Web Access User Validator

User Guide

Version 1.1

Revisions

1.0	Initial release
1.1	Bug fix for "Session started by another user" error Minor doc updates

Table of Contents

What is the Web Access User Validator?

The Web Access User Validator (WWWVALID) is a service program which contains a number of procedures which can be added to any existing CGI program, to provide user profile/password (credential) validation.

Put simply, including WWWVALID in your CGI applications allows you to define several options to provide enhanced application security:

- An application-specific sign-on page that is automatically displayed when the user calls a given CGI program (by submitting a page from a browser)
- User-defined credential validation, allowing you to define your own userid's and sign-on control
- Application timeout processing (per-page and per-session), so a user is forced to complete a transaction or submit a given page within a specified time limit
- Session-specific cookies, to ensure complete end-to-end session management
- The option to swap to run under the user profile used to sign-on, so the CGI program can run with advanced authorities

These additional options can be easily added to your existing CGI programs with only a few lines of code, and with no changes to any HTTP server configuration.

Why should I use the Web Access User Validator?

Currently, if you want application security for your IBM i CGI applications, you have a number of very limited options that you can use to control who can call your CGI programs:

1. Use the `UserID %%CLIENT%%` HTTP configuration directive
2. Specific hand-written security added to your CGI programs

In the first case, the user is presented with a sign-on pop-up, where they must enter a valid IBM i user profile and password, which are validated by the system. Whilst this functionality works in many cases, it has a number of drawbacks:

a) The validation pop-up is just that – a basic (ugly!) pop-up – it's not a true sign-on page, which might have a corporate look-and-feel, with a company logo, links, images etc.
b) The user must enter an actual IBM i user profile and password. This is fine for intranet use, but if you want to make your CGI application available on the internet, you're probably not going to want either to create individual user profiles for each possible user or to use a single user profile for all possible users.

c) The HTTP server will actually swap to use that user profile when running the CGI program. This might be what you want, but it might not, and you don't get the choice.

In the second case, you have to write your own credential validation processing. Obviously you can do this and add whatever additional processing you want, but it can be a lot of work for you. Additionally, the CGI program normally then has access to the password and user profile used to sign on, so you need to be sure that the CGI program is entirely secure.

However, if you use the Web Access User Validator, you can have the best of both worlds - an application-specific sign-on page, which automatically includes timeout processing for your CGI programs, without either the requirement either to substantially change any of your existing CGI programs or to write your own transaction and credential validation processing. Additionally, the user profile and password used to sign on are stored in an encrypted format and are 'insulated' from your CGI program, which can only access the user profile using a specific procedure and cannot access the password in any way.

How do I use the Web Access User Validator?

After installing the Web Access User Validator, simply add a call to the `www_validate` procedure as the first bit of processing in your CGI program, as the following example shows:

```
www_sessionid = www_validate();
if www_sessionid = *blanks;
  return;
endif;
...your CGI program code...
```

That's all there is to it!

Simply by adding these four lines of code, when the CGI program is first called from the browser, a sign-on page is displayed, where the user must enter a valid IBM i user profile and password. If the credentials are invalid (for instance, if the password is incorrect), the sign-on page is redisplayed with an appropriate error message, and the user must re-enter their credentials. If the credentials are valid, the CGI program continues its processing just as if you had not added the call to `www_validate`. On subsequent calls to the CGI program, the sign-on page is not displayed.

However, you may want to add more functionality - perhaps you want to ensure that the session times-out after a few minutes. To do this, simply pass a parameter to `www_validate`, like this:

```
www_sessionid = www_validate('-sessiontimeout 300');
if www_sessionid = *blanks;
  return;
endif;
...your CGI program code...
```

This time, in addition to the above processing, every time your CGI program is called, `www_validate` will check how long it was since you initially signed on. If it is more than 300 seconds (5 minutes), the sign-on page is redisplayed with a "Session has expired" error message. If the user signs on correctly again, a new session ID is generated by `www_validate`.

If you try out this example using the WWWVALIDT2 sample CGI program, you will see that the default sign-on page is very basic - just a userid field, a password field and a submit button. However, if you wish, you can create your own sign-on page as an HTML page in the IFS (using JavaScript, CSS, background images etc.). To specify that your sign-on page should be used instead of the default sign-on page, you use the `-signonpage` flag to specify the full path to the file, e.g.:

```
www_sessionid = www_validate('-sessiontimeout 300 -signonpage /pages/sign-on.html');
if www_ sessionid = *blanks;
  return;
endif;
...your CGI program code...
```

As you can see, you can specify multiple options in the parameter to `www_validate`. There are other options which allow you to specify a page timeout (a per-page timeout, either in addition to or instead of a session timeout), a cookie option to specify whether a single cookie will be used for the session, and more. See the Valid www_validate keywords section for details.

Note that, unlike the `UserID %%CLIENT%%` HTTP server configuration directive, if a user enters a valid IBM i user profile and password in the sign-on page, those credentials are not actually used by the CGI application - they are checked to ensure that they are valid, but the CGI program still runs using the default HTTP server user profile QTMHHTP1. However, you can choose to run the CGI program using the specified user profile, by calling the `www_swapusrprf` procedure (see below).

After the user has signed on, the CGI program can subsequently retrieve the user profile used to sign on using the `www_getusrprf` procedure (the password used to sign on is not available in any way to the CGI program).

Because an individual page request coming in from the browser might get routed by the IBM i to any one of a number of HTTP server CGI jobs, the information the Web Access User Validator stores about sessions is held in a user space in the same library as the WWWVALID service program. This user space is created if it does not exist. All credential information held in the user space is encrypted using RC4 stream cipher encryption.

A maximum of 32767 sessions can be active at any one time. If a user closes their browser window without explicitly ending the session, the session 'entry' in the user space is not freed up until the session times-out. If the session was started with no timeout value specified, the session entry becomes permanently unavailable for use, until the WWWVALID user space is deleted. Therefore, it is possible (although extremely unlikely) for all the sessions to eventually become permanently unavailable. In this case, any users who call a CGI program from the browser will receive an error page. If this happens, simply delete the WWWVALID user space, and it will automatically be created in a 'clean' state when the next user calls a CGI program from the browser.

Installing the Web Access User Validator

The Web Access User Validator is supplied as an IBM i save file (*SAVF), saved at V5R4M0 called wwwvalid.savf.

Perform the following steps to install the Web Access User Validator on your IBM i:

1. Copy the wwwvalid.savf file into the C:/Temp folder on your PC.
2. FTP the wwwvalid.savf save file to your machine as follows:
   a) Open a command window on your PC
   b) Type the following commands in turn, pressing Enter after each one. Replace `<machine>` with the name or IP address of your IBM i. Replace `<user-profile>` and `<password>` with the user profile/password of a user with sufficient authority to FTP objects to your machine.

   ```
   FTP <machine>
   <user-profile>
   <password>
   LCD C:/Temp
   CD QGPL
   BIN
   QUOTE SITE NAMEFMT 1
   PUT WWWVALID.SAVF WWWVALID.SAVF
   QUIT
   ```

   c) Check that the WWWVALID save file was successfully transferred to your machine by running the following command:

   ```
   DSPSAVF SAVF(QGPL/WWWVALID)
   ```

3. Restore the WWWVALID library from the save file using the following command:

   ```
   RSTLIB LIB(WWWVALID) DEV(*SAVF) SAVF(QGPL/WWWVALID)
   ```

   The WWWVALID library contains just a single source file called QSRC, which contains the following members:

   ```
   QSRC         *FILE       PF-SRC      Web Access User Validator Source
     ERROR        HTML        HTML source for error page
     LOGIN        HTML        HTML source for login page
     SETUP        CLLE        Web Access User Validator Setup Program
   * WWWVALID     RPGLE       Web Access User Validator
     WWWVALID_P   RPGLE       Web Access User Validator Prototypes
     WWWVALIDEP   RPGLE       Web Access User Validator Example Exit Program
   * WWWVALIDSP   BND         Web Access User Validator
   ```

```
  WWWVALIDT1  RPGLE       Web Access User Validator Test 1 - no WWWVALID
  WWWVALIDT2  RPGLE       Web Access User Validator Test 2
  WWWVALIDT3  RPGLE       Web Access User Validator Test 3
  WWWVALIDT4  RPGLE       Web Access User Validator Test 4
```

***** If you have the WWWVALID 'runtime version', the WWWVALID and WWWVALIDSP source members are not included in the QSRC file and the WWWVALID library will contain a copy of the WWWVALID service program object itself.

4. Compile the SETUP CL program using the following command:

   ```
   CRTBNDCL PGM(WWWVALID/SETUP) SRCFILE(WWWVALID/QSRC) SRCMBR(SETUP)
   ```

5. Call the SETUP set-up program from a command line to create all the WWWVALID objects:

   ```
   CALL PGM(WWWVALID/SETUP)
   ```

   The SETUP program compiles the source members into objects, creates a binding directory and creates a directory called `wwwvalid` in the root directory and copies the ERROR and LOGIN source members into that directory as stream files.

   A completion message is sent when the setup completes successfully.

Note: If you are upgrading from a previous release of WWWVALID, you should still run the above process, to ensure that the latest changes are used.

In order for your CGI programs to use any of the Web Access User Validator procedures, they must be compiled to bind to the WWWVALID service program. To do this, ensure that the WWWVALID binding directory in the WWWVALID library is included in the compilation command, e.g.:

```
CRTBNDRPG PGM(MYCGILIB/MYCGIPGM)...BNDDIR(WWWVALID/WWWVALID)
```

and that your CGI programs include a reference to the WWWVALID_P copybook:

```
 /COPY WWWVALID/QSRC,WWWVALID_P
```

These programs can then successfully use `www_validate` and the other procedures exported from the WWWVALID service program.

No changes are required to any HTTP server configuration to enable you to use the Web Access User Validator with your CGI programs, unless you plan to use the supplied sample test CGI programs (see the Example Web Access User Validator usage section for more details).

Web Access User Validator functionality

All the following template variables and procedure prototypes are found in the WWWVALID_P copybook in the QSRC source file in the WWWVALID library.

Template variables

www_sessionid_t - Session identifier

Definition:

```
D www_sessionid_t...
D                  S              16A   Based(TEMPLATE)
```

The `www_sessionid_t` template variable is used to uniquely define a 'session identifier'. A session starts when a user signs on using the sign-on page displayed by `www_validate`, which then generates a session identifier. The CGI program can then pass the session identifier as a parameter to other Web Access User Validator procedures, to perform additional functionality. A session can last over multiple calls to the same or different CGI programs, as long as they all include an initial call to `www_validate`. The session identifier is internally linked to the cookie value that is used by the client (browser) to ensure continuous session management.

For examples of how to refer to the `www_sessionid_t` template variable, see the example for the `www_validate` procedure below.

Procedures

www_validate() - Validate browser access for user

Parameters:

```
D www_validate     PR                   Like(www_sessionid_t)
D                                       Extproc('www_validate')
D   parms                       1024A   Const Options(*Nopass)
```

Description:

The `www_validate` procedure is used to perform credential validation. It is the core of the Web Access User Validator, and must be the first procedure called in your CGI program, before the CGI program writes any data to the browser. See the Valid www_validate keywords section for details of the parameter keywords.

Return value:

The `www_validate` procedure returns the session identifier (or blanks if the user has not yet successfully signed-on). The session identifier can be passed as an input parameter to other procedures called by the CGI program (or by sub-programs called by that CGI program).

Example:

```
D this_session    S                    Like(www_sessionid_t)

 /free

   this_session = www_validate('-sessiontimeout 300');
   if this_session = *blanks;
     return;
   endif;
   ...your CGI program code...

 /end-free
```

www_endsession() - End the session

Prototype:

```
D www_endsession  PR            10I 0 Extproc('www_endsession')
D   sessionid                         Const Like(www_sessionid_t)
```

Description:

The `www_endsession` procedure is called from a CGI program that previously called `www_validate` to end a session that was started using `www_validate`. If you do not call `www_endsession` from your CGI program, then the session will remain active and the user could potentially sign on again and use the same session, until the session timeout value (if specified) is reached. Calling `www_endsession` is an added security measure that you can use to ensure that sessions cannot be re-used. It also frees up space in the control user space for more sessions - a maximum of 32767 sessions can be active concurrently before the user space must be deleted.

Return value:

The `www_endsession` procedure returns a 0 if it was successful or -1 if an error occurred. If an error occurred, a message is written to the job log.

Example:

```
 /free

   rc = www_endsession(this_session);

 /end-free
```

www_getusrprf() - Retrieve the user profile for the session

Prototype:

```
D www_getusrprf    PR            10A   Extproc('www_getusrprf')
D   sessionid                          Const Like(www_sessionid_t)
```

Description:

The `www_getusrprf` procedure can be called by the CGI program that called `www_validate` (or by any program that the CGI program calls) to return the user profile that was used to sign on at the sign-on page. Using `www_getusrprf` does not swap to run using that user profile, nor does it change anything about the job or the CGI program, which will continue to use the default QTMHHTP1 user profile. To swap to run the CGI program using the specified user profile, use `www_swapusrprf`.

Return value:

The `www_getusrprf` procedure returns the user profile used to sign on or blanks if an error occurred. If an error occurred, a message is written to the job log.

Example:

```
D usrprf          S              10A

 /free

   usrprf = www_getusrprf(this_session);
   dsply ('The user signed on as ' + usrprf);

 /end-free
```

www_swapusrprf() - Swap to the specified user profile

Prototype:

```
D www_swapusrprf   PR            10I 0 Extproc('www_swapusrprf')
D   sessionid                         Const Like(www_sessionid_t)
```

Description:

The `www_swapusrprf` procedure swaps the job to run using the user profile used to sign on. If `www_swapusrprf` is called and the profile is already swapped (because of a prior call to `www_swapusrprf`), the profile remains swapped and no error is sent. If the `www_swapusrprf` procedure is called and an exit program was originally specified which did not return EXITPGM_SYSTEM_VALIDATION, then `www_swapusrprf` will return an error.

Return value:

The `www_swapusrprf` procedure returns a 0 if it was successful or -1 if an error occurred. If an error occurred, a message is written to the job log.

Example:

```
 /free

   rc = www_swapusrprf(this_session);

 /end-free
```

www_resetusrprf() - Swap back to the original user profile

Prototype:

```
D www_resetusrprf...
D                 PR            10I 0 Extproc('www_resetusrprf')
D   sessionid                         Const Like(www_sessionid_t)
```

Description:

The `www_resetusrprf` procedure swaps the job back to using the original user profile under which it was running. If `www_resetusrprf` is called and the profile is not currently swapped (because `www_swapusrprf` has not been called), no profile swapping occurs and no error is sent. If `www_resetusrprf` is called and an exit program was originally specified which did not return EXITPGM_SYSTEM_VALIDATION, then `www_resetusrprf` will return an error.

Return value:

The `www_resetusrprf` procedure returns a 0 if it was successful or -1 if an error occurred. If an error occurred, a message is written to the job log.

Example:

```
 /free

   rc = www_resetusrprf(this_session);

 /end-free
```

Valid www_validate keywords

The following is a list of the keywords that can be passed to `www_validate`. Each keyword must be specified with a matching value, with keywords and values separated by a blank. If a keyword is specified more than once, the value specified for the last instance will be used.

For instance, in the example below, the value used for page timeouts will be 45 seconds, since that was the last value specified for the `-pagetimeout` keyword:

```
'-pagetimeout 30 -sessiontimeout 600 -pagetimeout 45 -cookieoption page'
```

Each keyword is listed below with details about its use. All keywords have default values, which will be used if the keyword is not specified or if an invalid value is specified. If an invalid keyword is specified, a warning message is written to the job log, but no error occurs and `www_validate` will continue.

-pagetimeout value-in-seconds

The `-pagetimeout` keyword specifies the page timeout value - that is, the length of time (in seconds) between calls to the CGI program (which will occur as a result of a webpage being submitted). A value of 0 implies that no page timeout will be used. A negative value is invalid and the default value will be used. If a page timeout occurs, the sign-on page is redisplayed with an appropriate error message. If the user signs on, the existing session identifier continues to be used. Note that the `-sessiontimeout` keyword is checked before the `-pagetimeout` keyword.

Values:

`0` - No page timeout will be used. This is the default value.
`1-65535` - Value in seconds before the page times out.

Example:

```
'-pagetimeout 30'
```

-sessiontimeout value-in-seconds

The `-sessiontimeout` keyword specifies the session timeout value - that is, the total length of time (in seconds) from the time that the session started. A negative value is invalid and the default value will be used. If a session timeout occurs, the session is ended and the sign-on page is redisplayed with an appropriate error message. If the user signs on, a new session identifier is generated by `www_validate`. Note that the `-sessiontimeout` keyword is checked before the `-pagetimeout` keyword.

Values:

`0` - No session timeout will be used. This is the default value.
`1-65535` - Value in seconds before the session times out.

Example:

```
'-sessiontimeout 300'
```

`-signonpage path-to-sign-on-page`

The `-signonpage` keyword specifies the full path to the user-defined sign-on page. If the `-signonpage` keyword is not specified or if the user-defined sign-on page does not exist or cannot be used, the default sign-on page (`/wwwvalid/login.html`) is used. See the Creating a user-defined sign-on page section for details on how to create your own sign-on page.

Values:
`path-to-sign-on-page` – The full path to the user-defined sign-on page, including a leading slash.

Example:

```
'-signonpage /base/pages/sign-on.html'
```

`-errorpage path-to-error-page`

The `-errorpage` keyword specifies the full path to the user-defined error page. If the `-errorpage` keyword is not specified or if the user-defined error page does not exist or cannot be used, the default error page (`/wwwvalid/error.html`) is used. See the Creating a user-defined error page section for details on the circumstances under which the error page will be displayed and on how to create your own error page.

Values:
`path-to-error-page` – The full path to the user-defined error page, including a leading slash.

Example:

```
'-errorpage /base/pages/error.html'
```

`-cookieoption [session|page]`

The `-cookieoption` keyword specifies whether a new cookie value will be generated each time the web page is submitted (and the CGI program is called), or whether a single cookie should be generated for the entire session. Using a new cookie for each page is more secure, but adds slightly more processing overhead, since a unique cookie must be recalculated for every transaction.

Values:

`session` - A single cookie will be generated and used until the session is explicitly ended or until the session expires. This is the default value.
`page` - A new cookie value will be generated for each page

Example:

```
'-cookieoption page'
```

-msgf message-file-name

The `-msgf` keyword specifies the message file from which `www_validate` will retrieve any error messages that will be displayed to the user in place of the 'www_validate-errmsg' string in the sign-on page. The message file must exist in the same library as the WWWVALID service program. If this keyword is not specified, the default WWWVALID message file will be used. If an invalid value is specified, the messages will be retrieved from a hard-coded array in the WWWVALID service program. See the Using the Web Access User Validator in a non-English environment section for more details.

Values:

`message-file-name` - The name of a message file that exists in the same library as the WWWVALID service program.

Example:

```
'-msgf wwwmsgfrn'
```

-exitpgm qualified-exit-program

The `-exitpgm` keyword specifies whether a user-defined exit program should be used to perform validation on the credentials (userid and password) entered on the sign-on page. If the `-exitpgm` keyword is not specified, credentials are assumed to be an IBM i user profile and password and are validated as such. See the Creating a user-defined validation exit program section for details on how to write a validation exit program.

Values:

`library/program` - Specify the library-qualified name of a user-defined exit program

Example:

```
'-exitpgm mylib/myexitpgm'
```

Creating a user-defined sign-on page

The default sign-on page is shipped as the LOGIN member in the QSRC file and it is copied to `/wwwvalid/login.html` during setup. If you want to use your own sign-on page, you must create it as a file in the IFS (either in the `wwwvalid` directory or elsewhere) and then specify the full path to it using the `-signonpage` keyword in the parameter to `www_validate`.

The page you create must contain certain elements within an HTML <form> in order to work, but in addition to these elements, you can add whatever you want – images, JavaScript, CSS, links etc.

The form elements your sign-on page MUST include are as follows:

- An input field of type TEXT with a NAME of 'www_validate-userid', e.g.:

  ```
  <input type="text" size="10" maxlength="10" name="www_validate-userid">
  ```

  If an exit program is not being used, `www_validate` will automatically convert any entered value to upper-case, so no client-side (browser) case conversion is necessary.

  If an exit program is being used, the value passed to the exit program will be exactly as it was entered in this field, with no automatic case conversion, but with leading and trailing blanks removed. In this case, the `maxlength` attribute can be increased to any value up to 128, to ensure that a value longer than 10 characters can be entered.

- An input field of type PASSWORD with a NAME of 'www_validate-passwd', e.g.:

  ```
  <input type="password" size="10" maxlength="10" name="www_validate-passwd">
  ```

  If an exit program is not being used, `www_validate` will automatically convert any entered value to upper-case if QPWDLVL = 0 or QPWDLVL = 1, so no client-side (browser) case conversion is necessary.

  If an exit program is being used or if QPWDLVL = 2 or QPWDLVL = 3, the value passed will be exactly as it was entered in this field, with no automatic case conversion, but with leading and trailing blanks removed. In this case, the `maxlength` attribute can be increased to any value up to 128, to ensure that a value longer than 10 characters can be entered.

- The text string 'www_validate-errmsg', as in the following example:

  ```
  <font color="red"><b>www_validate-errmsg</b></font>
  ```

  When the sign-on page is loaded, if an error message from `www_validate` will be displayed (for instance because the sign-on page is being displayed due to a timeout being exceeded), the 'www_validate-errmsg' string is replaced with the error message text, giving e.g.:

  ```
  <font color="red"><b>Password not specified.</b></font>
  ```

If no error message will be displayed, the 'www_validate-errmsg' string is replaced with blanks when the sign-on page is displayed.

A user-defined sign-on page MAY also contain the following 2 text strings:

- The text string 'www_validate-pagetimeout', as in the following example:

  ```
  You have www_validate-pagetimeout seconds to press Submit on each page.
  ```

  When the sign-on page is loaded, the 'www_validate-pagetimeout' string will be replaced with the value specified for the `-pagetimeout` flag in the call to `www_validate`, giving e.g.:

  ```
  You have 30 seconds to press Submit on each page.
  ```

- The text string ' www_validate-sessiontimeout', as in the following example:

  ```
  This session will timeout in www_validate-sessiontimeout seconds.
  ```

  When the sign-on page is loaded, the 'www_validate-sessiontimeout' string will be replaced with the value specified for the `-sessiontimeout` flag in the call to `www_validate`, giving e.g.:

  ```
  This session will timeout in 600 seconds.
  ```

If you create your own sign-on page, it is suggested that you copy the default `/wwwvalid/login.html` page.

Creating a user-defined error page

The error page is displayed when an error occurs in `www_validate` which would result in the user not being able to sign on. For instance, if the WWWVALID user space can't be accessed, or if the maximum number of users are already signed on, the error page will be displayed, ensuring that users won't simply see the sign-on page repeatedly. The default error page is shipped as the ERROR member in the QSRC file and it is copied to `/wwwvalid/error.html` during setup. If you want to use your own error page, you must create it as a file in the IFS (either in the `wwwvalid` directory or elsewhere) and then specify the full path to it using the `-errorpage` keyword in the parameter to `www_validate`.

The only element your error page MUST include is as follows:

- The text string 'www_validate-errmsg', as in the following example:

  ```
  <h2>www_validate-errmsg</h2>
  ```

When the sign-on page is loaded, if an error message from `www_validate` will be displayed (for instance because the sign-on page is being displayed due to a timeout being exceeded), the 'www_validate-errmsg' string is replaced with the error message text, giving e.g.:

```
<h2>Error in Web Access User Validator.</h2>
```

If no error message will be displayed, the 'www_validate-errmsg' string is replaced with a default 'Error occurred in Web Access User Validator' error message.

If you create your own error page, it is suggested that you copy the default `/wwwvalid/error.html` page.

Creating a user-defined validation exit program

By default, credential validation is controlled using system API's. This means that the user must enter a valid IBM i user profile and password at the sign-on page, and is known in this document as ***system credential validation***. However, you may wish to substitute your own ***user-defined credential validation***, for a number of reasons:

1. You may want to make your CGI application accessible over the internet, and therefore not want to give out a user profile and password for your IBM i to users
2. You may want to allow users to create their own userid/password combination, perhaps including special characters that are not allowed in IBM i user profiles and passwords and allowing longer userid's than the 10 character maximum allowed for IBM i user profiles
3. You may want to have a finer-grained level of control over who can sign on to your CGI application than by user profile

For all these reasons, it may make sense to write your own exit program to provide user-defined credential validation. The user-defined credential validation provided by this program can be either instead of system credential validation or in addition to system credential validation.

To implement your own user-defined credential validation, you must perform the following steps:

1. Write the exit program
2. Specify the exit program in your call to `www_validate` by using the `-exitpgm` keyword.

The source for an example exit program is shipped in the QSRC source file in the WWWVALID library, and is called WWWVALIDEP. You can use this as the basis for your own exit programs.

If the name of an exit program is specified in the parameter to `www_validate`, then it will be called by `www_validate` when the sign-on page is submitted. Depending on the values returned from the exit program, the credentials may be considered valid or invalid or they will be additionally validated in `www_validate` as an IBM i user profile and password, using the system API's.

Exit programs must be written such that they have a standard interface (parameters). Below is the interface for the WWWVALIDEP sample exit program. Your own exit program must have the same parameters.

```
D WWWVALIDEP      PR                  ExtPgm('WWWVALIDEP')
D   userid                     128A   Const
D   passwd                     128A   Const
D   valid                       10I 0
D   errmsg                     100A         Varying
D   rtnusrprf                   10A
```

The first two parameters (userid and passwd) are passed from `www_validate` exactly as they were entered on the sign-on page. The other three parameters are returned from the exit program to `www_validate`, and control the actions that `www_validate` should take.

Each exit program parameter is defined in detail as follows:

userid (User Identifier)

Type: 128-byte character string (128A), Input

The user ID entered on the sign-on page. It can contain any characters, including embedded blanks. This value is passed directly from `www_validate`, without any case conversion, although leading and trailing blanks are removed. The following are examples of valid user ID's:

RORY
MichaelWhite
'''f'''
~%^@54
Ziggy Marley
A really long user name for a really cool guy

passwd (Password)

Type: 128-byte character string (128A), Input

The password entered on the sign-on page. Like the `userid` parameter, it can contain any characters, including embedded blanks. This value is passed directly from `www_validate`, without any case conversion, although leading and trailing blanks are removed.

valid (Credential Validity Flag)

Type: 4-byte integer (10I 0), Output

A return flag indicating the validity of the credentials passed to the exit program. One of the following values must be returned to `www_validate`:

`EXITPGM_INVALID_CREDENTIALS`: The credentials are invalid, and the sign-on page will be redisplayed with the 'www_validate-errmsg' string replaced with the value returned by the exit program in the `errmsg` parameter.
`EXITPGM_VALID_CREDENTIALS`: The credentials are valid, and no further validation is required.
`EXITPGM_SYSTEM_VALIDATION`: The credentials are valid, but will be additionally validated as an IBM i user profile and password after the exit program returns control to `www_validate`.

The `EXITPGM_INVALID_CREDENTIALS`, `EXITPGM_VALID_CREDENTIALS` and `EXITPGM_SYSTEM_VALIDATION` constants are defined in the WWWVALID_P copybook with values of 0, 1 and 2 respectively.

Note that the `www_swapusrprf` and `www_resetusrprf` procedures cannot subsequently be used by your CGI program if the exit program returns a value other than `EXITPGM_SYSTEM_VALIDATION`.

errmsg (Error Message)

Type: 100-byte varying character string (100A Varying), Output

The error message to display on the sign-on page when it is redisplayed. If this parameter is blank, a generic "Invalid credentials" error message will be used. This parameter is ignored unless the `valid` parameter is returned with a value of EXITPGM_INVALID_CREDENTIALS.

rtnusrprf (Returned User Profile)

Type: 10-byte character string (10A), Output

The user profile to link with this session. This may be a valid IBM i user profile, or it may be any other value. The CGI program may subsequently retrieve this value using the `www_getusrprf` procedure.

If you study the source of the WWWVALIDEP program, you will see how the validation process works, and you can adjust it to suit your needs in your own exit programs. If you call the WWWVALIDT4 sample test program, you can debug the HTTP server and step through WWVALIDEP line-by-line.

Example Web Access User Validator usage

The source for four sample test CGI programs is shipped in the QSRC source file in the WWWVALID library, to enable you to check how the Web Access User Validator works. They are automatically compiled during WWWVALID setup. They are defined as follows:

1. WWWVALIDT1: This program does not include any WWWVALID functionality, and simply returns a page to the browser containing basic information about the HTTP server instance in which it is running, the environment variables which are being used for the job and a microsecond count of how long the program took to create the page. Once the information page has been displayed in the browser, the user can press F5 (refresh) repeatedly to redisplay it.

   This program is included as a 'baseline', both to show how simple regular CGI programming functionality can be and also to show that calls to the WWWVALID procedures do not noticeably impact performance. All the other sample CGI programs listed below are based on WWWVALIDT1.

2. WWWVALIDT2: This program is a copy of WWWVALIDT1, but contains a call to `www_validate` using the following parameters:

   ```
   www_sessionid = www_validate('-pagetimeout 30 -sessiontimeout 300');
   ```

   When WWWVALIDT2 is called, the default sign-on page is displayed. The user must enter a valid IBM i user profile and password, at which point the information page is displayed (this time also showing basic WWWVALID-related information). Again, they can press F5 repeatedly to redisplay the page. However, if they don't press F5 within 30 seconds (the value passed for the `-pagetimeout` keyword in the call to `www_validate`), when they next press F5, the sign-on page is displayed with a page timeout error message. Correctly re-entering the same credentials displays the information page (using the same session identifier). After 5 minutes, the session will be ended, because of the value passed for the `-sessiontimeout` keyword. At any point, the user can press the 'Log off' button on the information page, which calls www_endsession to end the session. If they subsequently press F5, the sign-on page is redisplayed with an error message. Correctly entering any credentials displays the information page (using a new session identifier).

3. WWWVALIDT3: This program is a copy of WWWVALIDT2, but additionally contains calls to `www_swapusrprf` and `www_resetusrprf` to show how swapping user profiles can work in a CGI program.

4. WWWVALIDT4: This program is a copy of WWWVALIDT2, but the parameters in the call to `www_validate` include the use of an exit program (the shipped WWWVALIDEP program). Users can debug the HTTP server job and see how WWWVALIDEP functions.

If you want to run these programs, you must temporarily add the following directives to your HTTP server configuration file:

```
<Directory /QSYS.LIB/WWWVALID.LIB/>
   Options +ExecCGI
   Allow From all
</Directory>
ScriptAliasMatch ^/WWWVALID/(.*)  /QSYS.LIB/WWWVALID.LIB/$1.PGM
```

They can then be run from a browser using the following URLs:

```
http://my-ibm-i/wwwvalid/wwwvalidt1
http://my-ibm-i/wwwvalid/wwwvalidt2
http://my-ibm-i/wwwvalid/wwwvalidt3
http://my-ibm-i/wwwvalid/wwwvalidt4
```

where `my-ibm-i` is the name or IP address of your IBM i (including the port specified in the HTTP server configuration).

Once you have tested out these programs, you can remove the HTTP configuration directives, as they are only needed for the example test programs and are not required for any other Web Access User Validation functionality.

Using the Web Access User Validator in a production environment

In a run-time production environment, the only required objects are the WWWVALID service program (*SRVPGM) and the WWWVALID message file (*MSGF) (and its translated copies, if any).

After testing, you can optionally move the WWWVALID service program from the WWWVALID library into a different library (perhaps a production library where your CGI programs exist). If you do this, you must change the binding directory entry in the WWWVALID binding directory to point to the copy of the WWWVALID service program in the production library, so any CGI programs that use any of the Web Access User Validator procedures will bind to the correct copy of the WWWVALID service program.

For instance, if your production library is called CGIPGMLIB, you should run the following commands:

```
MOVOBJ OBJ(WWWVALID/WWWVALID) OBJTYPE(*SRVPGM) TOLIB(CGIPGMLIB)
MOVOBJ OBJ(WWWVALID/WWWVALID) OBJTYPE(*MSGF) TOLIB(CGIPGMLIB)
RMVBNDDIRE BNDDIR(WWWVALID/WWWVALID) OBJ((WWWVALID/WWWVALID *SRVPGM))
ADDBNDDIRE BNDDIR(WWWVALID/WWWVALID) OBJ((CGIPGMLIB/WWWVALID *SRVPGM))
```

Any CGI programs created after this which specify `...BNDDIR(WWWVALID/WWWVALID)...` in their creation command will be bound to the copy of WWWVALID in CGIPGMLIB.

Using the Web Access User Validator in a non-English environment

The `www_validate` procedure writes all job log messages in English. However, any error messages that are displayed in the sign-on page in place of the 'www_validate-errmsg' string can be supplied in any language. By default, these messages are retrieved from the WWWVALID message file, which contains English messages, but by using the `-msgf` keyword in the parameter to `www_validate`, you can override to use a different message file.

For instance, if you wish to display French text for the sign-on page error messages, copy the WWWVALID message file to e.g. WWWMSGFRN, and then translate the messages in the WWWMSGFRN message file to be in French.

At runtime, if the message file specified using the `-msgf` keyword (or the WWWVALID message file, if the `-msgf` keyword is not specified) does not exist in the same library as the WWWVALID service program or if any of the individual messages do not exist in the message file, the text for those messages is retrieved from a hard-coded array in the WWWVALID service program. This hard-coded text is in English, and is as follows:

WWW0001	Error in Web Access User Validator.
WWW0002	Invalid user ID.
WWW0003	Session has timed out. Sign in to start a new session.
WWW0004	User ID not specified.
WWW0005	Password not specified.
WWW0006	Invalid credentials.
WWW0007	Session has ended. Sign in to start a new session.
WWW0008	Page has timed out. Sign in to reconnect to your session.
WWW0009	This session was started by a different user.
WWW0010	Session not found.
WWW0011	Data level incompatibility.

*** END OF DOCUMENT ***