Security Issues

Note: The topic of security is a broad one and outside the scope of this document. However, as new topics arise that relate to the overall topic of configurating web applications, those topics will be included here until they are incorporated into other documents directly related to security. At that time the information will be removed from here and a pointer provided to the other document.

HTTP Authentication based on web.xml definitions rather than Protect statements

Authentication involves challenging the user at the browser to supply a userid and password. That user is allowed to proceed only if the information they supply is properly validated. Basic authentication has long been a feature of the IBM HTTP Server. That authentication may now be performed based on security definitions in the web applications deployment descriptor (`web.xml` file inside the WAR file) rather than with `Protect` statements in the `httpd.conf` file. However, at the time of this writing the IBM HTTP Server is still required to be part of the picture.

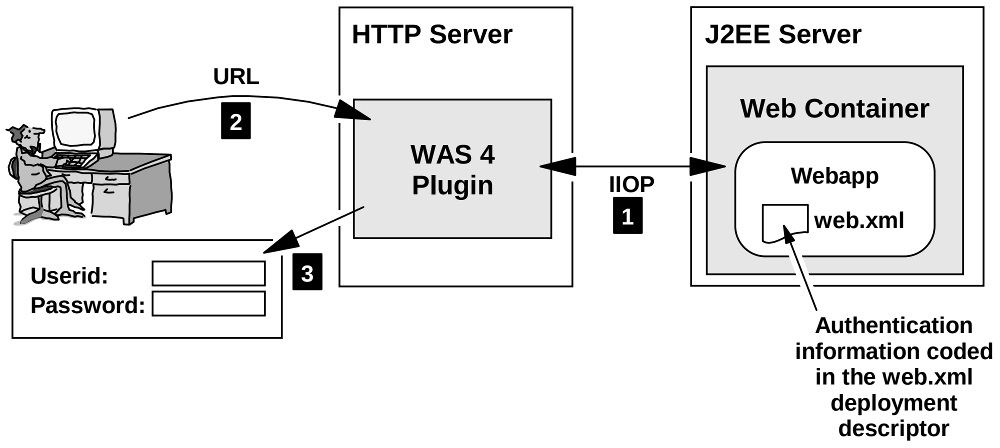
Quick summary of updates required

The following chart summarizes the updates required to enable HTTP authentication out of the web container. These are *in addition to* normal updates required as discussed in other parts of the document.

`httpd.conf`	None (no `PROTECT` statements for the resource are needed)
`httpd.envvars`	`JAVA PROPAGATE=NO`
`was.conf`	None
`webcontainer.conf`	None
WAR file contents	`web.xml` file updated with: • URL resource to be protected specified in `<security-constraint>` stanza with `<url-pattern>` tag • Define the `<auth-constraint>` role to be applied to this protected resource (on `<role-name>` tag) • Define `<security-role>` with the RACF `EJBROLE` `<role-name>` to which authenticated users must have access See "Background: example of definitions in web.xml file" on page 88 for an example of these definitions.
RACF	• Grant HTTP Server's ID `READ` access to `CB.CBIND.`*`server_name`* profile • Grant `READ` access for each authorized userid to the `EJBROLE` profile defined in the `web.xml` file of the web application's WAR file.

Background: how the Web Container performs HTTP authentication

This works as a coordinated effort between the definitions in the `web.xml` file and the HTTP Server:



High-level view of HTTP authentication from web container

1. The web container reads the deployment descriptor of each deployed webapp and passes to the Plugin information related to HTTP authentication.
2. The user sends in a URL that matches the `<url-pattern>` defined in the `web.xml` file. The `<url-pattern>` tag defines the template, or mask, used to match against an inbound URL to determine if protection is required.
3. If a match on the `<url-pattern>` value is made, the plugin asks the HTTP server to pop the logon window requesting the user's userid and password.

Note: The webserver pops the login panel, but *not* based on a `Protect` statement in its `httpd.conf` file. This is based on information passed it by the web container regarding the security constraints defined in the `web.xml` file for deployed webapps in the container.

If the userid and password is valid *and* the userid has at least `READ` access to the `EJBROLE` profile defined in the `web.xml` file, the user is allowed to access the resource implied on the URL.

Question: should I still code the Protect directive in the httpd.conf file?

You should *not* code authentication in both the webapp's deployment descriptor (`web.xml` file) *and* the `httpd.conf` file. Pick one or the other, but don't use both for the same URL resource.

The servlet specification 2.2 defines the location of security constraints and authentication rules to be the webapp's deployment descriptor. If you wish to develop and deploy web applications that adhere to that specification, you should code your security constaints in the deployment descriptor and *not* code `httpd.conf Protect` statements.

Question: can I use this with the new Transport Handler?

At the time of the writing of this version of this document, no. But plans are in place to incorporate into the Transport Handler this function in the near term.

Background: example of definitions in web.xml file

The following example illustrates the contents of the `web.xml` deployment descriptor that relate to HTTP authentication:

Note: The `web.xml` file is typically generated by the tooling program used to create the webapp (for example, WSAD). You wouldn't normally hand-edit the `web.xml` file to set these properties.

```
<security-constraint>
  <web-resource-collection>
    <web-resource-name>Sample Web Resource Collection</web-resource-name>
    <url-pattern>/secret/*</url-pattern>
    <http-method>GET</http-method>
    <http-method>PUT</http-method>
  </web-resource-collection>
  <auth-constraint>
    <description>Sample Security Constraints:+:</description>
    <role-name>Manager</role-name>
  </auth-constraint>
  <user-data-constraint>
    <transport-guarantee>NONE</transport-guarantee>
  </user-data-constraint>
</security-constraint>
<login-config>
  <auth-method>BASIC</auth-method>
  <realm-name>Sample_Security_Realm</realm-name>
</login-config>
<security-role>
  <description>RACF EJBROLE Manager</description>
  <role-name>Manager</role-name>
</security-role>
```

Example of web.xml deployment descriptor and authentication properties

The `<url-pattern>` tag defines the URL string that, if matched, signifies a URL that is to be protected. In this example any URL with `/secret/*` is to be protected. The `<role-name>` tag defines the `EJBROLE` profile that applies to this protection mechanism, in this example `Manager`. Anyone who wishes to access the URL `/secret/*` resource must have `READ` access to the `Manager` profile.

Activity: set security constraint properties for your webapp

This will be done in the tooling you use to create your webapp, and each tool has a different way of setting these values. The bottom line is the WAR file you wish to deploy into the J2EE server must have the `web.xml` file updated with the security constraint values.

Activity: httpd.envvars

- ☐ Code `JAVA_PROPAGATE=NO` in the `httpd.envvars` file. Failure to do that will result in the authentication failing with an obscure message in the "ncf" log:

  ```
  login for userid <userid>with password failed -- rc = 9D000498
  ```

- ☐ Restart the HTTP Server to pick up this new environment variable.

Activity: RACF updates

- ☐ Grant the HTTP Server's ID (the ID under which the server process runs) `READ` access to the `CB.CBIND.`*`server_name`* profile, where *`server_name`* is the name of the J2EE server in which the web container resides.
- ☐ Grant `READ` access to the `EJBROLE` profile named in the `web.xml` file to every userid you wish to have authority over the URL resource. For example, if the `EJBROLE` name

is `Manager`, and you wish the userids `SMITH` and `JOHNSON` to have access to the resource, then grant both userids `READ` access to the `EJBROLE` profile `Manager`.

Question: what's the advantage of web containter authentication vs. Webserver?

The two provide essentially the same level of authentication. However, specifying authentication within the security constraint of a web application's deployment descriptor is one of the things defined in the servlet specification 2.2. For a web application developer who wants to abide by the J2EE specification, coding security constraints within the webapp's deployment descriptor is the correct method.

As stated earlier, to accomplish this you need the WAS 4 plugin running inside the HTTP Server. The plugin has been developed to communicate with the web containers and to understand the security constraints defined in the webapps deployed there. At the present time the Transport Handler is not capable of performing this function, though that should be corrected in the near term.