を返します。

呼出形式　var data = dec.ReadLine();

戻り値　復号化された文字列を返します。

引数　なし

例外

CRS-34	メモリ不足です
SEC-1	有効な入出力オブジェクトが指定されていません
SEC-7	オペレーションシーケンスが不正です
SEC-9	Read メソッドがありません
SEC-10	読み込み操作を完了できませんでした

使用例

```
var fs = new FileSystem();
var fp = fs.open("data.txt",FileSystem.OPEN_READ);
var dec = new Decryptor(fp, "passwd", Decryptor. CALG_3DES_EEE);
var d = dec.ReadLine();
dec.Close(true);
```

関連項目

12.3.2.11 Decryptor.Write メソッド

説明　復号化されたデータを書き込みます。

Write メソッドに渡したデータは復号化された後、Write オブジェクトの Write メソッドに引き渡されます。

暗号化アルゴリズムはブロック単位で処理するため、Write メソッドを実行しても即座に Write オブジェクトに渡されず、内部のバッファに記録された状態で復帰することがあります。必ず最後に Close メソッドを呼び出してバッファをフラッシュする必要があります。

呼出形式　dec.Write(***data***);

戻り値　なし

引数

String ***data***	出力するデータを指定します。

例外

CRS-34	メモリ不足です
SEC-1	有効な入出力オブジェクトが指定されていません
SEC-5	出力データの型が不明です
SEC-7	オペレーションシーケンスが不正です

使用例

```
var fs = new FileSystem();
var fp = fs.open("data.txt",FileSystem.OPEN_WRITE);
var dec = new Decryptor(fp, "passwd", Decryptor. CALG_3DES_EEE);
dec.Write(encrypted_data);
dec.Close(true);
```

関連項目

12.3.2.12 イベント

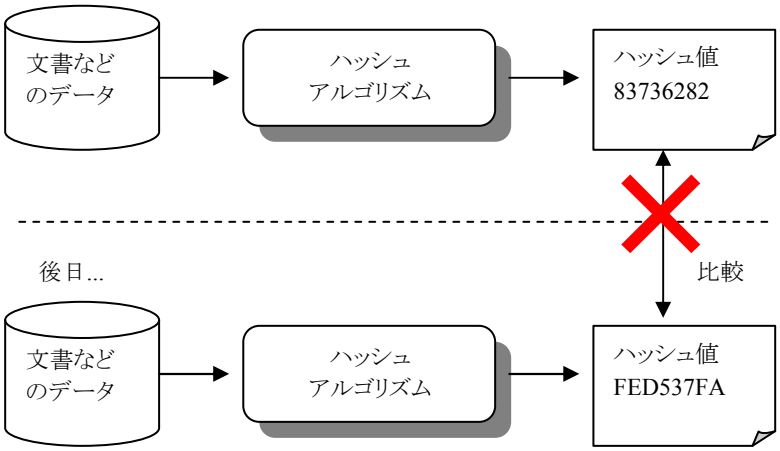
Decryptor で定義されるイベントはありません。

12.3.3 Hash

Hash はハッシュ値を計算するためのクラスです。

ハッシュ値とは、文書や数値などのデータからハッシュアルゴリズムにより算出される固定長の値で、元のデータの要約として利用できます。ハッシュ値は単なる数値の羅列となるため、ハッシュ値から元のデータを推定することは不可能となります。

この特徴を使って、ある文書のハッシュ値を算出して保存しておき、後日同じ文書から算出したハッシュ値と保存しておいたハッシュ値と比較すれば、その間に元の文書が改竄されていないことを確認することができます。元の文書全体を別の場所に保存して比較しても同様の事ができますが、ハッシュ値は文書の大きさに関わりなく固定長の短いデータとなるため保存したり比較する事が容易になります。



電子署名でもハッシュが使われており、署名するデータのハッシュ値(メッセージダイジェスト)を算出して公開鍵暗号アルゴリズムで暗号化されています。公開鍵で解読したハッシュ値と、元のデータから算出し直したハッシュ値が一致すれば、間違いなく公開鍵の持ち主により作成されたデータで、改竄もされていないことが保障されます。

Hash オブジェクトでハッシュ値を算出する方法は、2 通り用意されています。

ひとつは、File オブジェクトや HTTPResponse オブジェクトなどの入出力データから算出する方法で、Hash オブジェクトを生成するときのコンストラクタにデータの入出力に利用するオブジェクトを指定します。このオブジェクトを Read オブジェクトまたは Write オブジェクトといいます。

Read オブジェクトは Read メソッドを持つオブジェクトで、読み込みモードで作成した File オブジェクトや HTTPResponse オブジェクトが該当します。コンストラクタに Read オブジェクトを指定した場合、Hash オブジェクトの出力系メソッド、Write と WriteString を使用することはできません。

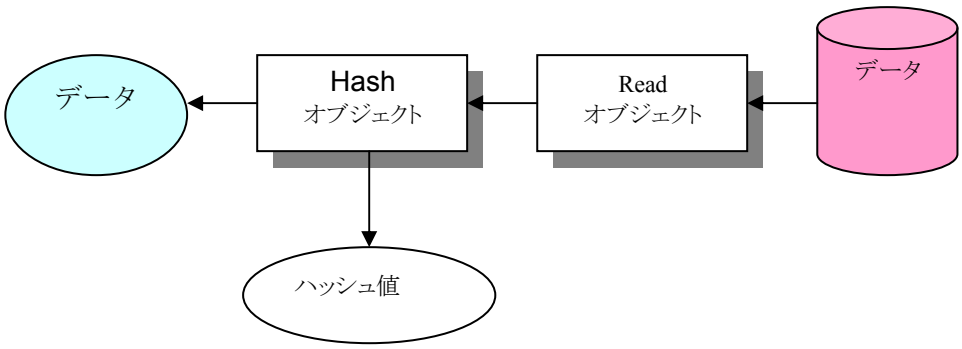
Write オブジェクトは Write メソッドを持つオブジェクトで、書き込みモードで作成した File オブジェクトや HTTPRequest オブジェクトが該当します。コンストラクタに Write オブジェクトを指定した場合、Hash オブジェクトの入力系メソッド、Read と ReadString を使用することはできません。

使用例

```
var fp = fs.Open("data.txt",FileSystem.OPEN_READ);
var hs = new Hash( fp, Hash.HALG_MD5);
hs.Read();
hs.Close(true);
MessageBox( hs.GetHashData(), "Hash");
```

この例では、Hash オブジェクト hs を File オブジェクト fp を引数に初期化しています。続く Read メソッドでファイルをロードしていますが、Hash オブジェクトを経由するため、fp の指すファイル data.txt のハッシュ値も同時に算出されます。

このように Hash オブジェクトは、ハッシュ値の算出を行うフィルターのような動作をします。



もうひとつの方法は、Hash クラスのスタティックメソッド、HashString を利用して文字列から直接 Hash 値を算出する方法です。

使用例

```
var hash_value = Hash.HashString( "ハッシュ値を算出する文字列", Hash.HALG_MD5 );
```

この例では、"ハッシュ値を算出する文字列" のハッシュ値を hash_value 変数に格納しています。

12.3.3.1 プロパティ

Hash で定義されるプロパティは以下のとおりです。

プロパティ名	アクセス	型	デフォルト	説明
Algorithm	R	integer		アルゴリズム
FileName	R	String		ファイル名
PathName	R	String		パス名

アクセス	R	読み込み可能
	W	書き込み可能
	C	初期化可能

12.3.3.2 Hash.Algorithm プロパティ

ハッシュアルゴリズムを番号で示します。
現在以下のアルゴリズムが定義されています。

アルゴリズム	シンボル	値
SHA1 アルゴリズム	Hash.HALG_SHA1	0
MD4 アルゴリズム	Hash.HALG_MD4	1
MD5 アルゴリズム	Hash.HALG_MD5	2

12.3.3.3 Hash.FileName プロパティ

Read/Write オブジェクトのファイル名を示します。
このプロパティは、コンストラクタで指定した Read/Write オブジェクトの FileName プロパティがコピーされています。Read/Write オブジェクトに FileName プロパティが存在しない場合、空文字列となります。

12.3.3.4 Hash.PathName プロパティ

データを入出力するファイル名を示します。
このプロパティは、コンストラクタで指定した Read/Write オブジェクトの PathName プロパティがコピーされています。Read/Write オブジェクトに PathName プロパティが存在しない場合、空文字列となります。

12.3.3.5 メソッド

Hash で定義されるメソッドは以下のとおりです。

(コンストラクタ)	Hash オブジェクトを初期化します

スタティックメソッド

メソッド名	説明
HashString	文字列からハッシュ値を算出します

メソッド

メソッド名	説明
Close	クローズします
Read	データを読み取ります
ReadLine	文字列を 1 行読み出します
Write	データを出力します
WriteString	文字列を書き込みます
GetHashData	算出されたハッシュ値を取得します

12.3.3.6 Hash.コンストラクタ

説明　Hash オブジェクトを初期化します。

呼出形式　var h = new Hash(***rw_obj***, ***halg***);

戻り値　初期化された Hash オブジェクト

引数

object ***rw_obj***

ハッシュ値を算出するデータの出力先となる Write オブジェクトまたは、入力元となる Read オブジェクトを指定します。

Write オブジェクトは、Write メソッドを持つオブジェクトで、書き込みモードで作成した File オブジェクトなどが相当します。

Read オブジェクトは、Read メソッドを持つオブジェクトで、読み込みモードで作成した File オブジェクトなどが相当します。

Read オブジェクトを指定した場合、Read または ReadString で Read オブジェクトから読み取られるデータのハッシュ値が算出されます。

Write オブジェクトを指定した場合、Write または WriteString メソッドで Write オブジェクトに書き込まれるデータのハッシュ値が算出されます。

どちらの場合でも、必ず Close メソッドを呼び出して入出力操作を完結した後で、GetHashData メソッドでハッシュ値を取得します。

Integer ***halg***

以下からハッシュアルゴリズムを 1 つ指定します。

アルゴリズム	シンボル	値
SHA アルゴリズム	Hash.HALG_SHA	0
MD4 アルゴリズム	Hash.HALG_MD4	1
MD5 アルゴリズム	Hash.HALG_MD5	2

例外

CRS-16	コンストラクタ引数が不正です
SEC-1	有効な入出力オブジェクトが指定されていません
SEC-12	ハッシュアルゴリズムが不明です

使用例

```
var fs = new FileSystem();
var fp = fs.open("data.txt",FileSystem.OPEN_READ);
var hs = new Hash(fp,Hash.HALG_MD5);
var data = hs.Read();
hs.Close(true);
MessageBox( hs.GetHashData(), "data.txtのハッシュ値");
```

関連項目

12.3.3.7 Hash. HashString メソッド

説明　スタティックメソッド。
文字列からハッシュ値を算出します。

HashString メソッドはスタティックなメソッドで Hash オブジェクトを生成することなく呼び出すことができます。

呼出形式　var data = Hash.HashString (***data***, ***halg***);

戻り値　算出したハッシュ値を Base64 形式の文字列で返します。

引数

String ***data***	ハッシュ値を算出する文字列
Integer ***halg***	以下からハッシュアルゴリズムを 1 つ指定します。

アルゴリズム	シンボル	値
SHA1 アルゴリズム	HASH.HALG_SHA1	0
MD4 アルゴリズム	HASH.HALG_MD4	1
MD5 アルゴリズム	HASH.HALG_MD5	2

例外

CRS-34	メモリ不足です
SEC-12	ハッシュアルゴリズムが不明です
SEC-13	セキュリティプロバイダでエラーが発生しました

使用例

```
var hash_value = Hash.HashString("データ",HASH.HALG_MD5);
```

関連項目

12.3.3.8 Hash.Close メソッド

説明　出力または入力を終了します。

Hash オブジェクトからハッシュ値を取得する場合、必ず Close メソッドの呼び出しが必要です。Close メソッドを呼び出さずに GetHashData メソッドを呼び出すと、SEC-7 例外が発生します。

呼出形式　hs.Close([***deep***]);

戻り値　なし

引数

boolean ***deep***	true を指定した場合、Read/Write オブジェクトの Close メソッドを呼び出します。 それ以外の場合、Hash オブジェクトをクローズしますが、Read/Write オブジェクトの Close メソッドは実行されません。

例外

CRS-34	メモリ不足です
SEC-1	有効な入出力オブジェクトが指定されていません
SEC-7	オペレーションシーケンスが不正です
SEC-13	セキュリティプロバイダでエラーが発生しました

使用例

```
var fs = new FileSystem();
var fp = fs.open("data.txt",FileSystem.OPEN_READ);
var hs = new Hash(fp, Hash.HALG_MD5);
var data = hs.Read();
hs.Close(true);
var hash_value = hs.GetHashData();
```

関連項目

12.3.3.9 Hash.Read メソッド

説明　コンストラクタで指定した Read オブジェクトよりデータを読み込みます。読み込まれたデータは、内部でハッシュ値の算出が行われます。

Read オブジェクトのデータをすべて読み込まなかった場合でもハッシュ値の算出は Read オブジェクトの残りのデータを含めて算出されます。

呼出形式　var data = hs.Read([***size***]);

戻り値　ByteArray オブジェクトで読み込まれたデータを返します。

引数

Integer ***size***	Read オブジェクトから読み込む最大のバイト数を指定します。省略するか -1 を指定するとファイルの最後まで読み取ります。

例外

CRS-34	メモリ不足です
SEC-1	有効な入出力オブジェクトが指定されていません
SEC-7	オペレーションシーケンスが不正です
SEC-9	Read メソッドがありません
SEC-10	読み込み操作を完了できませんでした
SEC-13	セキュリティプロバイダでエラーが発生しました

使用例

```
var fs = new FileSystem();
var fp = fs.open("data.txt",FileSystem.OPEN_READ);
var hs = new Hash(fp, Hash.HALG_MD5);
var d = hs.Read();
hs.Close(true);
```

関連項目

12.3.3.10 Hash.ReadLine メソッド

説明　Read オブジェクトより、データを 1 行読み込みます。読み込まれたデータは、内部でハッシュ値の算出が行われます。

Read オブジェクトのデータをすべて読み込まなかった場合でもハッシュ値の算出は Read オブジェクトの残りのデータを含めて算出されます。

呼出形式　var data = hs.ReadLine();

戻り値　読み込まれた文字列を返します。

引数　なし

例外

CRS-34	メモリ不足です
SEC-1	有効な入出力オブジェクトが指定されていません
SEC-7	オペレーションシーケンスが不正です
SEC-9	Read メソッドがありません
SEC-10	読み込み操作を完了できませんでした
SEC-13	セキュリティプロバイダでエラーが発生しました

使用例

```
var fs = new FileSystem();
var fp = fs.open("data.txt",FileSystem.OPEN_READ);
var hs = new Hash(fp, Hash.HALG_MD5);
var d = hs.ReadLine();
hs.Close(true);
```

関連項目

12.3.3.11 Hash.Write メソッド

説明　データをWriteオブジェクトに書き込みます。書き込まれたデータは内部でハッシュ値の算出が行われます。

呼出形式　hs.Write(***data***);

戻り値　なし

引数		
	String ***data***	出力するデータを指定します。

例外		
	CRS-34	メモリ不足です
	SEC-1	有効な入出力オブジェクトが指定されていません
	SEC-5	出力データの型が不明です
	SEC-7	オペレーションシーケンスが不正です

使用例

```
var fs = new FileSystem();
var fp = fs.open("data.txt",FileSystem.OPEN_WRITE);
var hs = new Hash(fp, Hash.HALG_MD5);
hs.Write(data);
hs.Close(true);
```

関連項目

12.3.3.12 Hash.WriteString メソッド

説明　Write オブジェクトに文字列を書き込みます。書き込まれたデータは内部でハッシュ値の算出が行われます。

呼出形式　hs.WriteString(***data***);

戻り値　なし

引数

String	***data***	出力するデータを指定します。

例外

CRS-34	メモリ不足です
SEC-1	有効な入出力オブジェクトが指定されていません
SEC-7	オペレーションシーケンスが不正です

使用例

```
var fs = new FileSystem();
var fp = fs.open("data.txt",FileSystem.OPEN_WRITE);
var hs = new Hash(fp, Hash.HALG_MD5);
hs.WriteString("ハッシュ値を算出するデータ");
hs.Close(true);
```

関連項目

12.3.3.13 Hash.GetHashData メソッド

説明　算出されたハッシュ値を取得します。

ハッシュ値の算出は、Read、ReadLine、Write、WriteString の各メソッドの呼び出して入出力されたデータに対して行われ、内部のバッファに記録されます。

Close メソッドを呼び出しハッシュ値を確定してから GetHashData メソッドを呼び出してください。Close メソッドを呼び出さずに GetHashData を呼び出すと SEC-7 例外が Throw されます。

呼出形式　var hash_value = hs.GetHashData();

戻り値　ハッシュ値を Base64 形式の文字列で返します。

引数　なし

例外　SEC-7　オペレーションシーケンスが不正です

使用例

```
var fs = new FileSystem();
var fp = fs.open("data.txt",FileSystem.OPEN_WRITE);
var hs = new Hash(fp, Hash.HALG_MD5);
var d = hs.WriteString("ハッシュ値を算出するデータ");
hs.Close(true);
MessageBox(hs.GetHashData());
```

関連項目

12.3.3.14 イベント

Hash で定義されるイベントはありません。

12.4 グローバル関数

12.4.1 HasConnectionLicense 関数

説明　***URL*** で示される接続先に対する接続ライセンスを取得しているか調べます。

接続ライセンスの確認は、***URL*** のプロトコル名、ホスト名、ポート番号を示す文字列(http://server.co.jp:8088 など)で識別されます。IP アドレスの直接表記やホスト名の別名など、同一のサーバコンピュータを示す各種の表記方法がありますが、接続ライセンス証明書に記載されている URL と厳密に一致しなければなりません。

呼出形式　var flag = HasConnectionLicense(***URL***)

戻り値　***URL*** に接続可能なライセンスを取得している場合、true が返ります。それ以外の場合、false が返ります。

引数　String ***URL***　　確認する接続先を示す URL

例外　なし

使用例

```
if( ! HasConnectionLicense("http://server.co.jp") ) {
        ImportConnectionLicese("http://server.co.jp/License.xml");
}
```

関連項目　ImportConnectionLicense 関数

12.4.2 ImportConnectionLicense 関数

説明　***URL*** で示される場所から接続ライセンス証明書をダウンロードして登録します。

URL は接続ライセンス証明書に記載されているURLとは無関係で、任意の ***URL*** に配置することができます。

ライセンス証明書は、ImportConnectionLicense を呼び出した Biz/Browser プロセスへの登録、およびファイルへの記録を行います。同時に実行されている別の Biz/Browser プロセスに対して直接作用する事はありません。

ファイルへの記録は、Biz/Broswer のインストールフォルダの settings.v2¥license ディレクトリに保存されます。
ファイルへの記録が何らかの理由により失敗した場合でも、Biz/Browser のプロセスへの登録は有効で、エラーとはなりませんが、ファイルへの記録は行われずプロセスの終了とともに登録されたライセンス情報は失われます。
ファイルに記録されたライセンスは、次回の起動時に自動的に Biz/Browser にロードされます。

ロードする接続ライセンスと同じシリアル番号を持つ接続ライセンスが、既にプロセスに登録されている場合、発効日が新しい場合には古いものを上書きし、古い場合には CRS-399 例外が throw されます。

呼出形式　ImportConnectionLicense (***URL***)

戻り値　なし

引数		
	String URL	ライセンス証明書をダウンロードする URL

例外		
	CRS-331	通信エラー
	CRS-366	ライセンス証明書が正規のライセンスとして認識できませんでした
	CRS-367	指定のライセンス証明書よりも新しいライセンスが既に登録されています

使用例

```
if( ! HasConnectionLicense("http://server.co.jp") ) {
        ImportConnectionLicense ("http://server.co.jp/License.xml");
}
```

関連項目　HasConnectionLicense 関数

Biz/Browser Mobile 追補マニュアル

2006 年 10 月 20 日　 Ver.1.0

発行:アクシスソフト株式会社
〒170－0013　東京都豊島区東池袋 3－23－5　ダヴィンチ池袋ビル