WEBADE 4.3.0

Administrator's Guide

Client: BC Provincial Government
Date: July 24, 2012
Revision: 9

Document Change Control

REVISION NUMBER	DATE OF ISSUE	AUTHOR(S)	DESCRIPTION
1	Dec 3, 2004	Jason Ross	Original draft
2	May 3, 2005	Jason Sherman	Updated LDAP Preferences regarding employee indicator.
3	November 7, 2005	Jason Ross	Updated to include instructions for the new CAP web services functionality
4	January 30, 2006	Jason Ross	Updated to remove reference to "Global User Agreements" and add WebADE Preferences
5	March 8, 2006	Jason Ross	
6	September 25, 2006	Jason Ross	Updated with text updates and added troubleshooting information.
7	January 25, 2007	Jason Ross	Added more troubleshooting tips to the Appendix.
8	March 31, 2008	Jason Ross	Updated document for WebADE 4.2.0
9	July 24, 2012	Andrew Wilkinson	Updated document for WebADE 4.3.0. Removed LDAP. Updated WebADE User Providers. Added User Agreements, Application Enabled Flag, and Organization Selection.

Table of Contents

1. OVERVIEW

1.1   THE WEBADE FRAMEWORK

The WebADE framework is a combination of database tables and stored procedures, Java Servlet and JSP technology, and Java libraries that provide a common set of functionality to web-based applications.

This document describes the details of installing and administrating the WebADE framework. This administration can be separated into two parts: General WebADE administration and application-specific administration tasks.

In addition to the basic WebADE framework, there are a number of optional extensions that can be used by applications implementing the WebADE framework. These include Report Generation and Mail Scheduling. The administration of these extensions are detailed in the documentation included with the extensions' individual release bundle.

2. INSTALLING THE WEBADE

This section describes how to:

1. Install the database component for the WebADE framework.
2. Install the WebADE Java web application framework.
3. Deploy and configure web applications that use the framework.

2.1   TERMINOLOGY

The following terms are used in this manual:

WebADE Application Server	The machine which the target J2EE application server is installed on.
WebADE Database	The database of configuration information for WebADE environment and the applications installed in the environment.
Deployment	This refers to moving the code for an application onto the WebADE Application Server.
Configuration	This refers to making the necessary settings in the WebADE Database for an application.

2.2   INSTALLING THE WEBADE

There are two components that make up the WebADE; there is the Java Web server side and the Database side. The WebADE should be installed for the Dev, Test, and Production environments (additional environments may be added. E.g. Training).

2.2.1   WEBADE DATABASE INSTALL

NOTE: Before installing the WebADE database, you need to have already installed a proper MoF database environment. Please contact Mark Will at the Ministry of Forests for more information.

The following steps need to be performed to set up the WebADE Database environment:

Obtain the WebADE DDL Scripts used to create the WebADE tables, indexes, and foreign keys on an ORACLE database, from the WebADE custodian.
Install the database objects by executing the DDL scripts in SQL Plus, or an equivalent interface.
Obtain the scripts used to create the WebADE ORACLE stored procedures and packages, from the WebADE Custodian.
Install the stored procedures and packages by executing the scripts in SQL Plus or equivalent interface.

Create a WebADE Proxy ID that has SELECT authority on the WebADE tables and EXECUTE on the WebADE procedures and packages. This userid and password will be required for the configuration of the WebADEConnection.java file.

Configuring the WebADE database is further outlined in section 3.1.

2.2.2   WEBADE JAVA INSTALL

The WebADE application consists of a set of Java classes compiled and packaged into Jar files. The main WebADE framework is contained in a single JAR file called WebADE.jar.

Also, each application requires the database URL, user name, and password to be used to connect to the WebADE Database. These settings are stored in the WebADEConnection.java source file, and need to be configured, based on the location and login settings of the Oracle database that hosts the WebADE Database.

To install the WebADE Java components, perform the following steps:

Acquire the WebADE.jar class library.
Acquire the WebADEConnection.java source file.
Edit the WebADEConnection.java file to configure it for the environment it is being installed into (see section 3.1).
Compile the source WebADEConnection.java file into a class file.
Zip the class file into a library called WebADEConnection.jar.
Copy the WebADEConnection.jar file to the J2EE application server's shared library directory.
The WebADE.jar file is installed in the particular web application's library directory: WEB-INF/lib.

3. CONFIGURING THE WEBADE

This section details the required steps to configure an application server and database server to run the WEB ADE components.

3.1 WEBADE CONNECTION JAR

The WebADEConnection.java file must be modified and compiled to contain the appropriate configuration settings. Java files can be compiled using any Java IDE. The compiled class file must be added into a new zip file called WebADEConnection.jar.

The WebADEConnection.java file contains the following fields that must be configured:

3.1.1 MANDATORY FIELDS

The following fields vary per WebADE database environment, and should be configured for each WebADE connection jar.

WEBADE_ENVIRONMENT

The WEBADE_Environment constant must be set to the database environment the application will be accessing.

Example:

```
private static final String WEBADE_ENVIRONMENT = "DEV";
```

The value of this constant must match one of the environment codes contained in the ENVIRONMENT table in the WebADE Database.

If the WEBADE_ENVIRONMENT code does not match the code returned by the Oracle stored procedure WEBADE_4_GET_ENVIROMENT then an exception is raised in the application.

WEBADE_JDBCURL

The WEBADE_JDBCURL constant specifies the Oracle JDBC URL to connect to the WEB ADE database with.

Example:

```
private static final String
WEBADE_JDBCURL="jdbc:oracle:thin:@200.22.22.22:1521:DEV";
```

WEBADE_USERID

The WEBADE_USERID constant specifies the Oracle userid to use to connect to the WEB ADE database with.

Example:

```
private static final String WEBADE_USERID = "WEBADE_ADMIN";
```

WEBADE_PASSWORD

The WEBADE_PASSWORD constant specifies the oracle password to use to connect to the WEB ADE database with.

Example:

```
private static final String WEBADE_PASSWORD = "secret";
```

3.1.2 OPTIONAL FIELDS

The following fields allow for configuration of the connection pool, but it is likely these values will be set identically for all connection jars, and do not require changes for each jar creation.

WEBADE_MIN_CONNECTIONS

Specifies the minimum number of connections that can be pooled for the WebADE database connection pool. The default value (DEFAULT_MIN_CONNECTIONS) is used, if this is not specified.

Example:

```
private static final int WEBADE_MIN_CONNECTIONS = 0;
```

WEBADE_MAX_CONNECTIONS

Specifies the maximum number of connections that can be pooled for the WebADE database connection pool. The default value (DEFAULT_MAX_CONNECTIONS) is used, if this is not specified.

Example:

```
private static final int WEBADE_MAX_CONNECTIONS = 10;
```

WEBADE_MAX_CONNECTION_IDLE_TIME

The maximum time, in minutes, a connection should remain open while available in the pool (0 if the connection should be left open indefinitely). The default value (DEFAULT_MAX_IDLE_TIME) is used, if this is not specified.

Example:

```
private static final int WEBADE_MAX_CONNECTION_IDLE_TIME = 10;
```

WEBADE_MAX_CONNECTION_CHECKOUT_TIME

The maximum time, in minutes, a connection should remain checked out of the pool (0 if the connection should be left open indefinitely). The default value (DEFAULT_MAX_CHECKOUT_TIME) is used, if this is not specified.

Example:

```
private static final int WEBADE_MAX_CONNECTION_CHECKOUT_TIME = 10;
```

WEBADE_MAX_CONNECTION_WAIT_TIME

The maximum time, in milliseconds, a thread should block while waiting for a connection from the pool (0 if the thread should block indefinitely, -1 of it should never block). The default value (DEFAULT_MAX_WAIT_TIME) is used, if this is not specified.

Example:

```
private static final int WEBADE_MAX_CONNECTION_WAIT_TIME = 0;
```

WEBADE_MONITOR_SLEEP_TIME

The wait time, in minutes, the monitor thread will wait between connection pool checks. The default value (DEFAULT_MONITOR_SLEEP_TIME) is used, if this is not specified.

Example:

```
private static final int WEBADE_MONITOR_SLEEP_TIME = 1;
```

WEBADE_POOL_CONNECTIONS_FLAG

A flag indicating whether or not to pool database connections. The default value (DEFAULT_POOL_CONNECTIONS_FLAG) is used, if this is not specified.

Example:

```
private static final boolean WEBADE_POOL_CONNECTIONS_FLAG = true;
```

WEBADE_PING_CONNECTIONS_FLAG

A flag indicating whether or not to test every database connection before allowing it to be checked out. Acceptable values are true or false. The default value (DEFAULT_PING_CONNECTIONS_FLAG) is used, if this is not specified.

Example:

```
private static final boolean WEBADE_PING_CONNECTIONS_FLAG = false;
```

3.2   WEBADE REQUIRED PREFERENCES

All WebADE applications require certain application preference settings to properly function. This section describes all of the preferences that must exist in the WebADE Preferences for all applications.

NOTE: Like the sensitive Proxy Control connection information, all application preferences with a 'WDE' preference type are considered sensitive, and are not exposed to the end developer. These values are only viewable by the WebADE itself.

3.2.1   ANONYMOUS-USER (DEPRECATED)

NOTE: The 'anonymous-user' preference has been deprecated. With the release of WebADE 4.1, authorizations can now be granted, using ADAM, to all users, or all IDIR, BCeID, or MYID users. Please see the ADAM documentation for more information.

The 'anonymous-user' preference defines the username, including domain name, of the user to query for authorization to application actions and resources for users accessing the system anonymously.

The 'anonymous-user' preference's database record should look similar to this:

COLUMN NAME	COLUMN VALUE	DESCRIPTION
PREFERENCE_ID	preference_seq.NEXTVAL	Unique sequence-generated database id.
PREFERENCE_TYPE_CODE	'WDE'	Indicates that this is an application preference.
PREFERENCE_SUB_TYPE	'app-config'	Indicates that this is an application configuration preference.
APPLICATION_ACRONYM	*NULL*	No application acronym makes this a default preference.
PREFERENCE_SET_NAME	*NULL*	This preference is a standalone preference, and should have no set name
PREFERENCE_NAME	'anonymous-user'	The preference name
PREFERENCE_VALUE	'IDIR/ANONUSER1'	The DOMAIN/USERNAME string
EUSER_ID	*NULL*	This column is only used for preferences of type 'USR'

CREATED_BY	*'db_admin'*	The database user creating the record.
CREATED_DATE	*SYSDATE*	Current time and date.
UPDATED_BY	*'db_admin'*	The database user creating the record.
UPDATED_DATE	*SYSDATE*	Current time and date.
REVISION_COUNT	*1*	Increment this value every time this record is edited.

Sample 'anonymous-user' Preference Record

3.2.2   USER-MAX-CACHE-TIME

The 'user-max-cache-time' preference defines how long, in milliseconds, a user's information is cached in the database between user info lookups. Each time a user's information is looked up in a WebADE application, the update timestamp on the user's record in the database is checked against the current time. If the user was last updated at a time-difference longer than the 'user-max-cache-time', then the record in the database is updated from the user provider it was retrieved from, and then the information is returned to the user.

The 'user-max-cache-time' preference's database record should look similar to this:

COLUMN NAME	COLUMN VALUE	DESCRIPTION
PREFERENCE_ID	preference_seq.NEXTVAL	Unique sequence-generated database id.
PREFERENCE_TYPE_CODE	'WDE'	Indicates that this is an application preference.
PREFERENCE_SUB_TYPE	'app-config'	Indicates that this is an application configuration preference.
APPLICATION_ACRONYM	*NULL*	No application acronym makes this a default preference.
PREFERENCE_SET_NAME	*NULL*	This preference is a standalone preference, and should have no set name
PREFERENCE_NAME	'user-max-cache-time'	The preference name
PREFERENCE_VALUE	*'86400000'*	The cache time, in milliseconds (86400000 ms is 24 hours)
EUSER_ID	*NULL*	This column is only used for preferences of type 'USR'
CREATED_BY	*'db_admin'*	The database user creating the record.
CREATED_DATE	*SYSDATE*	Current time and date.
UPDATED_BY	*'db_admin'*	The database user creating the record.
UPDATED_DATE	*SYSDATE*	Current time and date.
REVISION_COUNT	*1*	Increment this value every time this record is edited.

Sample 'user-max-cache-time' Preference Record

3.2.3   GROUP-MAX-CACHE-TIME (DEPRECATED)

The 'group-max-cache-time' preference defines how long, in milliseconds, a group's information is cached in memory between user provider lookups. Each time a group's information is looked up in a WebADE application, the update timestamp on the group's record in memory is checked against the current time. If the group was last updated at a time-difference longer than the 'group-max-cache-time', then the record in the database is updated from the user provider directory it was retrieved from, the memory copy is updated, and then the information is returned to the user.

The 'group-max-cache-time' preference's database record should look similar to this:

COLUMN NAME	COLUMN VALUE	DESCRIPTION
PREFERENCE_ID	preference_seq.NEXTVAL	Unique sequence-generated database id.
PREFERENCE_TYPE_CODE	'WDE'	Indicates that this is a WebADE internal preference.
PREFERENCE_SUB_TYPE	'app-config'	Indicates that this is an application configuration preference.
APPLICATION_ACRONYM	*NULL*	No application acronym makes this a default preference.
PREFERENCE_SET_NAME	*NULL*	This preference is a standalone preference, and should have no set name
PREFERENCE_NAME	'group-max-cache-time'	The preference name
PREFERENCE_VALUE	*'86400000'*	The cache time, in milliseconds (86400000 ms is 24 hours)
EUSER_ID	*NULL*	This column is only used for preferences of type 'USR'
CREATED_BY	*'db_admin'*	The database user creating the record.
CREATED_DATE	*SYSDATE*	Current time and date.
UPDATED_BY	*'db_admin'*	The database user creating the record.
UPDATED_DATE	*SYSDATE*	Current time and date.
REVISION_COUNT	*1*	Increment this value every time this record is edited.

Sample 'group-max-cache-time' Preference Record

3.3   WEBADE USER PROVIDERS

WebADE 4.1, there is a new concept of user providers. A user provider is a pluggable component that can look up user information on behalf of the WebADE. WebADE requires at least one user provider to be configured.

Ministry applications use CITS' CAP web services to query user information. Developers, if developing off-site, can still point to these web services for testing purposes, but they are required to be connected to the BC Government VPN. If testing using the VPN is prohibitive, you may wish to use the WebADE-Developer Module.

As of version 4.3 WebADE supports two versions of the CAP Web Services User Provider, V4 and V8.

3.3.1   CAP WEB SERVICES USER PROVIDER VERSION 4 (V4)

The V4 CAP web services user provider uses version 4 of the BCIED Web Service. It requires a set of preferences, all of which share the same preference type, preference sub-type, and preference set name. Below is a description of the common column values:

COLUMN NAME	COLUMN VALUE	DESCRIPTION
PREFERENCE_ID	preference_seq.NEXTVAL	Unique sequence-generated database id.
PREFERENCE_TYPE_CODE	'WDE'	Indicates that this is a WebADE internal preference.
PREFERENCE_SUB_TYPE	'webade-user-provider'	Indicates that this is a WebADE user provider preference.
APPLICATION_ACRONYM	NULL	No application acronym makes this a default preference.
PREFERENCE_SET_NAME	'cap-web-services-provider-v4'	All preferences for a particular user provider should have a common PREFERENCE_SET_NAME value.
PREFERENCE_NAME	PREFERENCE NAME	The preference name
PREFERENCE_VALUE	PREFERENCE VALUE	The preference value
EUSER_ID	NULL	This column is only used for preferences of type 'USR'
CREATED_BY	USER	The database user creating the record.
CREATED_DATE	SYSDATE	Current time and date.
UPDATED_BY	USER	The database user creating the record.
UPDATED_DATE	SYSDATE	Current time and date.
REVISION_COUNT	1	Increment this value every time this record is edited.

CAP Web Services Common Column Values

Described below is the set of preferences required for the V4 CAP web services user provider:

PREFERENCE NAME	MANDATORY	DESCRIPTION	PREFERENCE VALUE
class-name	Y	Class name for the WebADE BCeID web services provider.	ca.bc.gov.webade.user.provider.capws.v4.CAPWebServicesProvider
supported-user-types	Y	Comma separated list of user type codes (GOV, BUP, or UIN). Indicates that the WebADE BCeID web services provider supports user info lookups for these user types.	'GOV,BUP,UIN'
GOV-user-type-domain	Y	Specifies the user directory for government users.	'IDIR'
BUP-user-type-domain	Y	Specifies the user directory for business partner users.	'BCEID'

UIN-user-type-domain	Y	Specifies the user directory for unverified individual users.	*'MYID'*
cap-web-services-provider-url	Y	The URL of the CAP web services. (The test web services URL is "*https://gws1.test.bceid.ca/enhancedwebservices/enhancedservices.asmx*")	*'https://gws1.bceid.ca/enhancedwebservices/enhancedservices.asmx'*
cap-web-services-seconds-timeout	Y	The number of seconds to wait for a response from the BCeID web services before a timeout occurs.	*'60'*
cap-web-services-connect-username	Y	The connect IDIR username used to authenticate against to the new BCEID web services. **NOTE:** Do <u>not</u> prefix the username with IDIR.	*username*
cap-web-services-connect-password	Y	The connect password used to authenticate against to the new BCEID web services.	*password*
system-user-type-code	Y	The user type code (GOV, BUP, or UIN) of a user to be used as the requesting user of the CAP web services requests in a WebADE application request where there is no associated current user, such as an automated process.	'GOV', 'BUP', or 'UIN'
system-user-guid	Y	The user's GUID (a 32 character hexadecimal value, e.g. '9E0DEC464BE047C8A398010275675BAE') of a user to be used as the requesting user of the CAP web services requests in a WebADE application request where there is no associated current user, such as an automated process.	*User's guid*
system-user-account-name	Y	The user's full account name (e.g. IDIR\JDOE) of a user to be used as the requesting user of the CAP web services requests in a WebADE application request where there is no associated current user, such as an automated process.	*Domain/account_name*
enabled	N	Set to 'false' to disable this user provider. This allows multiple providers to be configured, but only one enabled (as the default or for a particular application).	*'true'*

V4 CAP Web Services Preferences

3.3.2   CAP WEB SERVICES USER PROVIDER VERSION 8 (V8)

The V8 CAP web services user provider uses version 8 of the BCIED Web Service. It requires a set of preferences, all of which share the same preference type, preference sub-type, and preference set name.

A significant improvement of the V8 provider is the support for Verified Individual users (user type code VIN, directory VFDID).

The V8 user provider also requires that all applications using it have an online service ID (OSID) configured via a preference.

Below is a description of the common column values:

COLUMN NAME	COLUMN VALUE	DESCRIPTION
PREFERENCE_ID	preference_seq.NEXTVAL	Unique sequence-generated database id.
PREFERENCE_TYPE_CODE	'WDE'	Indicates that this is a WebADE internal preference.
PREFERENCE_SUB_TYPE	'webade-user-provider'	Indicates that this is a WebADE user provider preference.
APPLICATION_ACRONYM	*NULL*	No application acronym makes this a default preference.
PREFERENCE_SET_NAME	'cap-web-services-provider-v8'	All preferences for a particular user provider should have a common PREFERENCE_SET_NAME value.
PREFERENCE_NAME	*PREFERENCE NAME*	The preference name
PREFERENCE_VALUE	*PREFERENCE VALUE*	The preference value
EUSER_ID	*NULL*	This column is only used for preferences of type 'USR'
CREATED_BY	*USER*	The database user creating the record.
CREATED_DATE	*SYSDATE*	Current time and date.
UPDATED_BY	*USER*	The database user creating the record.
UPDATED_DATE	*SYSDATE*	Current time and date.
REVISION_COUNT	*1*	Increment this value every time this record is edited.

CAP Web Services Common Column Values

Described below is the set of preferences required for the CAP V8 web services user provider:

PREFERENCE NAME	MANDATORY	DESCRIPTION	PREFERENCE VALUE
class-name	Y	Class name for the WebADE BCeID web services provider.	ca.bc.gov.webade.user.provider.capws.v8.CAPWebServicesProvider
supported-user-types	Y	Comma separated list of user type codes (GOV, BUP, UIN, or VIN). Indicates that the WebADE BCeID web services provider supports user info lookups for these user types.	'GOV,BUP,UIN,VIN'
GOV-user-type-domain	Y	Specifies the user directory for government users.	*'IDIR'*

BUP-user-type-domain	Y	Specifies the user directory for business partner users.	*'BCEID'*
UIN-user-type-domain	Y	Specifies the user directory for unverified individual users.	*'MYID'*
cap-web-services-provider-url	Y	The URL of the CAP web services. (The test web services URL is "*https://gws1.test.bceid.ca/webservices/Client/V8/BCeIDService.asmx*")	*'https://gws1.bceid.ca/webservices/Client/V8/BCeIDService.asmx'*
cap-web-services-seconds-timeout	Y	The number of seconds to wait for a response from the BCeID web services before a timeout occurs.	*'60'*
cap-web-services-connect-username	Y	The connect IDIR username used to authenticate against to the new BCEID web services. **NOTE:** Do <u>not</u> prefix the username with IDIR.	*username*
cap-web-services-connect-password	Y	The connect password used to authenticate against to the new BCEID web services.	*password*
system-user-type-code	Y	The user type code (GOV, BUP, UIN, or VIN) of a user to be used as the requesting user of the CAP web services requests in a WebADE application request where there is no associated current user, such as an automated process.	'GOV', 'BUP', 'UIN', or 'VIN'
system-user-guid	Y	The user's GUID (a 32 character hexadecimal value, e.g. '9E0DEC464BE047C8A398010275675BAE') of a user to be used as the requesting user of the CAP web services requests in a WebADE application request where there is no associated current user, such as an automated process.	*User's guid*
system-user-account-name	Y	The user's full account name (e.g. IDIR\JDOE) of a user to be used as the requesting user of the CAP web services requests in a WebADE application request where there is no associated current user, such as an automated process.	*Domain/account_name*
enabled	N	Set to 'false' to disable this user provider. This allows multiple providers to be configured, but only one enabled (as the default or for a particular application).	*'true'*

V8 CAP Web Services Preferences

In addition, each application must be configured with an online service ID (OSID):

COLUMN NAME	COLUMN VALUE	DESCRIPTION
APPLICATION_ACRONYM	*APPLICATION ACRONYM*	The application acronym that the OSID belongs to.
PREFERENCE_NAME	*'cap-web-services-online-service-id'*	The preference name
PREFERENCE_VALUE	*PREFERENCE VALUE*	The OSID for this application.

CAP Web Services Preference Values

3.4 WEBADE DEFAULT PREFERENCES

As all WebADE applications require certain application preference settings to properly function, and many of these preferences are identical for all applications, it is recommended that a set of default preferences be set to reduce the amount of configuration required for each application. For more information on default preferences, see the sub-section Default Preference Settings of the WebADE Preferences section of this document.

3.4.1 EXTENSION DEFAULT PREFERENCES

WebADE Extensions are now configured via the preferences table. As most extensions are configured the same for each application, it is recommended to make these configurations default preferences. This allows applications to only have to declare that they use these extensions to use them, and not have to configure them, if it isn't needed.

To explain what that means, we'll give a simple example with the Reporting extension. To configure an extension for use in a WebADE application, you must perform two preference table entries; one set of preferences configuring the extension, and a matching preference that declares that the application uses the extension.

3.4.2 CONFIGURING AN EXTENSION'S PREFERENCES

To configure any extension, there is a mandatory preference, the 'extension-class-name'. This preference is configured as described below.

COLUMN NAME	COLUMN VALUE	DESCRIPTION
PREFERENCE_ID	preference_seq.NEXTVAL	Unique sequence-generated database id.
PREFERENCE_TYPE_CODE	'EXT'	Indicates that this is an extension preference.
PREFERENCE_SUB_TYPE	*'extension-name'*	This is a unique extension name that should be defined by the extension as a standard name,

		but may be any value. Examples are: 'reporting', 'services', and 'mailscheduler'. This value is to be set as the sub-type value for all configuration preferences and preference sets used to configure this extension.
APPLICATION_ACRONYM	*NULL*	No application acronym makes this a default preference.
PREFERENCE_SET_NAME	*NULL*	The 'extension-class-name' preference is a standalone preference.
PREFERENCE_NAME	'extension-class-name'	The preference name
PREFERENCE_VALUE	*'ca.bc.gov.mof.MyExtensionClass'*	The fully-qualified class name for the target extension.
EUSER_ID	*NULL*	This column is only used for preferences of type 'USR'
CREATED_BY	*'db_admin'*	The database user creating the record.
CREATED_DATE	*SYSDATE*	Current time and date.
UPDATED_BY	*'db_admin'*	The database user creating the record.
UPDATED_DATE	*SYSDATE*	Current time and date.
REVISION_COUNT	*1*	Increment this value every time this record is edited.

Sample 'extension-class-name' Preference Record

In addition, you may also define any preference and preference sets that the extension needs to be configured. These preferences should all be of preference type 'EXT' and their preference sub-type MUST match the sub-type value of the 'extension-class-name' preference. If you wish these preferences to be default preferences, leave the application_acronym column as null.

To actually register the extension with a WebADE application, you must define a matching application preference as described below.

COLUMN NAME	COLUMN VALUE	DESCRIPTION
PREFERENCE_ID	preference_seq.NEXTVAL	Unique sequence-generated database id.
PREFERENCE_TYPE_CODE	'APP'	Indicates that this is an application preference.
PREFERENCE_SUB_TYPE	'app-config'	Indicates that this is an application configuration preference.
APPLICATION_ACRONYM	*APP_ACRONYM*	This column should never be null, as this would make it a default preference, and would result in ALL WebADE applications connected to that WebADE database to load that extension.
PREFERENCE_SET_NAME	'extensions'	The 'extensions' preference set contains all extension declarations.
PREFERENCE_NAME	'extension'	The preference name. All extensions that an application will use should have the same preference type, sub-type, set name, and preference name. This means that the 'extension' preference will be assigned multiple values (Which is perfectly legal in this implementation

		of WebADE).
PREFERENCE_VALUE	*'extension-name'*	This name should match the *'extension-name'* preference sub-type value for the target extension, as described above in Configuring an Extension's Preferences.
EUSER_ID	*NULL*	This column is only used for preferences of type 'USR'
CREATED_BY	*'db_admin'*	The database user creating the record.
CREATED_DATE	*SYSDATE*	Current time and date.
UPDATED_BY	*'db_admin'*	The database user creating the record.
UPDATED_DATE	*SYSDATE*	Current time and date.
REVISION_COUNT	*1*	Increment this value every time this record is edited.

Sample 'extension' Preference Set Record

MOF WEBADE 4 EXTENSIONS

NOTE: These extensions are only available for applications in development for MOF. If you are not developing for MOF, you will not be able to use these extensions.

MOF currently supports the following WebADE 4 extensions:
- WebADE Reporting Extension
- WebADE Task Manager Extension
- WebADE Mail Scheduler Extension

WEBADE REPORTING EXTENSION PREFERENCES

WebADE Reporting Extension requires the following EXT type preferences:

PREFERENCE NAME	MANDATORY	DESCRIPTION	PREFERENCE VALUE
extension-class-name	Y	The extension class name	ca.bc.gov.mof.webade.reportgeneration.ADEReportManager

WebADE Reporting Extension preferences

WEBADE TASK MANAGER EXTENSION PREFERENCES

WebADE Task Manager Extension requires the following EXT type preferences:

PREFERENCE NAME	MANDATORY	DESCRIPTION	PREFERENCE VALUE
extension-class-name	Y	The extension class name	ca.bc.gov.webade.extension.taskmanager.TaskManagerExtension
load-file-as-resource	N	Indicates whether the file will be loaded as a resource or as a file from a local file location. This defaults to true, but should be set to false if you wish the file to be located somewhere other than WEB-INF/classes/tasks.xml	'true' *or* 'false'

| tasks-file-rewritable | N | Defaults to false, but should be set to true if your wish the task manager to obscure securable properties in the file. Cannot be set to true if the file is loaded as a resource. | 'true' or 'false' |
| tasks-file-location | N | This preference is used when the 'load-file-as-resource' preference is false. This will define the location of the tasks.xml file, usually outside of the web application's directory structure, in a local directory. | *The local fully-qualified path to the tasks.xml file.* |

WebADE Task Manager Extension preferences

WEBADE MAIL SCHEDULER EXTENSION PREFERENCES

WebADE Mail Scheduler Extension requires the following EXT type preferences:

PREFERENCE NAME	MANDATORY	DESCRIPTION	PREFERENCE VALUE
extension-class-name	Y	The extension class name	ca.bc.gov.mof.webade.mailscheduler.MailSchedulerManager
SmtpHost	N	Use this preference if you wish to use an SMTP host other than smtp.gov.bc.ca.	*The server name of the SMTP host server.*

WebADE Mail Scheduler Extension preferences

MOF WEBADE 3 EXTENSIONS

NOTE: These extensions are only available for WebADE 3 applications in development for MOF. If you are not developing or maintaining a WebADE 3 application for MOF, you will not be able to use these extensions.

MOF currently supports the following WebADE 3 extensions, using the webade-legacy library JAR to run a WebADE 3 application in the new WebADE 4 environment:
- WebADE Services Extension
- WebADE User Info Extension

WEBADE SERVICES EXTENSION PREFERENCES

WebADE Services Extension requires the following EXT type preferences:

PREFERENCE NAME	MANDATORY	DESCRIPTION	PREFERENCE VALUE
extension-class-name	Y	The extension class name	ca.bc.gov.mof.webade.services.ServiceManager

WEBADE USER INFO EXTENSION PREFERENCES

WebADE User Info Extension requires the following EXT type preferences:

PREFERENCE NAME	MANDATORY	DESCRIPTION	PREFERENCE VALUE
extension-class-name	Y	The extension class name	ca.bc.gov.mof.webade.user.UserInfoManager

WebADE User Info Extension preferences

4. CONFIGURING A NEW WEBADE APPLICATION

4.1   INSTALLING WEBADE APPLICATIONS

This section describes how to deploy and configure an application developed using the WebADE framework. This is just a general overview. For more comprehensive detail, please consult your J2EE application server's documentation.

4.1.1   DEPLOYMENT

Please consult your J2EE application server's documentation for instructions on deploying J2EE application EAR and WAR files.

4.1.2   CONFIGURATION

The following steps must be completed to ensure proper execution of the web application:

Ensure that the application has appropriate entries in the WebADE Application Database
Create the authorizations in ADAM for users that will access the application in the web server domain.

4.1.3   WEBADE EXTENSIONS

Some WebADE applications require the use of extensions to provide additional functionality to the WebADE. These extensions are configured by putting the proper entries in the PREFERENCES table for the target WebADE application.

See the User's Guide for more information on how to configure the PREFERENCES table.

4.2   USER AGREEMENTS

An application may be configured to require users to accept a user agreement before using the application. If this feature is configured for an application and a user has not yet accepted the agreement, the request will be redirected to a user agreement page. Without accepting the agreement, the user will not be allowed access to the application.

4.2.1   CREATE USER AGREEMENT

The agreement must be created in the ELECTRONIC_AGREEMENT table.

PREFERENCE COLUMN	VALUE
ELECTRONIC_AGREEMENT_ID	preference_seq.NEXTVAL
AGREEMENT_TYPE_CODE	'USR'
ELECTRONIC_AGREEMENT_NAME	The unique name of the user agreement. You will specify this name when creating the preference to enable the agreement.
AGREEMENT_TEXT	The text of the agreement that the user must accept.
EFFECTIVE_DATE	The date on and after which the agreement will be required.
EXPIRY_DATE	The date after which the agreement will no longer be required. Set to '9999-12-31' if the agreement never expires.
CREATED_BY	'*db_admin*'
CREATED_DATE	SYSDATE
REVISION_COUNT	1

Electronic Agreement Column Values

4.2.2   CONFIGURE THE AGREEMENT FOR APPLICATIONS

In order to enable the agreement for applications you must create a preference specifying which user agreement is to be enabled and for which application.

PREFERENCE COLUMN	VALUE
PREFERENCE_ID	preference_seq.NEXTVAL
PREFERENCE_TYPE_CODE	'APP'
PREFERENCE_SUB_TYPE	'app-config'
PREFERENCE_SET_NAME	NULL
PREFERENCE_NAME	'webade-agreement'
PREFERENCE_VALUE	Set to the same value used in ELECTRONIC_AGREEMENT_NAME above.
APPLICATION_ACRONYM	Set to the application acronym of the target WebADE application this preference is for (Exception, see Default Preference Settings).
EUSER_ID	NULL
CREATED_BY	'*db_admin*'
CREATED_DATE	SYSDATE
UPDATED_BY	'*db_admin*'
UPDATED_DATE	SYSDATE
REVISION_COUNT	1

User Agreement Preference Column Values

4.3   APPLICATION ENABLED FLAG

Applications may be enabled or disabled via the ENABLED_IND column in the APPLICATION table. Set to 'Y' to enable or 'N' to disable. On each request, if an application is disabled and a user tries to access it, the request will be redirected to an application disabled page. By default, WebADE does not check ENABLED_IND; it is ignored.

For all the preferences described in this section all values are the same, except for the name and value.

PREFERENCE COLUMN	VALUE
PREFERENCE_ID	preference_seq.NEXTVAL
PREFERENCE_TYPE_CODE	'APP'
PREFERENCE_SUB_TYPE	'app-config'
PREFERENCE_SET_NAME	NULL
APPLICATION_ACRONYM	Set to the application acronym of the target WebADE application this preference is for (Exception, see Default Preference Settings).
EUSER_ID	NULL
CREATED_BY	'*db_admin*'
CREATED_DATE	SYSDATE
UPDATED_BY	'*db_admin*'
UPDATED_DATE	SYSDATE
REVISION_COUNT	1

Application Enabled Preferences Column Values

4.3.1   ENABLING THE CHECK FOR DISABLED APPLICATIONS

To configure WebADE to check the enabled flag, you must create a preference. If the preference value is set to true, then the flag will be checked.

PREFERENCE COLUMN	VALUE
PREFERENCE_NAME	'webade-application-check-for-disabled-flag'
PREFERENCE_VALUE	Set to 'true' to enable this feature or 'false' to disable it.

Application Check for Disabled Flag Preference Column Values

4.3.2   OVERRIDING THE APPLICATION DISABLED MESSAGE

If an application is disabled, by default a standard WebADE error page is displayed with a generic message. This preference uses the default WebADE error page layout, but replaces the generic message with the value of the preference.

PREFERENCE COLUMN	VALUE
PREFERENCE_NAME	'webade-application-disabled-message'
PREFERENCE_VALUE	Set to the text of the message to be displayed on the application disabled page.

Application Disabled Message Preference Column Values

4.3.3   OVERRIDING THE APPLICATION DISABLED PAGE

If an application is disabled, by default a standard WebADE error page is displayed with a generic message. This preference overrides the default page with one inside the application's EAR, using the preference value as the relative path to the page. For example, this could be a JSP, servlet, or struts action path.

PREFERENCE COLUMN	VALUE
PREFERENCE_NAME	'webade-application-disabled-page'
PREFERENCE_VALUE	Set to the relative path of the application disabled page.

Application Disabled Page Preference Column Values

4.4   ORGANIZATION SELECTION

Authorization Profiles granted to users may be secured by organization. In other words, a user granted that profile will be able to perform the actions of the profile on behalf of a specific organization. Users can be granted authorization to multiple profiles. If the profiles are secured by different organizations, the user has the ability to act on behalf of multiple organizations. As such, WebADE can be configured to allow the user to select which organization on behalf of which they want to act.

For all the preferences described in this section, all values are the same, except for the name and value.

PREFERENCE COLUMN	VALUE
PREFERENCE_ID	preference_seq.NEXTVAL
PREFERENCE_TYPE_CODE	'APP'
PREFERENCE_SUB_TYPE	'app-config'
PREFERENCE_SET_NAME	NULL
APPLICATION_ACRONYM	Set to the application acronym of the target WebADE application this preference is for (Exception, see Default Preference Settings).
EUSER_ID	NULL
CREATED_BY	'*db_admin*'
CREATED_DATE	SYSDATE
UPDATED_BY	'*db_admin*'
UPDATED_DATE	SYSDATE
REVISION_COUNT	1

Organization Selection Preferences Column Values

4.4.1   ENABLING ORGANIZATION SELECTION

To configure WebADE to allow users to select an organization, you must create a preference to enable this. If the preference value is set to true, when the user begins their session the request will be redirected to an organization selection page. However, if the user has no profiles secured by organization or only one, then the user will not be redirected to this page.

The Organization Selection page allows the user to select which organization to act on behalf of for the duration of their session. They can also indicate that their selection is their default organization. If the user indicates a default, then when starting future sessions WebADE will remember it and preselect it on the selection page.

PREFERENCE COLUMN	VALUE
PREFERENCE_NAME	'webade-default-organization-enabled'
PREFERENCE_VALUE	Set to 'true' to enable this feature or 'false' to disable it.

Organization Selection Preference Column Values

4.4.2   USING THE DEFAULT ORGANIZATION WITHOUT PROMPTING

In some cases, it may not make sense to prompt the user for organization selection each time they start their session. If this preference is enabled and the user has selected a default organization, then request will not be redirected to the organization selection page. The selection page will be skipped entirely.

PREFERENCE COLUMN	VALUE
PREFERENCE_NAME	'webade-use-default-organization-enabled'
PREFERENCE_VALUE	Set to 'true' to enable this feature or 'false' to disable it.

"Use Default" Preference Column Values

4.4.3   FILTERING ORGANIZATIONS BY TYPE

By default, all organizations on behalf of which a user is authorized to act will listed on the organization selection page. This list can be filtered to show only government organizations or only non-government organizations.

PREFERENCE COLUMN	VALUE
PREFERENCE_NAME	'webade-default-organization-select-by-organization-type'
PREFERENCE_VALUE	Set to 'government' to list only government, 'non-government' to list only non-government or 'all' to list all organizations (default).

Filter Organizations Preference Column Values

4.4.4 OVERRIDING THE ORGANIZATION SELECTION PAGE

By default, a standard WebADE organization selection page is displayed. This preference overrides the default selection page with one inside the application's EAR, using the preference value as the relative path to the page. For example, this could be a JSP, servlet, or struts action path.

PREFERENCE COLUMN	VALUE
PREFERENCE_NAME	'webade-default-organization-switch-page'
PREFERENCE_VALUE	Set to 'true' to enable this feature or 'false' to disable it.

Selection Page Preference Column Values

5. WEBADE PREFERENCES

WebADE allows the setting and retrieval of five types of preferences: Global, Application, WebADE, Extension, and User. These preferences are all stored in the PREFERENCE table, and, as such, have different constraints on that table. Each preference has a type, sub-type, name, value, plus an optional preference set name. In addition, some preferences can have an application acronym and/or an EUser id, associating the preference with a specific WebADE application and/or user.

Below is a description of each preference type, the uses for each type, and the allowable values for the columns in the database.

5.1   GLOBAL PREFERENCES

Global Preferences are preferences that are not related to WebADE applications specifically, allowing preferences that could apply to the entire WebADE system. Such preferences could be standard URL locations, email addresses, or other such preferences.

As a Global Preference, there is no valid context of an application or user, so the APPLICATION_ACRONYM and EUSER_ID columns must be null for these preferences.

PREFERENCE COLUMN	DESCRIPTION
PREFERENCE_ID	Unique sequence generated id for the preference record.
PREFERENCE_TYPE_CODE	Set to "GLB" for Global Preferences.
PREFERENCE_SUB_TYPE	Describes a categorized sub-type for easy grouping of like-preferences and preference sets.
PREFERENCE_SET_NAME	(Optional) Groups a set of preferences by the set name.
PREFERENCE_NAME	The target name of the preference. This name should be unique within the set of preferences assigned the same Preference Sub Type. If the Preference Set Name is set, then this name should instead be unique within the set of preferences assigned the same Preference Sub Type and Preference Set Name.
PREFERENCE_VALUE	The value of the preference.
APPLICATION_ACRONYM	Always NULL for Global Preferences.
EUSER_ID	Always NULL for Global Preferences.
REVISION_COUNT	A counter that is incremented every time the record is updated.

Global Preference Column Uses

5.2   APPLICATION PREFERENCES

Application Preferences are preferences that are used by WebADE or WebADE applications, defining any sort of application-specific configuration setting, not including WebADE extension settings.

Application preferences should never have the EUSER_ID column set, as these settings are application-wide and have no direct bearing on a particular user. All

Application Preferences should have a properly set APPLICATION_ACRONYM column setting, assigning the preference to a particular WebADE application.

PREFERENCE COLUMN	DESCRIPTION
PREFERENCE_ID	Unique sequence generated id for the preference record.
PREFERENCE_TYPE_CODE	Set to "APP" for Application Preferences.
PREFERENCE_SUB_TYPE	Describes a categorized sub-type for easy grouping of like-preferences and preference sets.
PREFERENCE_SET_NAME	(Optional) Groups a set of preferences by the set name.
PREFERENCE_NAME	The target name of the preference. This name should be unique within the set of preferences assigned the same Preference Sub Type. If the Preference Set Name is set, then this name should instead be unique within the set of preferences assigned the same Preference Sub Type and Preference Set Name.
PREFERENCE_VALUE	The value of the preference.
APPLICATION_ACRONYM	Set to the application acronym of the target WebADE application this preference is for (Exception, see Default Preference Settings below).
EUSER_ID	Always NULL for Application Preferences.
REVISION_COUNT	A counter that is incremented every time the record is updated.

Application Preference Column Uses

5.3   WEBADE PREFERENCES

WebADE Preferences are preferences that are used by WebADE internally, and should not be set by WebADE applications.

WebADE preferences should never have the EUSER_ID column set, as these settings are application-wide and have no direct bearing on a particular user. All WebADE Preferences can have a null value or properly set APPLICATION_ACRONYM column setting, assigning the preference to a particular WebADE application.

PREFERENCE COLUMN	DESCRIPTION
PREFERENCE_ID	Unique sequence generated id for the preference record.
PREFERENCE_TYPE_CODE	Set to "WDE" for WebADE Preferences.
PREFERENCE_SUB_TYPE	Describes a categorized sub-type for easy grouping of like-preferences and preference sets.
PREFERENCE_SET_NAME	(Optional) Groups a set of preferences by the set name.
PREFERENCE_NAME	The target name of the preference. This name should be unique within the set of preferences assigned the same Preference Sub Type. If the Preference Set Name is set, then this name should instead be unique within the set of preferences assigned the same Preference Sub Type and Preference Set Name.
PREFERENCE_VALUE	The value of the preference.
APPLICATION_ACRONYM	Set to the application acronym of the target WebADE application this preference is for (Exception, see Default Preference Settings below).
EUSER_ID	Always NULL for WebADE Preferences.
REVISION_COUNT	A counter that is incremented every time the record is updated.

WebADE Preference Column Uses

5.4 EXTENSION PREFERENCES

Extension Preferences are preferences that are used by WebADE Extensions, defining any sort of extension-specific configuration settings.

Extension preferences should never have the EUSER_ID column set, as these settings are Extension-specific and have no direct bearing on a particular user. All Extension Preferences should have a properly set APPLICATION_ACRONYM column setting, assigning the preference to a particular WebADE application.

PREFERENCE COLUMN	DESCRIPTION
PREFERENCE_ID	Unique sequence generated id for the preference record.
PREFERENCE_TYPE_CODE	Set to "EXT" for Extension Preferences.
PREFERENCE_SUB_TYPE	Describes a categorized sub-type for easy grouping of like-preferences and preference sets.
PREFERENCE_SET_NAME	(Optional) Groups a set of preferences by the set name.
PREFERENCE_NAME	The target name of the preference. This name should be unique within the set of preferences assigned the same Preference Sub Type. If the Preference Set Name is set, then this name should instead be unique within the set of preferences assigned the same Preference Sub Type and Preference Set Name.
PREFERENCE_VALUE	The value of the preference.
APPLICATION_ACRONYM	Set to the application acronym of the target WebADE application this preference is for (Exception, see Default Preference Settings below).
EUSER_ID	Always NULL for Extension Preferences.
REVISION_COUNT	A counter that is incremented every time the record is updated.

Extension Preference Column Uses

5.5 USER PREFERENCES

User Preferences are preferences that are used by WebADE to determine user-specific preferences assigned to the user that has created a session with the WebADE application.

User preferences will usually have the EUSER_ID column set, as these settings are user-specific, allowing each user of an application to have a custom set of settings. All User Preferences should have a properly set APPLICATION_ACRONYM column setting, assigning the preference to a particular WebADE application.

PREFERENCE COLUMN	DESCRIPTION
PREFERENCE_ID	Unique sequence generated id for the preference record.
PREFERENCE_TYPE_CODE	Set to "USR" for User Preferences.
PREFERENCE_SUB_TYPE	Describes a categorized sub-type for easy grouping of like-preferences and preference sets.
PREFERENCE_SET_NAME	(Optional) Groups a set of preferences by the set name.
PREFERENCE_NAME	The target name of the preference. This name should be unique within the set of preferences assigned the same Preference Sub Type. If the Preference Set Name is set, then this name should instead be unique within the set of preferences assigned the same Preference Sub Type and Preference Set Name.
PREFERENCE_VALUE	The value of the preference.

APPLICATION_ACRONYM	Set to the application acronym of the target WebADE application this preference is for (Exception, see Default Preference Settings below).
EUSER_ID	Set to the EUSER_ID of the target WebADE user this preference is for (Exception, see Default Preference Settings below).
REVISION_COUNT	A counter that is incremented every time the record is updated.

User Preference Column Uses

5.6 DEFAULT PREFERENCE SETTINGS

Individual Application, WebADE, Extension, and User Preferences (In or out of a Preference Set) can be assigned a "default" value. A default value is one that is used when the preference is not defined for a specific application. This allows common preferences that are replicated in many applications to be defined once, instead of adding and managing an identical record for each application. To make a preference a default preference, do not assign it an application acronym, instead leave the APPLICATION_ACRONYM column blank.

If a particular application wishes to set the preference to a value other than the default, it only needs to create a preference record in the table with identical values for the PREFERENCE_TYPE_CODE, PREFERENCE_SUB_TYPE, PREFERENCE_SET_NAME, PREFERENCE_NAME columns, assign the preference to the target APPLICATION_ACRONYM, and set the PREFERENCE_VALUE column to the desired value. When a preference is defined twice, once as a default and once specific to the application, the application-specific value is always returned.

DEFAULT USER PREFERENCES

User Preferences can also be defined as "default" with respect to user. If a User Preference has no EUSER_ID set, then the preference value is used for a user if no User Preference of the same type has been specifically set for the target user and target WebADE application.

Also, a User Preference can have both the EUSER_ID and APPLICATION_ACRONYM columns set to null. Default User Preferences of this type apply to all WebADE applications for all users that are do not have a user-specific preference set with the same identifying preference settings.

As this allows four potential User Preference records that match the user, it might seem a little cloudy which default preference takes precedence when multiple matches are found for a target User Preference. The following table summarizes the default User Preference hierarchy.

EUSER_ID	APPLICATION_ACRONYM	PRIORITY	DESCRIPTION
Defined	*Defined*	1	A record with a defined user and application acronym will always be used first to return the User Preference value.
Defined	*NULL*	2	If no record exists with a defined user and acronym, but one exists with a defined user and no application acronym, this record's preference value is used instead.

| NULL | Defined | 3 | If the first two records do not exist, but a default User Preference record exists with a NULL user id and the target application acronym value set, this preference value is used. |
| NULL | NULL | 4 | If no other matching User Preference record can be found, the NULL user and application acronym User Preference value is used. |

User Preference Hierarchy

6. DYNAMIC FILTERS

As of WebADE 4.2.0, applications can now add javax.servlet.Filter implementations to an internal filter chain managed by the WebADEFilter. The purpose of this internally-managed filter chain is to allow the addition and removal of filters to your application without having to make changes to the web.xml file or having to do a redeployment of the application's EAR. This is useful for a couple of specific situations:

- The use of the WebADE Developer Module Filter to handle application authentication in a non-ministry environment.
- The addition of debugging and logging filters to troubleshoot an already deployed application.

6.1   WEBADE DEVELOPER MODULE FILTER

For previous versions of the WebADE API, developers using the WebADE Developer Module to mimic SiteMinder for authentication would have to explicitly declare the WebADE Developer Module Filter in the web.xml, making sure it was mapped to the same URLs or Servlets that the WebADE Filter itself was mapped to. This would require a web.xml file for local development that would then have to be modified to remove the Developer Module tags before delivering the application to the ministry.

With dynamic filters, the Developer Module Filter can be configured in the WebADE preference table, allowing the WebADE Filter to load the Developer Module Filter at runtime and execute it before the WebADE Filter code itself is run. This allows the application to be migrated to a ministry environment without having to modify application source.

6.2   DEBUGGING AND LOGGING FILTERS

The other main purpose of dynamic filters is to allow an administrator or developer to add or remove filters on the fly to an application, without have to restart or redeploy the application. This allows for on-the-fly analysis of an application that is not behaving as expected.

In a later release of the WebADE, we will include a number of pre-built utility filters to aid developers and administrators with this analysis. For now, it is up to the developer to create any utility filters that they need to analyse the web application environment.

6.3   MANAGING THE WEBADE DYNAMIC FILTER CHAIN

Adding or removing filters to the WebADE Filter-managed filter chain requires the addition/removal of certain preferences in the WebADE Preference table.

6.3.1 DYNAMIC FILTER WEBADE PREFERENCES

Each filter to be added to the WebADE filter chain must define a set of WebADE preferences. The set of preferences for each filter must all share the following attributes:

PREFERENCE COLUMN	PREFERENCE COLUMN VALUE
PREFERENCE_TYPE_CODE	"WDE"
PREFERENCE_SUB_TYPE	"dynamic-filter"
PREFERENCE_SET_NAME	Developer defined. All preferences for a dynamic filter should share the same preference set name.
APPLICATION_ACRONYM	Set to the target application's application acronym, or NULL to configure this filter for all WebADE applications connecting to this WebADE database.

Each set of dynamic filter preferences must contain the following preferences:

PREFERENCE NAME	VALID VALUES	DESCRIPTION
class-name	The fully qualified class name of the Filter implementation Java class.	Used by the WebADE to load an instance of the Filter class into memory. The Filter class must be locatable in the application's class path.
chain-order	Any integer, positive or negative, except for 0.	Indicates the ascending order in which the filters are executed on an incoming request. A negative value indicates that the dynamic filter is executed before the WebADE Filter itself. A positive value indicates that the dynamic filter is to be executed after the WebADE performs its own execution, parsing the user's credentials form the session and loading the user's info and permissions into session memory. **NOTE:** Each integer value in the chain order can only be used by at most one dynamic filter. If two dynamic filters are configured with the same chain order integer value for a given WebADE application, an error will occur at runtime.
enabled	"true" or "false"	If the enabled flag is set to true, the WebADE filter will call the configured dynamic filter in chain order. Otherwise it will be ignored.
init-param-*param-name*	Any String value.	Replace *param-name* in "init-param-*param-name*" with any desired string value. Preferences with the "init-param-" prefix will be passed in to the filter in the same way that <init-param> name value pairs are passed in to filters configured in the web.xml file. The "init-param-" prefix will be stripped off of the parameter name before it is passed into the filter.

7. WEBADE DATASTORE IMPLEMENTATION CONFIGURATION

The WebADE API allows for ministry-specific implementations of the internal class WebADEDatastore to override the Organization data returned in secure-by-organization authorizations a user may have been granted. For most ministries, the default Datastore implementation is sufficient for their applications. However, for certain ministries (Ministry of Forests and Range and Ministry of Finance are currently the only ministries whose developers might be affected), there may be additional ministry-specific organization information that a WebADE application requires access to. A ministry-specific datastore implementation can extend the WebADEDatabaseDatastore class and implement the abstract methods dealing with Organization data loading, returning a ministry-customized Organization class.

There are two ways to configure the WebADE to use a specialized datastore implementation, via a setting in the WebADEConnection class Jar file and via a WebADE preference.

7.1   WEBADECONNECTION JAR DATASTORE SETTING

To set the WebADEDatastore implementation via the WebADEConnection Java class, edit the WEBADE_DATASTORE_IMPLEMENTATION private static variable value in the ca.bc.gov.webade.datastore.WebADEConnection class, setting it to the fully-qualified class name of the desired WebADEDatastore implementation. Then compile and JAR this class to create your customized connection Jar, replacing the normal connection JAR you created during the WebADE installation process.

7.2   WEBADE PREFERENCES DATASTORE SETTING

The main problem with the connection JAR method of datastore configuration is that confusion can easily arise when you have separate connection JARs for ministry-specific organization and non-ministry-specific organization apps both connecting to the same WebADE database.

As of WebADE 4.2.0, you may alternatively define the datastore implementation via a WebADE configuration preference. A SQL script is now included in the WebADE release that will create a default WebADE preference for you, prompting you for the fully-qualified class name of the datastore implementation. This file is named "user_provider_preference_data.sql", and it can be found in the preferences_data sub-directory in the webade-04_02_00-database-install.zip file included in the webade-04_02_00-distribution.zip file.

NOTE: If you wish to set this datastore for a specific application (instead of all applications, as is the default), modify the SQL script before running it, replacing the NULL application_acronym column value with the application acronym string (in quotes) for the target application.

8. WEBADE TECHNICAL NOTES

8.1   ACTIVE DIRECTORY NOTES

As WebADE user information is created and stored in active directories, but is also cached in the ADAM database, this presents a synchronization issue. When a user's information is requested, WebADE will first look up the record via the WebADE user provider (i.e. CAP web services provider or WebADE Developer Module). If a record is found, WebADE will compare the user's information against the information cached in the WebADE database, updating or inserting a record into the WebADE database if this information is out-of-synch or non-existent.

9. TROUBLESHOOTING

9.1   LOGGING

WebADE can provide valuable debugging and error logs which are useful for troubleshooting issues with your WebADE deployment. If you encounter issues with your WebADE application deployment, we recommend that you turn on debug logging and analyze the information WebADE will write out to the logs.

9.1.1   ENABLING LOG4J LOGGING

The first step in configuring logging is to create a log4j.properties file. Here is an example of a log4j.properties file.

```
log4j.rootCategory=DEBUG, stdout

log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout

log4j.appender.stdout.layout.ConversionPattern=%x%-5p %c - %m%n

# specific category logging levels
#
log4j.category.ca.bc.gov.webade=INFO
log4j.category.org.apache.commons.httpclient=ERROR
log4j.category.org.apache.axis=ERROR
```

See the log4j website for instructions on setting a log4j.properties file.

9.1.2   LOG4J LOGGING IN OC4J

Configuring Log4J logging in OC4J depends upon the location of the commons-logging JAR in your application's classpath.

If the common-logging and log4j jars are included in your application's WEB-INF/lib directory, place the log4j.properties file in WEB-INF/classes directory of your application.

If the common-logging JAR is added to your application's classpath via an externally-linked <library> tag in your EAR's META-INF/orion-application.xml file, place the log4j.properties file the same external directory as the commons-logging JAR.

9.1.3   LOG4J LOGGING IN JRUN 4

NOTE: These instructions assume that your JRun 4 installation matches that of the MOFR JRun 4 standard setup. This includes adding the mof-excluded libraries to the SERVER-INF/lib of your JRun 4 app server.

The first step to configuring logging in your application, place the log4j.properties file in SERVER-INF/lib directory of the app server hosting your application, alongside the log4j JAR file.

For JRun 4, in addition to a log4j.properties file, you will also need to create a commons-logging.properties file in the SERVER-INF/lib directory with the following content in the properties file:

```
org.apache.commons.logging.Log=org.apache.commons.logging.impl.Log4Jlogger
```

This is due to a problem with the default JRun 4 settings that overrides the default commons-logging implementation which messes up the logging of the application. By adding the commons-logging.properties file to the SERVER-INF/lib, this restores the commons-logging back to the default implementation.

9.1.4   LOG4J LOGGING IN TOMCAT

To configure Log4J logging for an application deployed in Tomcat, place the log4j.properties file in WEB-INF/classes directory of your application.

9.1.5   LOG4J LOGGING IN ECLIPSE

To configure Log4J logging for an application you are working on in Eclipse, add the log4j.properties file to the root of one of the source directories in your Eclipse project's Java Build Path.

9.2   ANALYZING WEBADE-GENERATED LOGS

The WebADE outputs at various log levels, in order from lowest to highest: TRACE, DEBUG, INFO, WARN, ERROR, and FATAL. For example; if you set the log level to DEBUG, you will see logs of all levels from DEBUG to FATAL. **NOTE:** For older versions of Log4J, TRACE may be logged as DEBUG, and FATAL may be logged as ERROR.

In a production or test environment, you will probably want to set the log level to WARN or ERROR, lowering the log level to INFO or DEBUG when a problem arises. In a development environment, you may want to leave the log level at DEBUG, although you will have to manage your logs (A rolling log file is good for this situation), as they will grow large quickly with this setup.

Once you have logging enabled and set to the desired level, WebADE will output logs that will help you analyze what is going wrong with your application.

9.3   WEBADE DEBUG LOGS

In this section we will list useful log excerpts that you can use while deploying and running a WebADE application, to help troubleshoot issues you may encounter. The

log level of the excerpt will be stated, as well, before each logging excerpt, to help you set your log levels accordingly.

9.3.1   WEBADE CONNECTION JAR DETAILS

Log (Level: DEBUG):

```
Loaded WebADE connection details: WebADEConnectionDetails[
jdbcUrl=jdbc:oracle:thin:@MSRMORA:1521:DEV2, userid=webaded$proxy,
webadeDatastoreImplementation=ca.bc.gov.webade.DefaultWebADEDatabaseDatastore
, webadeEnvironment=DEV, minConnections=0, maxConnections=10,
maxConnectionIdleTime=10, maxConnectionCheckoutTime=10,
maxConnectionWaitTime=0, monitorSleepTime=1, poolConnectionsFlag=true,
pingConnectionsFlag=false ]
```

Purpose: This log line is written by the WebADE on start-up, and displays all of the connection settings used to create the WebADE database connection pool (except for the proxy database user password). This will allow you to confirm that you are using the correct settings, and that you are connected to the correct WebADE database.

9.3.2   CAP WEB SERVICES PROVIDER DETAILS

Log (Level: DEBUG):

```
Initialized BCeIDWebServicesProvider: BCeIDWebServicesProvider[
providerUrl=https://gws1.bceid.ca/enhancedwebservices/enhancedservices.asmx?W
SDL, username=sa_wbsvc, supportedTypes={ BUP<BCEID>, GOV<IDIR>, UIN<MYID> },
timeout=60000ms ]
```

Purpose: This log line is written by the WebADE on start-up if you are using the CAP web services to load user info. It displays all of the connection settings used to create the user provider and connect to the CAP web services (except for the service user account password). This will allow you to confirm that you are using the correct settings, and that you are connected to the correct WebADE database.

9.3.3   WEBADE WRAPPED CONNECTIONS

NOTE: WebADE's wrapped connection functionality places a wrapper Java class around the Oracle Connection, Statement and ResultSet objects before returning them to the developer (See the WebADE 4 - Connection Pooling Guide, included in this release, for more information).

The purpose for this is to reduce the risk of connections, statements, and result sets not being properly closed by the developer before the developer's code loses a reference to the database object (Usually due to leaving the block of code the database object was created/retrieved in). Without the wrapped connection functionality, it is a known problem with Oracle's JDBC pooled connection implementation that connections/statements/result sets are not collected by the

garbage collector and are never closed in this situation. The purpose of adding the wrapper class is to give the developer's code maintains the only reference to the WebADE wrapped connection object. This allows the garbage collector to clean up this wrapper class when the developer's code loses reference to this object. The wrapper class will then, at garbage collection time, close the wrapped Oracle database object properly, if it hasn't been already closed by the developer, eliminating this Oracle pool problem.

Because of this, WebADE will not close your connection on you until after you have lost reference to the wrapper connection class (Or you call close() on the WebADE connection wrapper class). If you obtain the wrapped Oracle connection class from the wrapper class by calling the getWrappedConnection() method on the wrapper class, you **must** keep the reference to the wrapper class until you are finished with the Oracle connection class and should use the wrapper class' close() method to close the connection, instead of the wrapped Oracle connection's close() method to close the connection.

If you turn on WebADE Log4J logging using a DEBUG level, you will see some logging coming from the ca.bc.gov.webade.dbpool package when a connection is checked out from the pool, when it is closed, and when it returned to the pool.

For instance:

Log (Level: DEBUG):

```
Getting connection  #26956691 from the pool: WEBADE
```

Purpose: You should see the above debug log when a connection is checked out of the pool.

Log (Level: DEBUG):

```
Connection #26956691 closed properly
```

Purpose: You should see the above debug log when the wrapper class is cleaned up by the garbage collector and the connection is closed properly by the java.sql.close() method.

Log (Level: WARN):

```
Connection #26956691 closed by garbage collector.  Connection check-out call
stack trace follows:
```

Purpose: You should see the above warn-level log when the wrapper class is cleaned up by the garbage collector and the connection is not closed properly (the close() method was not called on the WebADE wrapper connection object before garbage collection).

NOTE: This error will be followed by a Java stack trace of the call stack as it was when the connection was retrieved by the developer from WebADE. This should give you the class name and line number of the class that retrieved the

connection object from WebADE, helping to track down where the connection was not closed properly.

Log (Level: DEBUG):

```
Returning connection #26956691 to the pool: WEBADE
```

Purpose: You should see the above debug log when a connection is returned to the pool.

NOTE: You will see similar logging as those listed above for the wrapper classes for Statement and ResultSet objects.

9.4   WEBADE ERROR LOGS

In this section we will list the most frequently encountered errors when deploying and running a WebADE application, and the logs to look for in each situation. The log level of the excerpt will be stated, as well, before each logging excerpt, to help you set your log levels accordingly.

9.4.1   WEBADE CONNECTION JAR MISSING

Log (Level: FATAL):

```
java.lang.NoClassDefFoundError: ca/bc/gov/webade/datastore/WebADEConnection
```

Reason: This issue will appear when your application is missing the WebADE Connection jar from your classpath. See above for information on creating the WebADE Connection jar.

Resolution: This connection jar is normally added to your app server's shared library directory, and not directly included in your application's ear file.

In OC4J, this jar can be added to the app server's j2ee/home/applib directory, or added as an external <library> tag in the ear's META-INF/orion-application.xml file.

In JRun 4, this jar can be added to the app server's SERVER-INF/lib directory (where the mof-excluded libraries are kept).

In Tomcat, this jar can be added to the app server's shared/lib directory.

In Eclipse, add the connection jar to your projects Java Build Path as a library.

9.4.2   WEBADE CONNECTION BAD USERNAME/PASSWORD

Log (Level: FATAL):

```
java.sql.SQLException: ORA-01017: invalid username/password; logon denied
```

Reason: This issue will appear when your application has an incorrect username/password in the WebADE Connection jar from your classpath. See above for information on creating the WebADE Connection jar.

Resolution: Correct the username/password attributes in the WebADEConnection.java source file and recompile the WebADE Connection jar.

9.4.3 WEBADE CONNECTION BAD DATABASE URL

Log (Level: FATAL):

```
java.sql.SQLException: Invalid Oracle URL specified
```

or

```
Io exception: Connection
refused(DESCRIPTION=(TMP=)(VSNNUM=153093120)(ERR=12505)(ERROR_STACK=(ERROR=(C
ODE=12505)(EMFI=4))))
```

Reason: This issue will appear when your application has an incorrect JDBC database URL in the WebADE Connection jar from your classpath. See above for information on creating the WebADE Connection jar.

Resolution: Correct the database URL attributes in the WebADEConnection.java source file and recompile the WebADE Connection jar.

9.4.4 MOF CUSTOM ORGANIZATION API LIBRARY MISSING

Log (Level: FATAL):

```
Error occurred while loading datastore class.
java.lang.ClassNotFoundException:
ca.bc.gov.webade.mof.MOFOrganizationDatastore
```

Reason: The MOF custom API library is not in the classpath.

Resolution: Add the webade-mof-custom jar file to your application classpath.

For EAR files, add the jar to your application's WEB-INF/lib directory.

For Eclipse, add the jar to your projects Java Build Path as a library

9.4.5 MOF CUSTOM ORGANIZATION API NOT SET CORRECTLY

Log (Level: *Application-specific, when calling the getMinistryInfo() method of the Organization object*):

```
java.lang.NullPointerException
```

or

```
java.lang.ClassCastException
```

Reason: The MOF custom API is not loaded and so is not populating the MOF-specific organization info in the Organization object. The WebADE Connection jar does not have the WEBADE_DATASTORE_IMPLEMENTATION attribute set to ca.bc.gov.webade.mof.MOFOrganizationDatastore

Resolution: Set the WEBADE_DATASTORE_IMPLEMENTATION attribute in the WebADEConnection.java source file to ca.bc.gov.webade.mof.MOFOrganizationDatastore and recompile the WebADE Connection jar.

9.4.6 CAP WEB SERVICES USER PROVIDER BAD USERNAME/PASSWORD

Log (Level: ERROR):

```
ca.bc.gov.webade.WebADEException: JAX-RPC RemoteException caught:
(401)Unauthorized
```

Reason: The Connect-username/connect-password preference values are not correct.

Resolution: Correct the preference_value values for the rows with a preference_set_name of "bceid-web-services-provider" and preference_name values of "bceid-web-services-connect-username" and "bceid-web-services-connect-password" in the WebADE PREFERENCE table and restart the WebADE application.

9.4.7 CAP WEB SERVICES USER PROVIDER BAD URL

Log (Level: ERROR):

```
ca.bc.gov.webade.WebADEException: JAX-RPC RemoteException caught: (404)Not
Found
```

Reason: The URL to the CAP Web Services is not correctly set.

Resolution: Correct the preference_value value for the row with a preference_set_name of "bceid-web-services-provider" and preference_name value of "bceid-web-services-provider-url" in the WebADE PREFERENCE table and restart the WebADE application.

9.4.8 WEBADE DEVELOPER MODULE MISSING USER INFO XML

Log (Level: ERROR):

```
ca.bc.gov.webade.developer.WebADEDeveloperException:
C:\userInfoFileLocation\mofr\mofrora-dev-webade-xml-user-info.xml.bad (The
system cannot find the file specified)
Caused by: java.io.FileNotFoundException:
C:\userInfoFileLocation\mofr\mofrora-dev-webade-xml-user-info.xml.bad (The
system cannot find the file specified)
```

Reason: The user-info XML file is not found.

Resolution: Correct or add the context-param "user.info.file.location" in the web.xml so the param-value points to the location of the user-info XML file and redeploy the application.

9.4.9   WEBADE USER PROVIDER FOR LOGGED-IN USER NOT LOADED

Log (Level: ERROR):

```
Could not load user info for current user 'IDIR\USERNAME'.  Verify that a
user provider is configured that can load this user's information for the
WebADE.
```

Reason: WebADE cannot load the user's information from any of the application's configured user providers.

Resolution: Verify that the user providers are correctly configured in the WebADE database for your application.

If using the WebADE developer module in your application deployment, make sure that it is configured properly as the WebADE user provider, either in the database or by the external XML preference file. It is possible that the developer module is set up to log people in, but not to respond to user info queries from the WebADE API.

9.4.10   NO WEBADE USER PROVIDERS LOADED

Log (Level: ERROR):

```
No user providers were loaded for this application.  Check the logs for
errors.
```

Reason: WebADE did not load any of the application's configured user providers.

Resolution: Check the logs for other errors. Any user provider that failed to load will add one or more errors to the logs. Also, if a user-provider is configured, but has the "enabled" preference set to "false", a WARN-level error should appear in the logs, indicating that WebADE is ignoring this provider for the application.

If there are no other errors in the logs, the other possibility is that there are no user providers configured for the application at all. Check the PREFERENCE table to make sure either the CAP Web Services user provider or WebADE Developer Module user provider are correctly set and enabled to run for the application.

9.4.11 WEBADE USER PROVIDER NOT LOADED

Log (Level: ERROR):

```
No user provider configured in this WebADE application that can load given
user 'IDIR\USERNAME'
```

Reason: WebADE cannot load the user's information from any of the application's configured user providers.

Resolution: Verify that a user provider that can return the user info for the target user is correctly configured in the WebADE database for your application .

If using the WebADE developer module in your application deployment, make sure that it is configured properly as the WebADE user provider, either in the database or by the external XML preference file. It is possible that the developer module is set up to log people in, but not to respond to user info queries from the WebADE API.

9.4.12 MISSING WEBADESERVLETCONTEXTLISTENER

Log (Level: FATAL):

```
WebADE is not initialized.  The Application singleton is not in the servlet
context.  The WebADE Servlet Context Listener configuration may be missing
from the web.xml file.
```

Reason: You application's WEB-INF/web.xml does not have the WebADEServletContextListener correctly defined.

Resolution: Update the web.xml file to add the WebADEServletContextListener in the appropriate location in the file and redeploy the application. See the WebADE User's Guide documentation for more information.

9.4.13 MISSING/INCORRECTLY CONFIGURED WEBADEFILTER

Log (Level: WARN):

```
Returning session null user permissions.  The WebADE Filter may not be pre-
processing the requests.
```

and/or

```
Returning request null user permissions.  The WebADE Filter may not be pre-
processing the requests.
```

Reason: You application's WEB-INF/web.xml may not have the WebADEFilter filter and/or filter mapping defined. **NOTE:** You may also experience NullPointerException events, as WebADE is returning a null user permissions object.

Resolution: Modify the web.xml file to add/update the WebADEFilter filter and filter-mapping entries in the appropriate locations in the file and redeploy the application. See the WebADE User's Guide documentation for more information.

9.4.14 ONESTOP HEADERS MISSING

Log (Level: *Web browser returns with a 400 error*):

```
Missing required OneStop attributes SMGOV_USERTYPE, SMGOV_USERGUID from
request headers.  User's session could not be initialized.
```

Reason: Site Minder is not returning the complete set of headers required by WebADE to determine the current user.

Resolution: Contact CITS for Site Minder support. To help troubleshooting with CITS, the 400 error will list exactly which request headers are missing (See example above).

9.4.15 MISSING/INCORRECT SYSTEM USER CREDENTIALS

Log (Level:):

```
Error occurred while initializing UserInfoProvider from class
'ca.bceid.webservices.BCeIDWebServicesProvider'
ca.bc.gov.webade.WebADEException: BCeID Web Services Provider 'bceid-web-
services-provider' could not load system user
'GOV:9A1DEC488AE04AC8A39A010645675AAF'.
```

Reason: The system-user preferences in the PREFERENCE table are not correctly set.

Resolution: Verify that the system-user preferences are in the PREFERENCE table and that the settings are for a user that is accessible from your application's user provider (User exists in the CAP Web Service if using the CAP Web Services Provider, or in the user-info XML if using the developer module).

9.4.16 MISSING/INCORRECT ANONYMOUS USER (DEPRECATED)

Log (Level: ERROR):

```
Could not load WebADE anonymous user.  Verify the anonymous-user preference
value in the database.
```

Reason: The anonymous-user preference in the PREFERENCE table is not correctly set.

Resolution: Verify that the anonymous-user preference is in the PREFERENCE table and that the preference-value is set to a user that is accessible from your application's user provider (User exists in the CAP Web Service if using the CAP Web Services Provider, or in the user-info XML if using the developer module).

9.4.17   WEBADE WRAPPED CONNECTION CLOSED UNEXPECTEDLY

Log (Level: ERROR):

```
java.sql.SQLException: Logical handle no longer valid
```

Reason: Your application is retrieving the native Oracle Connection object from the WebADE wrapper class by calling the getWrappedConnection() method and then losing reference to the wrapper connection class. When the Java garbage collection cleans up the wrapper connection class, the WebADE wrapper class will close the connection.

Resolution: Make sure your application maintains an object reference to the wrapper class for as long as you are using the Oracle connection class (This applies to the WebADE Wrapper Statement and ResultSet classes as well). NOTE: Remember to call the close() method on the wrapper class, and not the Oracle native connection class, to allow WebADE to clean up itself properly and not add warnings to the logs.

9.5   OTHER ISSUES

In this section we will list errors when deploying and running a WebADE application that are not obvious, as the logs either are not useful or they do not indicate that there is a problem.

9.5.1   TEST VS PRODUCTION SITE MINDER ISSUES

Issue: The Site Minder configuration is authenticating by connecting to one of either CITS' Production or Test environment, while your WebADE application's CAP Web Services Provider is connecting to the other.

Symptoms: IDIR users are able to log in to your application without issue, but BCeID and MYID users cannot. This is because IDIR users are available in all Site minder/CAP web services environments, but each environment has separate directories for BCeID and MYID users.

Resolution: Either change the URL preference in the PREFERENCE database table, or contact CITS to change the Site Minder configuration of your application.