インターネットプロジェクト

Global Program アプレットの作成

2000/7/7

改訂版号	更新者	更新日	更新内容
1.00	IBM 石村	2000/07/07	初版

目次

図表目次

1. はじめに

本書は東京海上殿インターネットプロジェクト、Global Program用に開発された、ファイルダウンロード、ダウンロードファイルの実行を実現するアプレットについて、またそのアプレットの作成方法について記したものです。

2. 処理概略

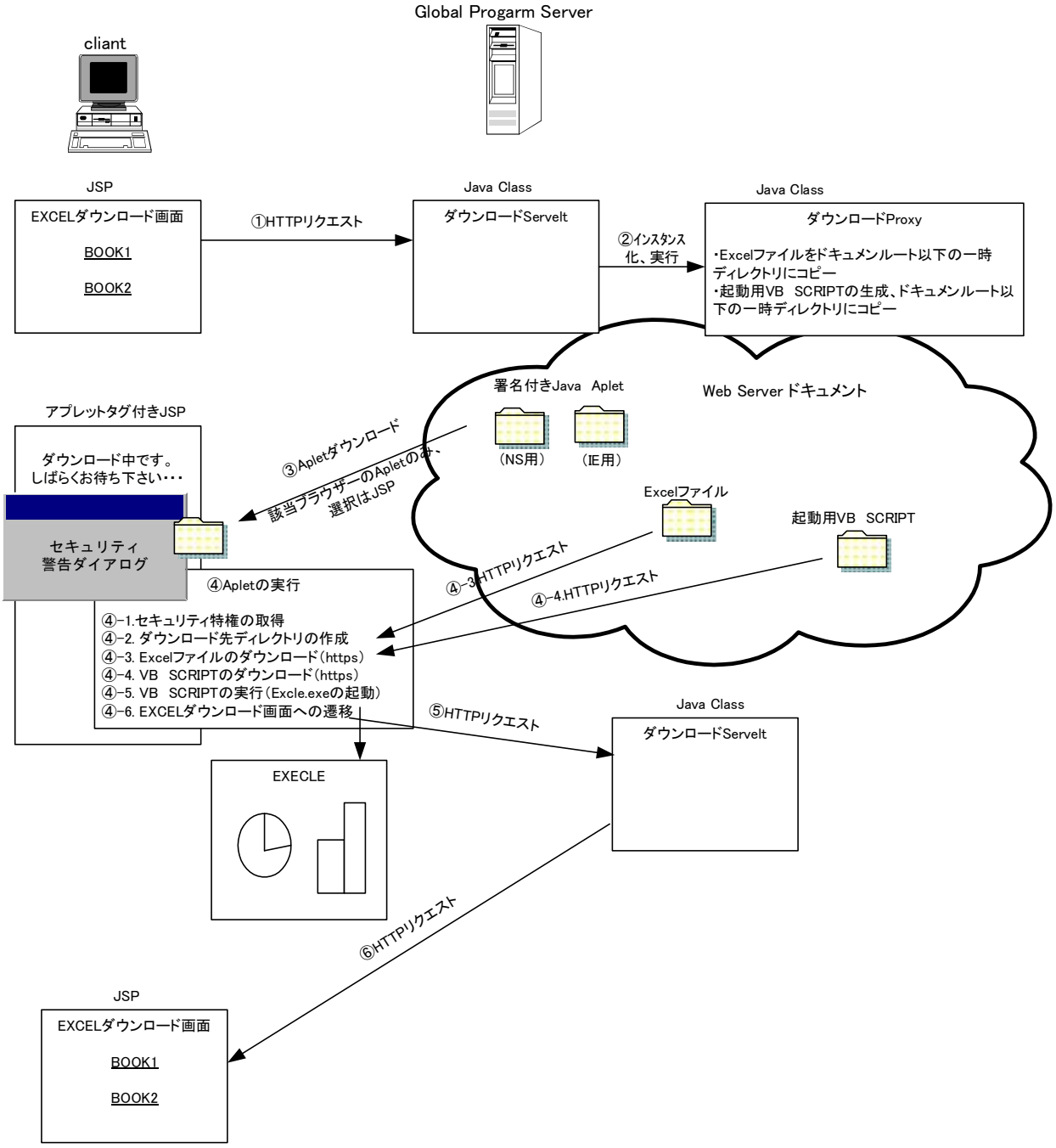

図 1　　ダウンロード概略図

① ライアントからダウンロード Servlet にファイルのダウンロードのリクエストを出します。

② ダウンロード Servlet からダウンロード Proxy をインスタン化、実行します。Proxy はダウンロード対象のファイルをドキュメントルート配下の一時ディレクトリにコピー、Excel 起動用の VB スクリプトファイルを生成して、一時ディレクトリにコピーします。

③ Proxy はアプレット実行の JSP をクライアントに返します。JSP は NS、MSIE を判断して該当ブラウザ用の署名付きアプレットをダウンロード、実行します。(呼び出しもとの JSP と、Servlet の情報をアプレットに引数として渡します)

④ アプレットでは、セキュリティ特権の取得、他必要処理の後ファイルをダウンロードして、Excel を起動します。

⑤ アプレットは Excel 起動後ダウンロード Servlet に対してレスポンスを返します。

⑥ ダウンロード Servlet は、HTTP リクエスト(レスポンス)から、ファイルダウンロードの処理を呼び出した画面をクライアントに返します。

3. アプレット処理フロー

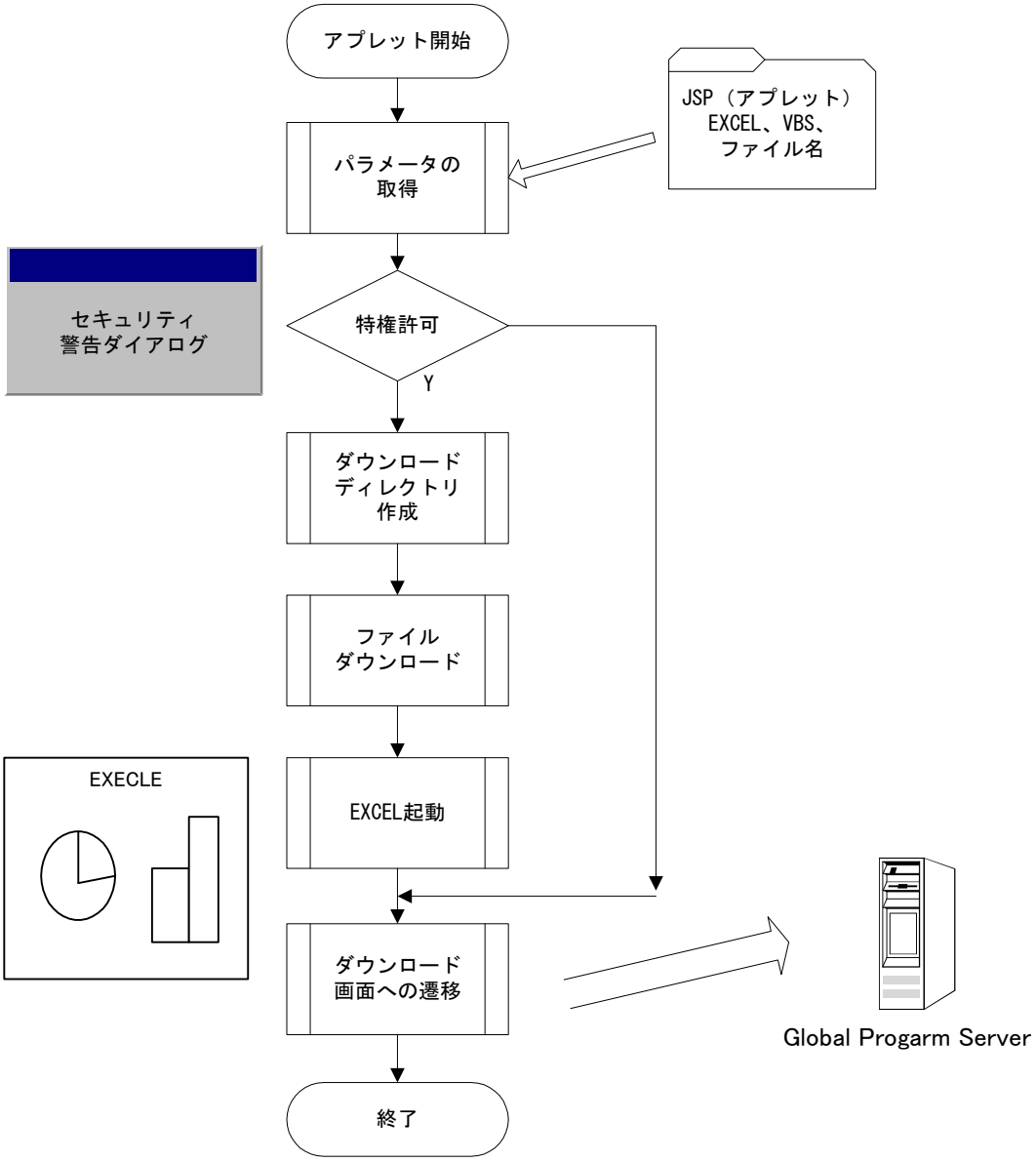

図 2　　アプレット処理フロー

4. アプレット

アプレットは、Netscape と、Microsoft Internet Explorer で処理、記述が異なるために、2つのアプレットファイルがあります。

ブラウザ	アプレット名
NSN	NsnPostExcel.jar
MSIE	MsiePostExcel.cab

表 1　　アプレット

4.1. アプレットの引数

アプレットに渡す引数の一覧です。

名前	説明	サンプル値
XLS_NAME	ローカル XLS ファイル名	c:/gblp/graph/gblp_graph.xla
VBS_NAME	ローカル VBS ファイル名	c:/gblp/graph/gblp_startup.vbs
XLS_FILE	リモート XLS ファイル名	http://www8.tokiomarine.co.jp/www16/Glp/DL/120000710175830/gblp_graph.xla
VBS_FILE	リモート VBS ファイル名	http://www8.tokiomarine.co.jp/www16/Glp/DL/120000710175830/gblp_startup.vbs
CALL_ME_URL	ダウンロードを呼び出した URL	http://www8.tokiomarine.co.jp/www16/servlet/Glp.GlpDownload.GlpDownloadServlet
CALL_ME_JSP_ID	ダウンロードを呼び出した JSPID	CLAIM22
CSV1_NAME	ローカル CSV1 ファイル名	c:/gblp/graph/gblp_graph.csv
CSV2_NAME	ローカル CSV2 ファイル名	c:/gblp/graph/gblp_select.csv
CSV1_FILE	リモート CSV1ファイル名	http://10.192.132.95/Glp/DL/120000710175830/gblp_graph.csv
CSV2_FILE	リモート CSV2 ファイル名	http://10.192.132.95/Glp/DL/120000710175830/gblp_select.csv

表 2　　アプレットの引数

4.2. アプレットの認証クラス

アプレットがクライアントのリソースにアクセスする為に、アプレットはクライアントに必要な特権を許可してもらう必要があります。

4.2.1. NSN のアプレット認証クラス

NSN では以下の 4 つの特権を要求しています。

```
PrivilegeManager.enablePrivilege( "UniversalFileRead" );          //ファイルの読み込み
PrivilegeManager.enablePrivilege( "UniversalFileWrite" );         //ファイルの書き込み
PrivilegeManager.enablePrivilege( "UniversalConnect" );           //他のサーバーへの接続
PrivilegeManager.enablePrivilege( "UniversalExecAccess");         //アプリケーションの実行
```

4.2.2. MSIE のアプレット認証クラス

MSIE では以下の2つの特権を要求しています。

```
PolicyEngine.assertPermission( PermissionID.FILEIO );             //ファイルの読み書き
PolicyEngine.assertPermission( PermissionID.EXEC );               //アプリケーションの実行
```

5. デジタル証明書でアプレットにサインする方法

アプレットにデジタル証明書を付加する方法はそれを認識するブラウザの種類によって異なります。

本書は Netscape Navigator と、Internet Explorer(以下はそれぞれ NSN、MSIE と表記)を対象とする。

5.1. NSN 編

NSN で実行されるアプレットはファイルの拡張子が jar となります。

以下の手順の作業は Netscape Navigator がインストールされている任意のユーザーのディレクトリで作業を行います。
(C:¥Program Files¥Netscape¥Users¥default) この場合ユーザーの default
Netscape 社提供の signtool をパスの通ったディレクトリに置きます。
signtool の実行は、ブラウザを終了した状態で行ってください。

あらかじめ、ブラウザに

1) 署名するファイルの準備
ユーザーディレクトリ配下に readfile のディレクトリを作成して、作成したディレクトリの配下にアプレットの Java のクラスファイル(実行に必要なファイル全て)を置きます。
パッケージ名も含めてディレクトリを作成し、ファイルを置いてください。

2) 証明書のインポート
ブラウザを表示させます。
ブラウザメニューの[communicator]-[ツール]-[セキュリティ情報]で、[証明書]-[本人]のページを開いて、[証明書をインポート]ボタンよりインポートします。
注)一度ブラウザを終了しないと証明書が有効にならない場合があります

3) 証明書の確認
サインする証明書が署名に使えることを確認します。
＞signtool -L
コマンドを実行すると、一覧が表示されます。行頭にアスタリスクマークを持っているものが署名に使えることを意味します。

4) 署名付き JAR ファイルの作成
signtool を使用して、署名付き期アプレット(サインドアプレット)を作成します。
signtool コマンドの-k オプションで、デジタル証明書を指定し、-Z オプションで JAR ファイル名を指定します。署名すべきクラスファイルが格納されているディレクトリ名をパラメータとして渡します。
＞signtool –k “Certificate Name” –Z NsnPostExce.jar readfile
Certificate Name : 署名する証明書の名前
NsnPostExce.jar : 作成するアプレットの名前
readfile : 署名するクラスファイルのディレクトリ(1)で作成したディレクトリ)

5) JAR ファイルの確認
作成されたファイルに正しくクラスが追加されていることを確認します。
＞signtool –v NsnPostExce.jar

5.1.1. テスト証明書の作成

実際に、VeriSign 等公的機関より証明書を取得できない場合に、テストとして「Test CA」の証明書を作成するには以下のコマンドを実行してください。コマンドを実行すると x509.cacert のファイルがカレントに作成されます。
＞signtool –G “Test CA”

5.2. MSIE 編

MSIE で実行されるアプレットはファイルの拡張子が cab となります。

以下の手順の作業は、作成済みの Java クラスファイルの存在するディレクトリで作業を行います。
Microsoft 社提供の PackSign にパスを通します。

1） 証明書の準備

アプレットに署名する証明書を Java クラスファイルの存在するディレクトリに置きます。

2） 署名付き CAB ファイルの作成

cabarc を使用して、署名付き期アプレット（サインドアプレット）を作成します。
コマンドにnと-p オプションを渡して、作成す CAB ファイル名と、CAB に組み込むクラスファイルを指定します。

＞ cabarc −p n MsiePostExcel.cab Glp¥GlpDownload¥PostExcel.class + Glp¥GlpDownload¥MsiePostExcel.class + Glp¥GlpDownload¥MsgDialog.class + Glp¥GlpDownload¥MsgDialog$IvjEventHandler.class

MsiePostExcel.cab	:作成するアプレットの名前
Glp¥GlpDownload¥PostExcel.class	:CAB ファイルに組み込むクラスファイル
Glp¥GlpDownload¥MsiePostExcel.class	: 〃
Glp¥GlpDownload¥MsgDialog.class	: 〃
Glp¥GlpDownload¥MsgDialog$IvjEventHandler.class	: 〃

3） CAB ファイルの確認

作成した CAB ファイルにクラスが正しく登録されているかの確認。

＞cabarc l MsiePostExcel..cab

4） CAB ファイルに証明を付ける

作成した CAB ファイルに署名を付け加えます。

＞signcode −j javasign.dll −jp low −spc mycredentials.spc −v myprivatekey.pvk MsiePostExcel.cab

mycredentials.spc	:SPC 形式の証明書
myprivatekey.pvk	:PVK 形式の証明書
MsiePostExcel.cab	:作成した CAB ファイル

5.2.1. テスト用証明書の作成

実際に、VeriSign 等公的機関より証明書を取得できない場合に、テストとして「Test CA」の証明書を作成するには以下のコマンドを実行してください。コマンドを実行すると msieapp.cer のファイル（x509 形式）がカレントに作成されます。

＞makecert –sk appkey –n “CN=Test CA” msieapp.cer

次に以下のコマンドで証明書を signcode コマンドの引数に指定するための形式(.spc)に変換します。

＞cert2spc msieapp.cer msieapp.spc

6. 署名アプレットの実行

verisign 等公的機関の証明書で作成したアプレットでない場合、クライアントに署名者の証明書をインストールする必要があります。
その場合、署名者の証明書をサーバーに置いてクライアントに設定する必要があります。

ブラウザ	MIME-TYPE	ファイル拡張子
NSN	application/x-x509-ca-cert	cacert
MSIE	application/pkix-cert	cer

表 3　　MIME-TYPE

verisign で署名したアプレットは、クラインとに署名者の証明書をインストールする必要はありません。
ブラウザが予め、verisign の署名者の証明書を持っています。

6.1. サーバーへファイルを置く

ドキュメントルート配下に、作成したアプレットのファイル(JAR、CAB)を置きます。
6.3アプレットタグ で指定する CODEBASE の URL の場所にクラスファイルを置きます。
ファイル、ディレクトリの属性値を変更してください。
(運用手順書第 1 版のディレクトリ構造を参照してください)

アプレットは、タグで指定された URL からアプレットをダウンロードしてきます。また、CODEBASE で指定された URL を基準としてクラスファイルを見つけます。

6.2. ブラウザの設定

NSN、MSIE 両ブラウザとも、Java を有効にしていないとアプレットは動作しません。
また、Global Program では、アプレットでの処理が終了すると、アプレットのコードの中に記述されている部分でアプレットを呼び出した画面に戻りますが、アプレットそのものが実行されないと「しばらくお待ちください」の画面のまま全画面に戻ることもできないので注意してください。
以下にそれぞれの設定を記します。

6.2.1. NSN のブラウザの設定

1) [編集]－[設定]メニューから設定プロパティダイアログを開いてください。
2) [詳細]タグの「Java を有効にする」をチェックしてください。(デフォルトでは ON になっています)

6.2.2. MSIE のブラウザの設定

MSIE のインストーラーの設定次第では JavaVM がインストールされていない場合があります。
その場合、MSIE のインストーラーで JavaVM のチェックをしてインストールしてください。

1) [ツール]－[インターネットオプション]メニューから、インターネットオプションダイアログを開いてください。
2) [セキュリティ]タブの「インターネット」または、「イントラネット」を選択してください。(サーバーがイントラのサーバーの場合、「イントラ」を、以外は「インターネット」を選択してください)
3) 「レベルのカスタマイズ」を押して、セキュリティの設定ダイアログを表示してください。
4) 「Java」－「Java の許可」をカスタム設定として、「Java のカスタム設定」を押して、「インターネット」のダイアログを表示してください。
5) 「権限の編集」タブで、「署名済みコンテンツ」－「署名済みコンテンツの実行」をダイアログを表示するにしてください。

6.3. アプレットタグ

HTML に記述するアプレットタグの表記は Netscape と Microsoft Internet Explorer では異なります。

以降で示すタグの URL で、IP アドレスの代わりにサーバー名を記述した場合、クライアントに DNS の設定が必要となる場合があります。

6.3.1. NSN でのアプレットタグ

```
<applet code=Glp/GlpDownload/NsnPostExcel.class
 archive=https://www8.tokiomarine.co.jp/www16/Glp/NsnPostExcel.jar
 codebase= https://www8.tokiomarine.co.jp/www16/Glp width=0 height=0>
```

applet code : アプレットを実行する為の最初に呼ばれる Main()、Init()等の関数があるクラスを指定
archive : アプレットの JAR ファイルを指定
codebase : アプレットのクラスを置いてある基準となる URL を指定

6.3.2. MSIE でのアプレットタグ

```
<applet code=Glp/GlpDownload/MsiePostExcel.class width=0 height=0>
<PARAM name=CABBASE value= /GlpMsiePostExcel.cab>
```

applet code : アプレットを実行する為の最初に呼ばれる Main()、Initi()等の関数があるクラスを指定
CABBASE value : アプレットの CAB ファイルを指定

7. セキュリティの警告ダイアログ

7.1. NSN のセキュリティ警告ダイアログ

NSN の場合、以下のような警告ダイアログが特権を求めている数だけ（4回）表示されます。
全てにおいて、許可を選択してください。
注）ここで、「いいえ」を選択すると、アプレットは Servlet にレスポンスを返して終了します。

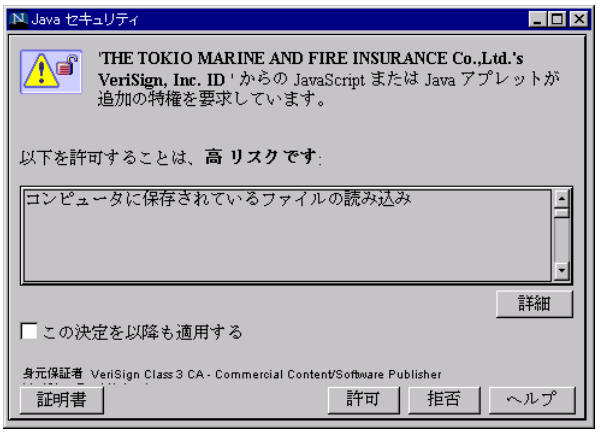

図 3　　NSN セキュリティ警告ダイアログ

7.2. MSIE のセキュリティ警告ダイアログ

MSIE の場合、以下のような警告ダイアログが表示されます。
「はい」を選択して、アプレットを実行してください。
注）ここで、「いいえ」を選択すると、まだアプレットが実行していない状況なので、6.2ブラウザの設定 で説明したのと同様に全画面に戻ることもできず、「しばらくお待ちください」の画面のままとなります。

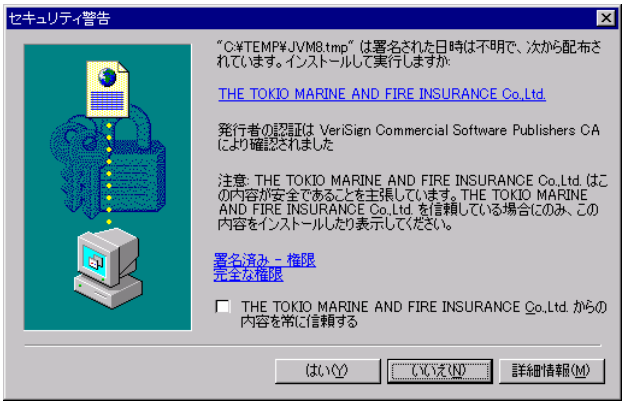

図 4　　MSIE セキュリティ警告ダイアログ

8. 関連コマンドリファレンス

■cabarc コマンド

```
Usage: CABARC [options] command cabfile [@list] [files] [dest_dir]

Commands:
   L   List contents of cabinet (e.g. cabarc l test.cab)
   N   Create new cabinet (e.g. cabarc n test.cab *.c app.mak *.h)
   X   Extract file(s) from cabinet (e.g. cabarc x test.cab foo*.c)

Options:
  -c   Confirm files to be operated on
  -o   When extracting, overwrite without asking for confirmation
  -m   Set compression type [LZX:<15..21>|MSZIP|NONE], (default is MSZIP)
  -p   Preserve path names (absolute paths not allowed)
  -P   Strip specified prefix from files when added
  -r   Recurse into subdirectories when adding files (see -p also)
  -s   Reserve space in cabinet for signing (e.g. -s 6144 reserves 6K bytes)
  -i   Set cabinet set ID when creating cabinets (default is 0)

Notes
-----
When creating a cabinet, the plus sign (+) may be used as a filename
to force a folder boundary; e.g. cabarc n test.cab *.c test.h + *.bmp

When extracting files to disk, the <dest_dir>, if provided, must end in
a backslash; e.g. cabarc x test.cab bar*.cpp *.h d:¥test¥

The -P (strip prefix) option can be used to strip out path information
e.g. cabarc -r -p -P myproj¥ a test.cab myproj¥balloon¥*.*
The -P option can be used multiple times to strip out multiple paths
```

■makecert コマンド

```
Usage: MakeCert [ basic|extended options] [outputCertificateFile]

 -sk  <keyName>      Subject's key container name; To be created if not present
 -ss  <store>        Subject's certificate store name that stores the output
                     certificate
 -sr  <location>     Subject's certificate store location.
                        <CurrentUser|LocalMachine>.  Default to 'CurrentUser'
 -#   <number>       Serial Number from 1 to 2^31-1.  Default to be unique
 -$   <authority>    The signing authority of the certificate
                        <individual|commercial>
 -n   <X509name>     Certificate subject X500 name (eg: CN=Fred Dews)
 -?                  Return a list of basic options
 -!                  Return a list of extended options
```

■cert2spc コマンド

Usage: Cert2Spc {cert1.cer|crl1.crl ... certN.cer|crlN.crl} output.spc

■signcode コマンド

Usage: SignCode [options] FileName

```
-spc  <file>       Spc file containing software publishing certificates
-v    <pvkFile>    Pvk file name containing the private key
-k    <KeyName>    Key container name
-n    <name>       Text name representing content of the file to be signed
-i    <info>       Place to get more info on content (usually a URL)
-p    <provider>   Name of the cryptographic provider on the system
-y    <type>       Cryptographic provider type to use
-ky   <keytype>    Key type
                      <signature|exchange|<integer>>
-$    <authority>  Signing authority of the certificate
                      <individual|commercial>
                      Default to using certificate's highest capability
-a    <algorithm>  Hashing algorithm for signing
                      <md5|sha1>. Default to md5
-t    <URL>        TimeStamp server's http address
-tr   <number>     The # of timestamp trial until succeeds. Default to 1
-tw   <number>     The # of seconds delay between each timestamp. Default to 0
-j    <dllName>    Name of the dll that provides attributes of the signature
-jp   <param>      Parameter to be passed to the dll
-c    <file>       X509 file containing encoded software publishing certificate
-s    <store>      Cert store containing certs. Default to my store
-r    <location>   Location of the cert store in the registry
                      <localMachine|currentUser>. Default to currentUser
-sp   <policy>     Add all the certificates in the chain or add until one cert
                   in the chain is from the spc store.
                      <chain|spcstore>. Default to spcstore
-cn   <name>       The common name of the certificate
-x                 Do not sign the file.  Only Timestamp the file
```

Note: To sign with a SPC file, the required options are -spc and -v if your private key is in a PVK file. If your private key is in a registry key container, then -spc and -k are the required options.

■signtool コマンド

```
Usage:  signtool [options] directory-tree
    -b"basename"              basename of .sf, .rsa files for signing
    -c#                       Compression level, 0-9, 0=none
    -d"certificate directory" contains cert*.db and key*.db
    -e".ext"                  sign only files with this extension
    -f"filename"              read commands from file
    -G"nickname"              create object-signing cert with this nickname
    -i"installer script"      assign installer javascript
    -j"javascript directory"  sign javascript files in this subtree
    -J                        directory contains HTML files. Javascript
                                  will be extracted and signed.
    -k"cert nickname"         sign with this certificate
    --leavearc                do not delete .arc directories created
                                  by -J option
    -m"metafile"              include custom meta-information
    --norecurse               do not operate on subdirectories
    -o                        optimize - omit optional headers
    --outfile "filename"      redirect output to file
    -p"password"              for password on command line (insecure)
    -s keysize                keysize in bits of generated cert
    -t token                  name of token on which to generate cert
    --verbosity #             Set amount of debugging information to generate.
                              Lower number means less output, 0 is default.
    -x"name"                  directory or filename to exclude
    -z                        omit signing time from signature
    -Z"jarfile"               create JAR file with the given name.
                              (Default compression level is 6.)

signtool -l
  lists the signing certificates in your database
signtool -L
  lists all certificates in your database, marks object-signing certificates
signtool -M
  lists the PKCS #11 modules available to signtool
signtool -v file.jar
  show the contents of the specified jar file
signtool -w file.jar
  if valid, tries to tell you who signed the jar file
For more details, visit
  http://developer.netscape.com/library/documentation/signedobj/signtool/
```