WEBADE 4

WebADE Developer Module User Guide

Client: BC Government
Date: May 3, 2012
Revision: 01_03_00

Vivid Solutions Inc.
Suite #1A, 2328 Government St.
Victoria, BC V8T 5G5
Phone: (250) 385-6040
Fax: (250) 385-6046
Website: www.vividsolutions.com

Document Change Control

REVISION	DATE OF ISSUE	AUTHOR(S)	DESCRIPTION
1	Nov 22, 2005	Vivid Solutions	Original draft
2	Apr 01, 2006	Vivid Solutions	Moved Database preferences to separate document. Updated documentation for 1.0.1 release.
3	Apr 10, 2006	Vivid Solutions	Updated documentation for 1.1.0 release. New Method of configuring the Developer Module.
4	July 4, 2006	Vivid Solutions	Updated documentation for 1.1.1 release.
5	Sep 7, 2006	Vivid Solutions	Updated documentation for 1.1.2 release.
6	Jan 30, 2007	Vivid Solutions	Updated documentation for 1.1.3 release.
7	Feb 19, 2007	Vivid Solutions	Updated documentation for 1.1.4 release.
8	Mar 29, 2007	Vivid Solutions	Updated documentation for 1.1.5 release.
9	Apr 30, 2007	Vivid Solutions	Updated documentation for 1.1.6 release.
10	June 22, 2007	Vivid Solutions	Updated documentation for 1.1.7 release.
11	March, 2008	Vivid Solutions	Updated documentation for 1.2.0 release.
12	August, 2008	Vivid Solutions	Updated documentation for 1.2.1 release.
13	June, 2008	Vivid Solutions	Updated documentation for 1.2.2 release.
14	May 3, 2012	Vivid Solutions	Updated documentation for 1.3.0 release.

Table of Contents

1. INTRODUCTION

1.1 OVERVIEW

This document explains how to install and configure the WebADE Developer Module. The Developer Module serves two purposes:

1. It enables the developer to emulate user information responses from unreachable external services (e.g. web-services behind an extranet firewall).
2. It enables the developer to emulate authentication from the Siteminder Common Login Page.

The Developer Module is necessary because the resources outlined above may not be available to a developer when developing and testing a WebADE based application. Therefore we require a means of emulating these resources while our application is being developed.

1.2 HOW IT WORKS

1.2.1 USER INFORMATION RETRIEVAL

WebADE usually retrieves extended user information from BC Government web services. This is not always practical when developing applications outside of the government extranet so the developer module provides an alternative.

The WebADE User API has been designed since version 4.1 to accept pluggable user information providers. The developer module includes a user information provider that reads the user information from an XML file. We use this provider to replace the default government web services provider.

This enables developers to build applications without the need to be connected via VPN to the BC Government extranet and web services. This also allows full control over user information when developing and testing applications.

An example user information XML file is supplied in the distribution. This contains an XML file defining a sample set of user information. Developers can edit the contents to contain any arbitrary list of test users. Also supplied is an XML schema useful for verifying the integrity of the user information XML file and an Ant build script to run the xml schema validation.

1.2.2 AUTHENTICATION

The developer module uses a servlet request filter to authenticate the user. If the request filter is configured for your application it will check if you are currently known to WebADE. If you have not already been authenticated by WebADE the filter will either redirect you to a login page where you will select a user and log in or challenge you for basic authentication credentials (depends on your

configuration of the filter). The filter then appends parameters to the request that enable WebADE to authenticate you.

2. THE PRODUCTION ENVIRONMENT
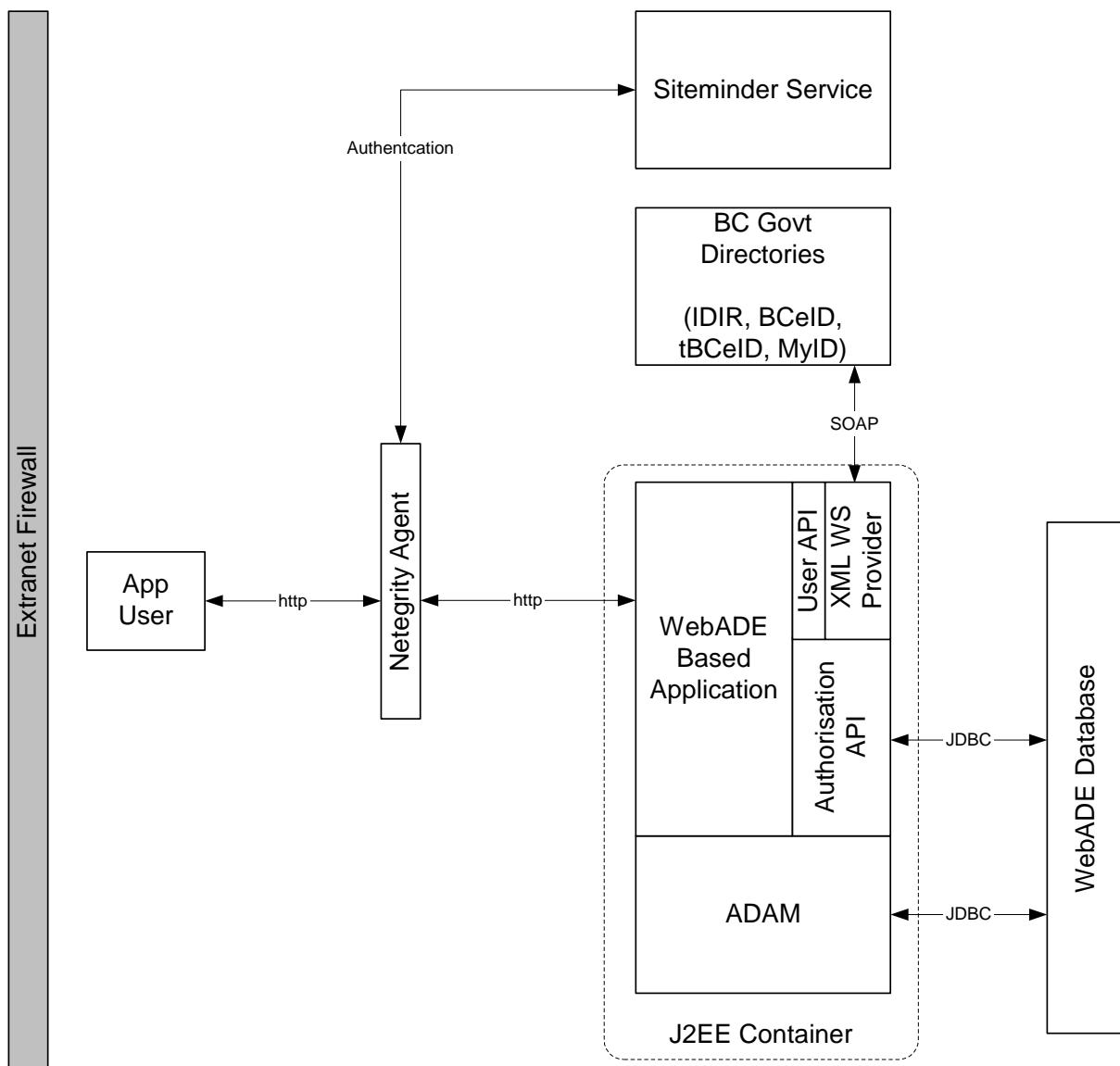
3. THE DEVELOPMENT ENVIRONMENT
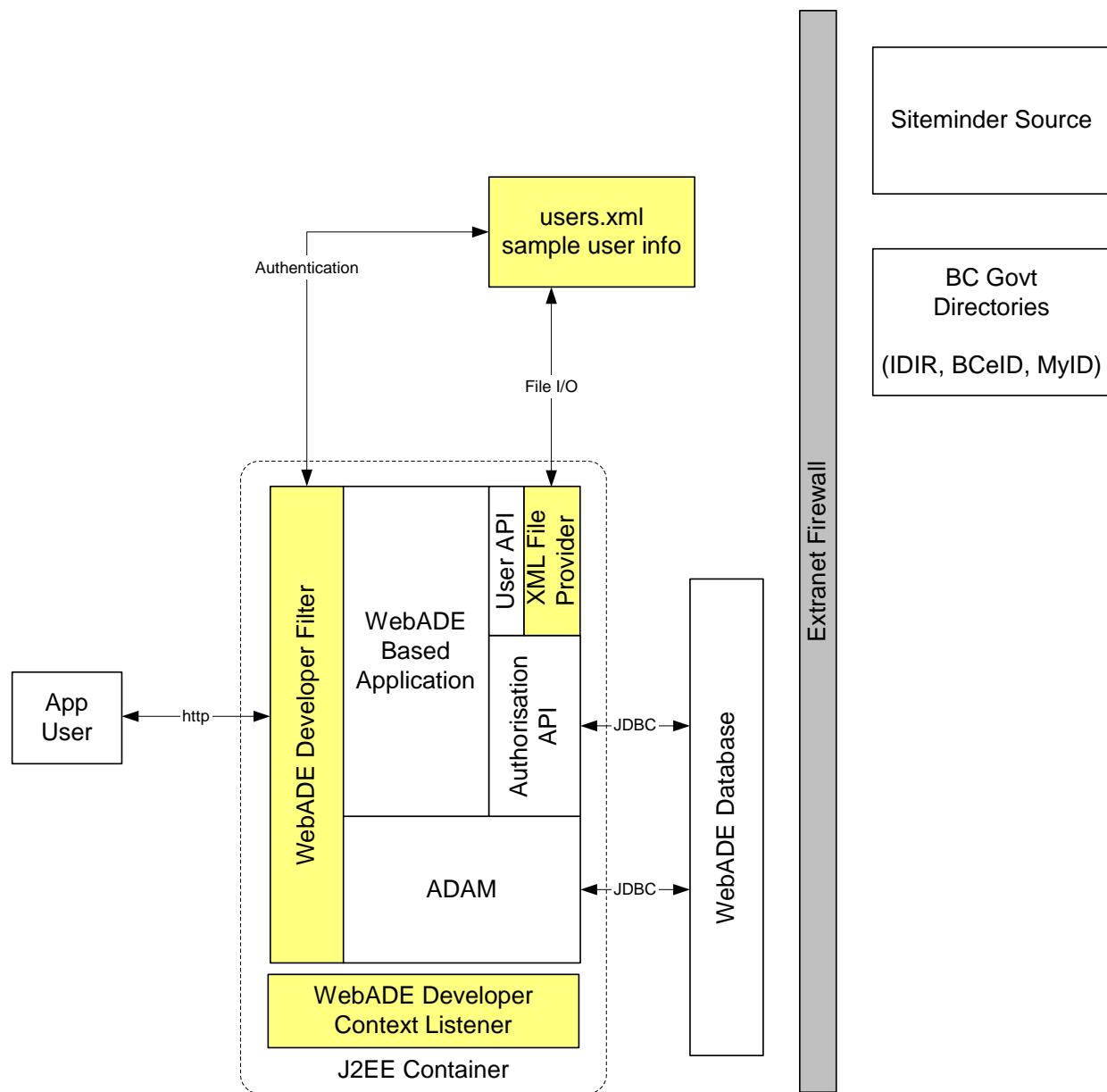
4. INSTALLING THE DEVELOPER MODULE

Note: Before installing the WebADE Developer Module, you need to have already installed WebADE and completed the database setup for your new application. (See the WebADE Administrator's Guide and WebADE Users Guide.)

Note: Parts of this setup process assume you have Ant and recent xml parser libraries installed.

Note: There are special requirements for installing XML parsing libraries in a Java Virtual Machine as Java will ignore certain packages in the classpath by default for security reasons. (See http://java.sun.com/j2se/1.4.2/docs/guide/standards)

Your WebADE application can be configured to use the WebADE Developer Module by following these steps:

4.1   UNPACK THE DEVELOPER MODULE

Unzip the supplied distribution archive to a convenient location.

4.2   INCLUDE REQUIRED CLASS LIBRARIES

Copy the `webade-developer-module-X.X.X.jar` file (found in the deliverable) to your application classpath. To do this you can copy the file to a common library location in your container, you could add it to your web application's class library directory. (e.g. `WEB-INF/lib`.), or change a classpath configuration setting for your container. It's up to you as long as **the jar library is available to your application in the classpath**.

4.3   CONFIGURE USER INFORMATION RETRIEVAL

WebADE needs to know where to look for user information. Normally it would request this information from government web services. In this step we will set up an XML file user information provider to replace the default web services user information provider.

4.3.1   PREPARE USER INFORMATION XML FILE

An example user information file `webade-xml-user-info.xml` can be found in the distribution, copy it to a network location where your application container can access the file. This usually means a directory on the same server or a mapped network drive.
Important Note: There should only be one xml user information file used for each WebADE database schema. This is because WebADE caches user information in the database and having multiple user information files floating

about could cause data to be overwritten. It is important to keep in mind that because we are specifying the location of the xml file on the server, in the WebADE database, all application servers using this WebADE database will need read access to the xml file. Every container must be able to find the xml file at the path retrieved form the database.

4.3.2   INSTALL THE NEW PROVIDER IN THE WEBADE DATABASE

The database installation SQL can be found in the distribution at `src/main/sql/setup/setup.sql`. Use this script to install the xml file user information provider and disable the default web services provider.

You will be prompted for the following value:

`user_info_file_location`
The path to the user information xml file prepared in the previous step.

User Information Retrieval is now configured. WebADE will retrieve user information from the xml file specified by user-info-file-location.

4.4   CONFIGURING USER AUTHENTICATION

Before we can log in to our application WebADE needs to know who is logging in. That's where authentication comes in.

4.4.1   ADD THE FILTER

To enable authentication we simply add the required WebADE preferences. The database installation SQL can be found in the distribution at `src/main/sql/setup/filter_setup.sql`

4.5   RESTART CONTAINER

Restart your container

The Developer Module is now installed and configured.

5. ADVANCED CONFIGURATION

5.1 CONFIGURING AUTHENTICATION

5.1.1 PASSWORD PROTECTION

If you wish to enable password security on user authentication set the following init-param preference value to 'true' in the WebADE Developer Filter preferences. A value of false will disable password security. Defaults to false if not specified.

e.g.

type_code	sub_type	set_name	name	default
WDE	dynamic-filter	webade-developer-filter	init-param-password-enabled	true

The user will be required to input a password which will be verified against the password attribute in the user information xml file.

5.1.2 FILTERING USERS BY ACCESS PRIVILEDGES

The list of selectable users displayed by the authentication form can be filtered to include only those with at least one WebADE role for the application you are trying to access. A value of 'true' will turn this filtering on. A value of false will disable the filtering and display all users listed in the user information xml file. Defaults to false if not specified.

type_code	sub_type	set_name	name	value
WDE	dynamic-filter	webade-developer-filter	init-param-filter-users-by-access	true

5.1.3 USING COOKIES FOR SINGLE SIGN-ON

To implement single sign-on functionality the developer module uses a browser session cookie containing the GUID of the selected user. This will eliminate the need to login again when browsing from one WebADE application to another within the same browser session.

If you wish to enable single sign-on on user authentication set the use-cookie value to 'true' in the WebADE Developer Filter preferences. A value of false will disable single sign-on. Defaults to false if not specified.

e.g.

type_code	sub_type	set_name	name	value
WDE	dynamic-filter	webade-developer-filter	init-param-use-cookie	true

By default the domain of the cookie will be restricted to the server domain. If you want to implement an organization wide single sign-on set the cookie-domain value to the domain of your organization in the WebADE Developer Filter preferences.

type_code	sub_type	set_name	name	value
WDE	dynamic-filter	webade-developer-filter	init-param-use-cookie	true
WDE	dynamic-filter	webade-developer-filter	init-param-cookie-domain	.mycompany.com

Note: Always begin the domain value string with a period character.

e.g.

```
.mycompany.com
```

Note: When password security is enabled an encrypted hash of the password is stored in the cookie. This provides a secure way providing single sign-on for password protected applications.

5.1.4 AUTHENTICATION MODES

Set the authentication mode, as follows:

form: Only form authentication is allowed.

type_code	sub_type	set_name	name	value
WDE	dynamic-filter	webade-developer-filter	init-param-authentication-mode	form

basic: Only BASIC authentication is allowed. If the client supplies valid BASIC headers in advance, they will be used. If the client specifies invalid BASIC headers in advance, or no BASIC headers, then the client will be challenged until valid BASIC headers are provided.

type_code	sub_type	set_name	name	value
WDE	dynamic-filter	webade-developer-filter	init-param-authentication-mode	basic

form+basic: BASIC and form authentication are allowed. If the client supplies valid BASIC headers in advance, they will be used. If the client supplies invalid BASIC headers in advance, the client will be challenged until valid BASIC headers are supplied. If the client does not provide BASIC headers, then form authentication is used.

type_code	sub_type	set_name	name	value
WDE	dynamic-filter	webade-developer-filter	init-param-authentication-mode	form+basic

The BASIC realm to use for authentication. Normally, this would never need to be changed. It is provided just in case it is necessary for some configurations. This setting is only relevant if the authentication method is either basic or form+basic. Default: WebADEDeveloperFilter

type_code	sub_type	set_name	name	value
WDE	dynamic-filter	webade-developer-filter	init-param-authentication-mode	form+basic
WDE	dynamic-filter	webade-developer-filter	init-param-authentication-realm	WebADEDeveloperFilter

5.1.5   USER INFORMATION FILE LOCATION (OPTIONAL)

Occasionally a developer wishes to use the WebADE Developer Module filter to authenticate the user but does not want to use the WebADE Developer Module User Provider. They instead want to use the default BCeID Web Services to provide user information. It is relatively simple to configure this setup. Two steps are required:
1. Ensure the BCeID Web Services are enabled (WebADE default) and the WebADE Developer Module User provider is disabled.
2. Add the user info file location to the filter parameters. Normally the filter would just retrieve this from WebADE preferences on servlet start-up but because the WebADE Developer ServletContextListener will never be initialised in this setup we need to provide the information directly to the filter.

type_code	sub_type	set_name	name	value
WDE	dynamic-filter	webade-developer-filter	init-param-user-info-file-location	/path/to/file.xml

5.2   THE DEVELOPER MODULE AND WEBADE LEGACY APPLICATIONS

If your application extends the WebADEActionServlet from WebADE 3 or earlier you will need to use a different configuration to override parameters in the web.xml deployment descriptor.

We will assume you are using the webade-legacy.jar to divert WebADE calls to the WebADE 4 API. If you have not set this up yet please consult the WebADE 4 Administrator documentation for how to do this. The WebADE Developer Module will not work with WebADE 3 Applications unless the webade-legacy redirect is in place.

WebADE 3 applications required the developer to extend the `ca.bc.gov.webade.http.WebADEActionServlet` class.

WebADE is initialized during the start-up of this servlet. Because some containers initialize servlets before ServletContextListeners the Developer Module ServletContextListener would not be guaranteed to execute before the WebADE is

initialized. This situation could prevent the developer module from working correctly.

To ensure the developer module is initialised before the WebADE we can use the WebADEDeveloperLegacyServlet instead of the WebADEDeveloperServletContextListener. We tell the container explicitly to start it first by specifying a load-on-startup value less than that of our application servlet (Must be a positive integer).

e.g. Simply add the following configuration to your web.xml file.

Note: The order of elements in the web.xml file is very important!

```
  <servlet>
    <servlet-name>WebADEDeveloperLegacyServlet</servlet-name>
    <servlet-
class>ca.bc.gov.webade.developer.listener.WebADEDeveloperLegacyServlet</servl
et-class>
    <load-on-startup>1</load-on-startup>
  </servlet>
```

Pay close attention to the load-on-startup values.
The load-on-startup value of your application servlet must be greater than that of the WebADEDeveloperLegacyServlet.

e.g.

```
<servlet>
  <servlet-name>action</servlet-name>
  <servlet-class>ca.bc.gov.sample.SampleControllerServlet</servlet-class>
  <init-param>
    <param-name>config</param-name>
    <param-value>/WEB-INF/struts-config.xml</param-value>
  </init-param>
  <load-on-startup>2</load-on-startup>
</servlet>
```

6. DEPRECATED CONFIGURATION

Please note the following configuration, although still supported, is deprecated and may be removed from upcoming releases. Please use WebADE preferences to configure the developer module for your application as described above.

6.1 CONFIGURING THE USER INFORMATION PROVIDER

6.1.1 EDITING THE USER INFO XML FILE

You can do this by editing existing or inserting new `government-user-info`, `business-partner-user-info`, `individual-user-info`, or `verified-individual-user-info` elements into the xml file's `<users>` element. The supplied sample file contains examples of each type of user element. The simplest way to add new user info elements is simply to copy an existing one then edit the new element to contain the data you require. Remember to ensure you **do not repeat unique identifiers** like GUID. You can validate the xml file by executing the Ant build script in the `xml` directory.

Note: Validation does not check for duplicate unique identifiers. You must ensure you do not duplicate these attributes in the xml file.
You may use the `webade-xml-user-info.xsd` schema file to ensure the integrity of the `webade-xml-user-info.xml` file at any time by navigating to the `xml` directory and executing the command:

`ant validate`

6.1.2 OVERRIDING CONFIGURATION SETTINGS IN WEB.XML

To override configuration settings in the web.xml we first need to add the WebADEDeveloperServletContextListener.
Copy and paste the following listener element into your web.xml deployment descriptor immediately before the WebADEServletContextListener.

Note: The order of elements in the web.xml file is very important! Ensure you add this listener element immediately before the WebADEServletContextListener in the web.xml deployment descriptor.

```
<listener>
  <listener-
class>ca.bc.gov.webade.developer.j2ee.WebADEDeveloperServletContextListener</
listener-class>
</listener>

<listener>
    <listener-
class>ca.bc.gov.webade.j2ee.WebADEServletContextListener</listener-class>
```

```
</listener>
```

To override a configuration parameter add a context param to the web.xml deployment descriptor with the appropriate param-name and value.

Note: context-param elements must be added to the web.xml deployment descriptor in the correct location. Check the DTD if you are not sure where to add a context-param element.

e.g.

```
<context-param>
   <param-name>user.info.file.location</param-name>
   <param-value>path/to/webade-xml-user-info.xml</param-value>
</context-param>
```

The following is a listing of the configuration parameters that can be overridden in the web.xml deployment descriptor.

- user.info.file.location
- user.info.response.delay
- webade.preference.file.location

6.1.3   OVERRIDING USER INFO FILE LOCATION

```
<context-param>
   <param-name>user.info.file.location</param-name>
   <param-value>path/to/webade-xml-user-info.xml</param-value>
</context-param>
```

```
<context-param>
   <param-name>user.info.file.location</param-name>
   <param-value>default</param-value>
</context-param>
```

Note: A value of 'default' can be specified for user.info.file.location. This will cause the included example file to be used from within the jar archive. This is handy if you want to hook up the developer module with a minimum of moving parts.

Note: user-info-file-location can also be specified as a System property.

```
System.setProperty("user-info-file-location", "path/to/webade-xml-user-
info.xml");
```

6.1.4 OVERRIDING WEBADE PREFERENCE FILE LOCATION

```
<context-param>
  <param-name>webade.preference.file.location</param-name>
  <param-value>path/to/webade-preferences.xml</param-value>
</context-param>
```

```
<context-param>
  <param-name>webade.preference.file.location</param-name>
  <param-value>default</param-value>
</context-param>
```

Note: A value of 'default' can be specified for webade.preference.file.location. This will cause the included example file to be used from within the jar archive. This is handy if you want to hook up the developer module with a minimum of moving parts.

Note: webade-preference-file-location can also be specified as a System property.

```
System.setProperty("webade-preference-file-location", "default");
```

6.1.5 OVERRIDING USER INFORMATION RESPONSE DELAY

The user information response delay emulates the latency of web services and helps the developer to avoid calling these expensive methods excessively.
The user information response delay defaults to 0 milliseconds if not specified in the database. The database install script sets it to 200 milliseconds by default. If you want to change the delay you can edit the database preference value (Recommended), or override the value using the following context param in the deployment descriptor.

```
<context-param>
  <param-name>user.info.response.delay</param-name>
  <param-value>300</param-value>
</context-param>
```

6.2 CONFIGURING AUTHENTICATION

6.2.1 AUTHENTICATION MODES

Note: The WebADE Developer Authentication Filter can be mapped to any url.

```
<!-- The WebADE Developer Filter is applied only to a protected URL -->
```

```
<filter-mapping>
  <filter-name>WebADE Developer Filter</filter-name>
  <url-pattern>/some/url/*</url-pattern>
</filter-mapping>
```

Set the authentication mode, as follows:

basic: Only BASIC authentication is allowed. If the client supplies valid BASIC headers in advance, they will be used. If the client specifies invalid BASIC headers in advance, or no BASIC headers, then the client will be challenged until valid BASIC headers are provided.

form+basic: BASIC and form authentication are allowed. If the client supplies valid BASIC headers in advance, they will be used. If the client supplies invalid BASIC headers in advance, the client will be challenged until valid BASIC headers are supplied. If the client does not provide BASIC headers, then form authentication is used.

form: Only form authentication is allowed.

Examples:

```
<!-- WebADE Developer Module Filter for authenticating user on the servlet-->
  <filter>
    <filter-name>WebADE Developer Filter Servlet</filter-name>
    <filter-
class>ca.bc.gov.webade.developer.filter.WebADEDeveloperFilter</filter-class>
    <init-param>
      <param-name>authentication-method</param-name>
        <param-value>form+basic</param-value>
        <description>
Set the authentication mode, as follows:
basic: Only BASIC authentication is allowed. If the client supplies valid
BASIC headers in advance, they will be used. If the client specifies invalid
BASIC headers in advance, or no BASIC headers, then the client will be
challenged until valid BASIC headers are provided.

form+basic: BASIC and form authentication are allowed. If the client supplies
valid ASIC headers in advance, they will be used. If the client supplies
invalid BASIC headers in advance, the client will be challenged until valid
BASIC headers are supplied. If the client does not provide BASIC headers,
then form authentication is used.

form: Only form authentication is allowed.

Values: basic, form+basic, form
Default: form
      </description>
    </init-param>
  </filter>
```

```
   <!-- WebADE Developer Module Filter for BASIC authentication on a protected
URL -->
  <filter>
    <filter-name>WebADE Developer Filter BASIC URL</filter-name>
    <filter-
class>ca.bc.gov.webade.developer.filter.WebADEDeveloperFilter</filter-class>
    <init-param>
            <param-name>authentication-method</param-name>
            <param-value>basic</param-value>
      </init-param>
    <init-param>
      <param-name>authentication-realm</param-name>
      <param-value>WebADEDeveloperFilter</param-value>
      <description>
The BASIC realm to use for authentication. Normally, this would never need to
be changed. It is provided just in case it is necessary for some
configurations. This setting is only relevant if the authentication method is
either basic or form+basic. Default: WebADEDeveloperFilter
      </description>
    </init-param>
  </filter>
```

```
   <!-- WebADE Developer Module Filter for form authentication on a protected
URL -->
  <filter>
    <filter-name>WebADE Developer Filter FORM URL</filter-name>
    <filter-
class>ca.bc.gov.webade.developer.filter.WebADEDeveloperFilter</filter-class>
    <init-param>
      <param-name>authentication-method</param-name>
      <param-value>form</param-value>
    </init-param>
  </filter>
```

6.2.2   PASSWORD PROTECTION

If you wish to enable password security on user authentication set the following init-param element value to 'true' in the WebADE Developer Filter. A value of false will disable password security. Defaults to false if not specified.

e.g.

```
<filter>
  <filter-name>WebADE Developer Filter</filter-name>
  <filter-
class>ca.bc.gov.webade.developer.filter.WebADEDeveloperFilter</filter-class>

  <init-param>
    <param-name>password-enabled</param-name>
    <param-value>true</param-value>
  </init-param>
```

```
</filter>
```

The user will be required to input a password which will be verified against the password attribute in the user information xml file.

6.2.3   FILTERING USERS BY ACCESS PRIVILEDGES

The list of selectable users displayed by the authentication form can be filtered to include only those with at least one WebADE role for the application you are trying to access. A value of 'true' will turn this filtering on. A value of false will disable the filtering and display all users listed in the user information xml file. Defaults to false if not specified.

```
<filter>
  <filter-name>WebADE Developer Filter</filter-name>
  <filter-
class>ca.bc.gov.webade.developer.filter.WebADEDeveloperFilter</filter-class>

  <init-param>
    <param-name>filter-users-by-access</param-name>
    <param-value>true</param-value>
  </init-param>

</filter>
```

6.2.4   USING COOKIES FOR SINGLE SIGN-ON

To implement single sign-on functionality the developer module uses a browser session cookie containing the GUID of the selected user. This will eliminate the need to login again when browsing from one WebADE application to another within the same browser session.
      If you wish to enable single sign-on on user authentication set the use-cookie init-param element value to 'true' in the WebADE Developer Filter. A value of false will disable single sign-on. Defaults to false if not specified.

e.g.

```
<filter>
  <filter-name>WebADE Developer Filter</filter-name>
  <filter-
class>ca.bc.gov.webade.developer.filter.WebADEDeveloperFilter</filter-class>

  <init-param>
    <param-name>use-cookie</param-name>
    <param-value>true</param-value>
  </init-param>
```

```
</filter>
```

By default the domain of the cookie will be restricted to the server domain. If you want to implement an organization wide single sign-on set the cookie-domain init-param element value to the domain of your organization in the WebADE Developer Filter.

```
<filter>
  <filter-name>WebADE Developer Filter</filter-name>
  <filter-
class>ca.bc.gov.webade.developer.filter.WebADEDeveloperFilter</filter-class>

  <init-param>
    <param-name>use-cookie</param-name>
    <param-value>true</param-value>
  </init-param>
  <init-param>
    <param-name>cookie-domain</param-name>
    <param-value>.gov.bc.ca</param-value>
  </init-param>

</filter>
```

Note: Always begin the domain value string with a period character.

e.g.

```
.mycompany.com
```

Note: When password security is enabled an encrypted hash of the password is stored in the cookie. This provides a secure way providing single sign-on for password protected applications.

7. UNINSTALLING THE DEVELOPER MODULE

7.1   REMOVE AUTHENTICATION

Disable the user authentication filter in the WebADE database preferences.
i.e. Set the enabled preference to 'false'

7.2   REMOVE USER INFORMATION RETRIEVAL

Disable the xml file user information provider in the WebADE database preferences.
i.e. Set the enabled preference to 'false'

Enable the web services provider in the WebADE database preferences.
i.e. Set the enabled preference to 'true'

7.3   REMOVE JAR FROM CLASS LIBRARIES

Remove the `webade-developer-module-X.jar` file from your application class path.

7.4   RESTART CONTAINER

Restart your container

The Developer Module is now uninstalled.

8. CONTENTS OF THE WEBADE DEVELOPER MODULE DISTRIBUTION

FILE	PURPOSE
webade-developer-module-X.X.X.jar	The WebADE Developer Module Java Library.
doc/	WebADE Developer Module Documentation
dependencies/	Libraries that the WebADE Developer Module requires
database/	Database install and configuration scripts
xml/user/webade-xml-user-info.xml	An example user information XML file.
xml/user/webade-xml-user-info.xsd	User info XML schema.
xml/preferences/webade-preferences.xml	An example WebADE preferences XML file.
xml/preferences/webade-preferences.xsd	WebADE Preferences XML schema.
xml/build.xml	Ant script to run validation on user info and preferences files