Handling Web Logins

By David Adams

TN 06-39

Overview

4th Dimension の Web サーバは、On Web Authentication データベースメソッドという形で HTTP パスワードをサポートしています。メソッドが存在し、設定されていれば、メソッドの実行その他のリクエストに対し、自動的にパスワードの入力要求がブラウザに送信されます。このシステムの制限は、仮にパスワード入力ダイアログがユーザによりキャンセルされた場合、ブラウザには空白ページが表示されるという点です。そのような画面はエンドユーザを当惑させ、サイトが消えてしまったかのような印象さえ与えるかもしれません。空白ページよりは、ヘルプテキスト、ログインのヒント、他のページへのリンクなど、関係する情報が表示されたほうが親切です。On Web Authentication データベースメソッドのこうしたデフォルト動作を回避することはそれほど難しくはありません。自動アクションに頼る代わりに数行のコードを On Web Connection および 4D ACTION でコールされるメソッドに追加するだけでカスタム認証をすることができるからです。

このテクニカルノートでは、On Web Authentication システムの動作の理由を説明した上で Web パスワードシステムの代替案を紹介しています。On Web Authentication の動作を説明するために Web_Login_Default、カスタムソリューションの例として Web_Login_Custom というサンプルデータベースが収録されています。

About Web Passwords

On Web Authentication の動作を理解するためには、HTTP パスワードの仕組みから考えなければなりません。パスワードで保護された Web サイトでパスワードダイアログをキャンセルした場合の動作を考慮してみましょう。ユーザの目には、次のような処理に映るかもしれません：

1. ブラウザがページをリクエストする。
2. サーバはユーザ名とパスワードの入力を求める。
3. パスワードダイアログをキャンセルした結果、サーバからエラーメッセージが返される。

しかし、実際には次のように処理されています：

1. ブラウザがページをリクエストする。
2. サーバは HTTP エラー401 とエラーページを送信する。

3. ブラウザはエラー401 に反応し、ユーザ名とパスワードが必要であることを認識する。
4. ブラウザは情報を取得するためにダイアログを表示する。
5. ユーザによりパスワードダイアログがキャンセルされた結果、ブラウザは先ほどサーバから送られたエラーページを表示する。

どちらもあまり変わらないように思えるかもしれません。次のように強調して表現すれば、肝心な相違点が浮き彫りになるでしょう：

- パスワードダイアログを表示し、制御しているのはサーバではなくブラウザです。そのため、違うブラウザを使用すれば、表示されるダイアログの外観も異なります。

- ブラウザによって表示されるパスワードダイアログをキャンセルした結果、サーバにメッセージが送られることはありません。このとき表示されるべきエラーページは、キャンセルの時点ですでにブラウザ側に送られていなければなりません。

以上の点を踏まえた上で 4th Dimension の動作を振り返ってみましょう。

4th Dimension's Default Behavior

On Web Authentication データベースメソッドは、Web 認証コードを記述する場所です。メソッドは、URL あるいはセミダイアナミックコールバックにより 4D メソッドがコールされるたびに実行されます。On Web Authentication データベースメソッドのパラメータには Web リクエストに関係のある 6 種類のテキストデータが自動的に代入されます。リクエストを許可するには$0 に True を返します。ブラウザにパスワードの入力を求めるには$0 に False を返します。次のサンプルコードには、同メソッドにおける処理の流れが示されています：

```
C_BOOLEAN($0;$acceptConnection_b)
C_TEXT($1)  ` ;$url_t)
C_TEXT($2)  `;$httpHeader_t)
C_TEXT($3)  `;$clientIPAddress_t)
C_TEXT($4)  `;$serverIPAddress_t)
C_TEXT($5;$userName_t)
C_TEXT($6;$password_t)

$userName_t:=$5
$password_t:=$6

If (WebLoginIsOkay($userName_t;$password_t)
   $acceptConnection_b:=True
Else
   $acceptConnection_:=False
End if

$0:=$acceptConnection_b
```

$0 に False を返した場合、4th Dimension はリクエストの処理を終了し、401 Unauthorized ステータスの HTTP 応答を返します。応答には WWW-Authenticate の HTTP ヘッダも含まれ、

HTTP パスワードチャレンジレスポンスと呼ばれるものになっています。このときの HTTP 応答ヘッダは次のようなものです：

```
HTTP/1.1 401 Authorization Required.
Server: 4D_WebStar_D/2004
Date: Tue, 10 Oct 2006 00:21:20 GMT
WWW-Authenticate: Basic realm="Web_Login_Default.4DB"
```

ステータス 401 の応答には、パスワードダイアログがキャンセルされた場合に表示されるページも含めることができるのですが、On Web Authentication が返す応答にはページが含まれていません。パスワードダイアログがキャンセルされた場合、空白のページが表示されるのはそのためです。この動作そのものを改めることはできないのですが、いったんリクエストを On Web Authentication で受け入れた後、On Web Connection または 4DACTION でコールされたメソッドの中で認証を処理すれば、結果的に問題を回避することができます。

Implementing a Custom Web Password Challenge

Overview

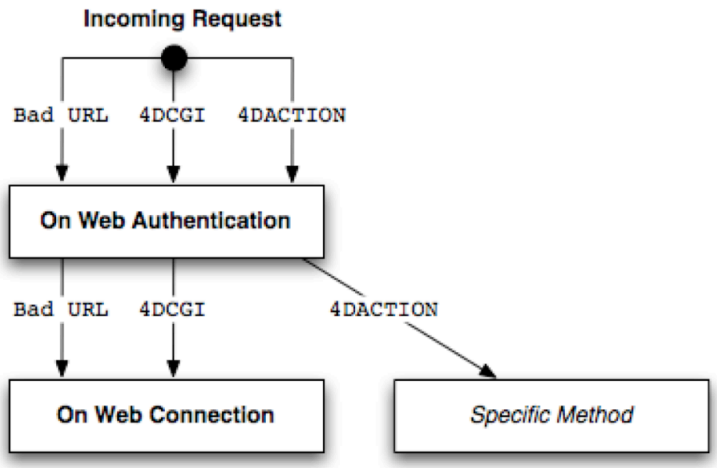
4th Dimension でカスタマイズされた Web パスワード認証システムを導入することは、それほど難しくはありません。作成しなければならないコードは、次のふたつだけです：

1. パスワードを評価するコード。
2. 完全な HTTP 応答エラー401 を送信するコード。

データベースで Web が使用されている仕方により、何種類かの方法が考えられます。このテクニカルノートに収録されているデータベースは、Web ページリクエスト、4DCGI、4DACTION を含むすべての非コンテキストモードアクセスをパスワード保護するサンプルです。

4D Web Server Flow of Control

次の図に示されているように、4D Web の内部的な処理の流れは、同じ非コンテキストモードであっても、コールの種類によっていくらか異なっています：



いずれの場合も On Web Authentication が実行され、4DACTION では On Web Connection が実行されないのがポイントです。共通のコードを置く場所には On Web Authentication が最適なのですが、今回は On Web Authentication の制限を回避するのが目的である以上、それはできません。On Web Connection を実行するコールはそこで認証を処理することができますが、4DACTION でコールされるメソッドについては、それぞれのメソッドで処理する必要があります。どちらの処理も簡単であり、Web_Login_Custom サンプルにはその例が含まれています。では、それぞれの場所に記述するコードを考慮することにしましょう。

Customizing On Web Authentication

カスタマイズされた Web パスワード認証を使用するためには On Web Authentication を修正し、暫定的にリクエストを許可するとともに、ユーザ名とパスワードを後で評価するために保存する仕組みを設けなければなりません。

```
C_BOOLEAN($0;$acceptConnection_b)
C_TEXT($1;$2;$3;$4;$5;$6)

` 4DACTIONメソッドで評価するためにユーザ名とパスワードを保存

C_TEXT(WebDemo_UserName_t)
C_TEXT(WebDemo_UserPassword_t)
WebDemo_UserName_t:=$5
WebDemo_UserPassword_t:=$6

$0:=True
```

True を返すことにより、4th Dimension のパスワード認証は通過することになります。あらゆる Web リクエストが許可される以上、コードで実際の認証を処理することが必要です。とはいえ、前述のようにそのコードを使用するのは 2 カ所、つまり On Web Connection および 4DACTION で直接コールされるそれぞれのメソッドだけです。4DACTION を 4DCGI で代用しているサイトの場合、On Web Connection にコードを追加するだけで十分です。

Tip　4DACTION が 4DCGI に勝ることは何もないと言ってよいでしょう。Web に実際のメソッド名が流出してしまうことは、セキュリティ面での不安は別にしても、メソッド名を変えれば外部リンクがすべて切れてしまうという意味で大きなデメリットです。

Customizing On Web Connection

On Web Connection メソッドは、未知の URL、あるいは 4DCGI コールを受け入れるたびに実行されます。Web_Login_Custom サンプルでは次のようなコードが使用されています：

```
C_TEXT($1;$url_t)
C_TEXT($2)  `;$httpHeader_t)
C_TEXT($3)  `;$clientIPAddress_t)
C_TEXT($4)  `;$serverIPAddress_t)
C_TEXT($5;$userName_t)
C_TEXT($6;$password_t)

$url_t:=$1
$userName_t:=$5
$password_t:=$6

C_BOOLEAN($loginOkay_b)
$loginOkay_b:=True

If (WebLoginIsRequired ($url_t))
   $loginOkay_b:=WebLoginIsOkay ($userName_t;$password_t)
   If (Not($loginOkay_b))
      WebLoginSendChallenge
   End if
End if


If ($loginOkay_b)

   Case of
      ¥ ($url_t="/protected.html")
         SEND HTML FILE("protected/page_reached.html")

      ¥ ($url_t="/4DCGI/CallMethodWith4DCGI@")
         MethodCalledBy4DCGI

      Else
         C_TEXT(WebDemo_RequestedURL_t)
         WebDemo_RequestedURL_t:=$1
         SEND HTML FILE("not_found.html")
   End case


End if
```

主要な処理は WebLoginIsRequired、WebLoginIsOkay、WebLoginSendChallenge というメソッドにまとめられています。いずれも簡潔でとてもシンプルなメソッドです。

The WebLoginIsRequired Method

多くの Web サイトには、一般に公開される部分と限定的に公開される部分で構成されているものです。WebLoginIsRequired メソッドは、特定の URL が認証を必要としているか否かを判別するための共通ルーチンとして使用されることを想定しています。内部的には、条件分岐によって保護されたリソースに対するアクセスを見定めています：

```
C_BOOLEAN($0;$loginRequired_b)
C_TEXT($1;$url_t)

$url_t:=$1
$loginRequired_b:=False

Case of
  ¥ ($url_t="/protected@")  ` protectedディレクトリ以下は認証が必要
    $loginRequired_b:=True

  ¥ ($url_t="/4DCGI@")  ` 4DCGIは認証が必要
    $loginRequired_b:=True

  ¥ ($url_t="/4DACTION@")  ` 4DACTIONは認証が必要
    $loginRequired_b:=True

  Else
    $loginRequired_b:=False
End case
```

サンプルデータベースのコードは骨組みにすぎません。データベースの必要にあわせて自由にコードを拡張することができます。

The WebLoginIsOkay Method

サンプルデータベースの WebLoginIsOkay メソッドは、実質的には効力のないスタブであり、Web で受け取ったユーザ名とパスワードをハードコーディングされた値と単純に比較しているだけですが、大事なのはメソッドが置かれている場所であり、すべての認証がここに集約されているという点です。メソッドのもっとも基本的な形は次のようになります：

```
C_BOOLEAN($0;$loginIsOkay_b)
C_TEXT($1;$userName_t)
C_TEXT($2;$password_t)

$loginIsOkay_b:=False

$userName_t:=$1
$password_t:=$2

If (($userName_t="guest") & ($password_t="4D"))
  $loginIsOkay_b:=True
End if

$0:=$loginIsOkay_b
```

実際のデータベースでは、上記の形を拡張し、引数の値をレコードや外部ドキュメントに保存されたユーザ名またパスワードと比較することができます。

一切、4DACTION を使用していなければ、メソッドは上記のままでも動作します。4DACTION が使用されている場合、ユーザ名とパスワードは On Web Authentication から何らかの方法で継承できるようにする必要があります。これにはいくつかの方法が考えされますが、今回のサンプルデータベースでは On Web Authentication でユーザ名とパスワードをプロセス変数に代入し、パラメータなしで WebLoginIsOkay がコールされた場合にはそれらの変数を調べるようにしています。拡張されたコードは次のような感じになります：

```
$loginIsOkay_b:=False

If (Count parameters=2)
   $userName_t:=$1
   $password_t:=$2
Else

   ` ユーザ名とパスワードはOn Web Authenticationから継承
   If (Undefined(WebDemo_UserName_t)) ` あり得ないこと
      WebDemo_UserName_t:=""
   End if
   If (Undefined(WebDemo_UserPassword_t)) ` あり得ないこと
      WebDemo_UserPassword_t:=""
   End if

   $userName_t:=WebDemo_UserName_t
   $password_t:=WebDemo_UserPassword_t

End if

If (($userName_t="guest") & ($password_t="4D"))
   $loginIsOkay_b:=True
End if

$0:=$loginIsOkay_b
```

このようにすれば、On Web Connection を実行するリクエスト、また 4DACTION でコールされたメソッドの両方でパスワードの認証をすることができるようになります。パラメータの値は、いずれにしても On Web Authentication でプロセス変数に代入されることを考えると、次のようにコードが短縮できるかもしれません：

```
C_BOOLEAN($0;$loginIsOkay_b)
C_TEXT($1)
C_TEXT($2)

$loginIsOkay_b:=False

   ` ユーザ名とパスワードはOn Web Authenticationから継承
If (Undefined(WebDemo_UserName_t)) ` あり得ないこと
   WebDemo_UserName_t:=""
End if
If (Undefined(WebDemo_UserPassword_t)) ` あり得ないこと
   WebDemo_UserPassword_t:=""
End if

If ((WebDemo_UserName_t="guest") & (WebDemo_UserPassword_t="4D"))
   $loginIsOkay_b:=True
End if

$0:=$loginIsOkay_b
```

The WebLoginSendChallenge Method

WebLoginSendChallenge メソッドは複雑なように見えるかもしれませんが、要するに HTTP パスワードチャレンジを促すために必要な HTTP ヘッダを作成し、エラーページのテキストを添えているだけです。メソッド全文を以下に転載します：

```
ARRAY TEXT(WebDemo_HttpHeaderNames_at;3)
ARRAY TEXT(WebDemo_HttpHeaderValues_at;3)
WebDemo_HttpHeaderNames_at{1}:="X-VERSION" ` 必ずこれが第一要素
WebDemo_HttpHeaderValues_at{1}:="1.1"

WebDemo_HttpHeaderNames_at{2}:="X-STATUS" ` 必ずこれが第二要素
WebDemo_HttpHeaderValues_at{2}:="401" ` エラーコード

C_TEXT($realm_text)
 ` パスワードを求めるために必要
 ` realm名は任意で変更して構わない
$realm_text:="" ` 例: Basic realm="Web Login Demo"
$realm_text:=$realm_text+"Basic realm="
$realm_text:=$realm_text+Char(Double quote )
$realm_text:=$realm_text+"Web Login Demo"
$realm_text:=$realm_text+Char(Double quote )

WebDemo_HttpHeaderNames_at{3}:="WWW-Authenticate"
WebDemo_HttpHeaderValues_at{3}:=$realm_text

SET HTTP HEADER(WebDemo_HttpHeaderNames_at;WebDemo_HttpHeaderValues_at)
SEND HTML FILE("password_challenge.html")
```

以下の HTTP/HTML は、上記コードにより出力される結果を示しています。<!DOCTYPE HTML 以降が password_challenge.html ファイルの内容です。ヘッダ部分は強調のためハイライトされています：

HTTP/1.1 401 Authorization Required.
Server: 4D_WebStar_D/2004
Date: Tue, 10 Oct 2006 02:28:11 GMT
WWW-Authenticate: Basic realm="Web Login Demo"
Connection: close
Last-Modified: Tue, 10 Oct 2006 02:28:11 GMT
Expires: Wed, 11 Oct 2006 02:28:11 GMT
Content-Type: text/html;Charset=Shift_JIS
Content-Length: 1092

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
        "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
 <title>パスワードが必要です。</title>

 <link
  rel="Stylesheet"
  type="text/css"
```

```
   title="Styles for 4D Web Log-in Examples"
   href="/styles.css"
   rev="Stylesheet">

</head>
<body>
<h1>ユーザ名とパスワードが必要です。</h1>
<p>要求されたメソッドまたはページにアクセスするためには、ユーザ名とパスワードが必要です。下図を参考にして、有効な ユーザ名とパスワードをブラウザが表示するウインドウに入力してください:</p>

<img
 src="password_help.jpg"
 alt="Password dialog"
 width="527"
 height="385">

<p><b>ヒント</b>: ユーザ名は <span class="html_text">guest</span> パスワードは <span class="html_text">4D</span>です。ユーザ名とパスワードが認証を通過した場合、一般的なブラウザは毎回のリクエストで同じ情報を送信するように設定されています。そのため、ブラウザを終了するまでは、毎回ユーザ名とパスワードを入力しなくても、アクセスを続けることができるはずです。</p>

<p><a href="/index.html">Home</a></p>

</body>
</html>
```

Authenticating from within 4DACTION Methods

サンプルデータベースのメソッドは、On Web Connection データベースメソッドおよび個別のプロジェクトメソッドのどちらからでもコールできる認証メソッドとして作成されています。処理の流れを示した前出の図から分かるように、4DACTION では On Web Authentication の後、目的のメソッドに処理が移ります。サンプルでは、ユーザ名とパスワードをプロセス変数に代入し、WebLoginIsOkay メソッドで取得するようになっています。このような仕組みなので、個別のプロジェクトメソッドでも次のようなコードで簡単に認証を扱うことができます：

```
C_TEXT($0;$1)  `4DACTIONでコールされたメソッド

If (WebLoginIsOkay =False)
   WebLoginSendChallenge

Else   ` 認証は成功

   SEND HTML FILE("protected/method_ran_through_4daction.html")

End if
```

On Web Authentication Versus the Custom Solution

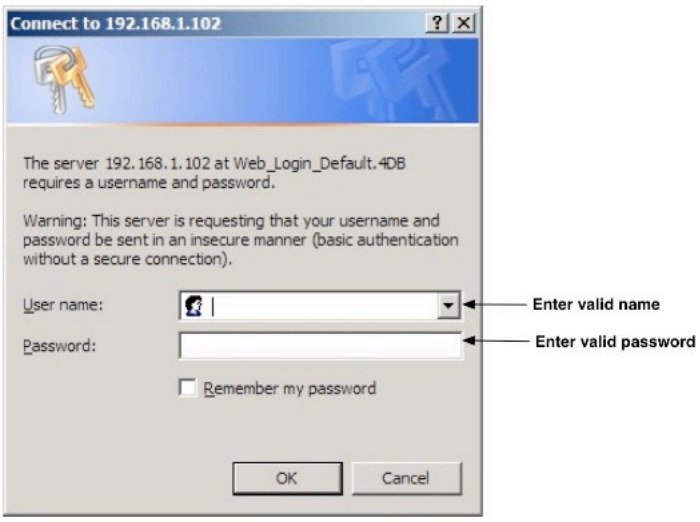
前述の HTTP ヘッダ出力は、機能的には On Web Authentication のパスワードチャレンジで返されるヘッダと同じです。関係ある部分を比較のため並べてみると次のようになります：

On Web Authentication	**Custom Solution**
HTTP/1.1 401 Authorization Required.	HTTP/1.1 401 Authorization Required.
Server: 4D_WebStar_D/2004	Server: 4D_WebStar_D/2004
WWW-Authenticate: Basic realm="Web_Login_Default.4DB"	WWW-Authenticate: Basic realm="Web Login Demo"

違っているのは realm の部分だけです。カスタムソリューションのほうでは realm 名を自由に設定することができるのに対し、On Web Authentication では自動的にデータベーストラクチャ名に基づいた realm 名が使用されます。この差は、HTTP の続く部分と関係があります。On Web Authentication の場合、続く部分には何もありません。カスタムソリューションでは、Web ページの情報が続いています。ブラウザのパスワードダイアログがキャンセルされた場合、パスワードチャレンジに含まれていたページ、サンプルデータベースであれば次のようなエラーページが表示されることになります：

ユーザ名とパスワードが必要です。

要求されたメソッドまたはページにアクセスするためには、ユーザ名とパスワードが必要です。下図を参考にして、有効な ユーザ名とパスワードをブラウザが表示するウインドウに入力してください:



ヒント: ユーザ名は guest パスワードは 4Dです。ユーザ名とパスワードが認証を通過した場合、一般的なブラウザは毎回のリクエストで同じ情報を送信するように設定されています。そのため、ブラウザを終了するまでは、毎回ユーザ名とパスワードを入力しなくても、アクセスを続けることができるはずです。

Home

Additional Notes on Web Passwords

Web Passwords Are Not Secure

このことはいくら強調しても強調しすぎるということはないでしょう。Web パスワードはセキュリティに優れたシステムではありません。これは欠陥ではなく、HTTP がそのように設計されていることに起因しています。ユーザ名とパスワードは結合され、bse64 エンコードで変換されたものが送信されます。たとえばユーザ名が guest でパスワードが 4D の場合、次のような文字列が送信されます：

Authorization: Basic Z3Vlc3Q6NEQ=

Z3Vlc3Q6NEQ=という文字列は、guest:4D という文字列を単純に base64 エンコードしたものです。HTTP ベーシック認証では、暗号化と呼べるような処理は何もしていません。認証を通過した後、ネットワークを経由して送信されるページやドキュメントも無防備です。セキュリティが問題であれば、SSL(HTTPS)を使用してください。

Do Not Use 4D Passwords Over the Web

4th Dimension の Web サーバでは、環境設定の Web>詳細ページでパスワード関係のオプションをいくらか設定することができます：

パスワード使用オプションを選択すると、4D パスワードを含むオプションも選択できるようになります。このオプションが有効にされている場合、4th Dimension は Web リクエストのユーザ名およびパスワードを 4D のユーザ名およびパスワードと比較します。Web パスワードはそれほどセキュリティが高くない点を考えると、通常、このオプションを使用することはあまり勧められません。特に 4D Server を Web で公開している場合、4th Dimension のユーザ名とパスワードが分かればどこからでも 4D Client で接続できるので危険です。

Note　このテクニカルノートで取り上げている Web パスワードシステムを実践するためには、パスワード使用オプションが有効にされている必要があります。

Browsers Remember Web Passwords

Web はステートのないプロトコルであり、リクエストからリクエストへ情報が継承されることはありません。パスワード保護された Web サイトにアクセスする場合、本来ならばリクエストのたびにパスワードを送信しなければなりません。しかしながら実際には最初の認証を通過した後は自由にブラウズできたほうが自然であり、快適な使用感を実現するためにブラウザが二度目からはパスワードを求めない場合がほとんどです。内部的にはブラウザがユーザ名とパスワードを記憶し、リクエストのたびに送信しています。Web パスワードを自分で処理する場合、その点を念頭に置くようにしてください。テストの過程では、ブラウザを終了して再起動することも必要です。あるいは複数のブラウザを開きながらテストするという方法もあります。

Tip　http://chrispederick.com/work/webdeveloper/で入手することができる FireFox の Web Developer エクステンションを使用すれば、HTTP パスワードをいつでもクリアすることができます。オプションの場所は Miscellaneous > Clear Private Data > Clear HTTP Authentication です。

Case-Sensitivity Is Optional

ユーザ名とパスワードで大文字と小文字を区別するか否かは各 Web サイト運営者が判断することです。4th Dimension の世界では、大文字と小文字は区別しないほうが楽かもしれません。4th Dimension で大文字と小文字の区別をする方法は、テクニカルノート 05-41「大文字と小文字を判別して処理する」の中で取り上げられています。収録されているサンプルデータベースには CS_AlphasAreEqual など、汎用的なメソッドがいくつも含まれています。

Summary

4th Dimension の Web サーバは、On Web Authentication データベースメソッドという形で HTTP パスワードをサポートしていますが、ブラウザのパスワード入力ダイアログがキャンセルされたときのためにエラーページを送信することはしません。とはいえこの制限は、On Web Connection データベースメソッドおよび 4DACTION で直接コールされるメソッドに少しのカスタムコードを追加することで簡単に回避することができます。