Oracle® Adaptive Access Manager

Proxy Integration Guide

10*g* (10.1.4.3.0)

December 2007

Oracle Adaptive Access Manager Proxy Integration Guide, 10g (10.1.4.3.0)

Contents

Preface

The Oracle® Adaptive Access Manager Proxy Integration Guide provides programming information and instructions on the installation of the Adaptive Access Manager proxy, one of the components in the Adaptive Access Manager UIO deployment. The Oracle Adaptive Access Manager's Universal Installation Option (UIO) offers multi-factor authentication to Web applications without requiring any change to the application code. The Oracle® Adaptive Access Manager Proxy and The Oracle® Adaptive Access Manager Proxy Web Publishing Configuration are guides specific to the UIO deployment.

Documentation

The Oracle Adaptive Access Manager 10g documentation includes the following:

- The Oracle® Adaptive Access Manager API Integration Guide, which provides information on natively integrating the client portion of the Adaptive Risk Manager Online solutions. In an API integration, the client application invokes the Adaptive Risk Manager Online APIs directly and manages the authentication and challenge flows.
- The Oracle® Adaptive Access Manager Database Installation Guide (Oracle), which provides information about installing the Adaptive Access Manager schema into an Oracle database. Access to the Adaptive Access Manager schema is a requirement of the Adaptive Access Manager Application Server, which hosts the Adaptive Strong Authenticator and the Adaptive Risk Manager. Note that the Adaptive Manager Access Manager schema needs to be installed into the Oracle database before proceeding to the installation of the proxy.
- The Oracle® Adaptive Access Manager Database Installation Guide for SQL Server 2005, which provides information about installing the Adaptive Access Manager schema into SQL Server 2005. Access to the Adaptive Access Manager schema is a requirement of the Adaptive Access Manager Application Server, which hosts the Adaptive Strong Authenticator and the Adaptive Risk Manager. Note that the Adaptive Manager Access Manager schema needs to be installed into SQL Server 2005 before proceeding to the installation of the proxy.
- The Oracle® Adaptive Access Manager Proxy Integration Guide, which provides programming information and instructions on the installation of the Adaptive Access Manager proxy, one of the components in the Adaptive Access Manager UIO deployment. The Oracle Adaptive Access Manager's Universal Installation Option (UIO) offers multi-factor authentication to Web applications without requiring any change to the application code. The Oracle® Adaptive Access Manager Proxy and The Oracle® Adaptive Access Manager Proxy Web Publishing Configuration are guides specific to the UIO deployment.
- The Oracle® Adaptive Access Manager Proxy Web Publishing Configuration, which provides information on creating web publishing rules and listeners so that Web applications and the WebUIO can be accessible from the Internet. The Oracle Adaptive Access Manager's Universal Installation Option (UIO) offers multi-factor authentication to Web applications without requiring any change to the application code. The Oracle® Adaptive Access Manager Proxy and The Oracle® Adaptive Access Manager Proxy Web Publishing Configuration are guides specific to the UIO deployment.

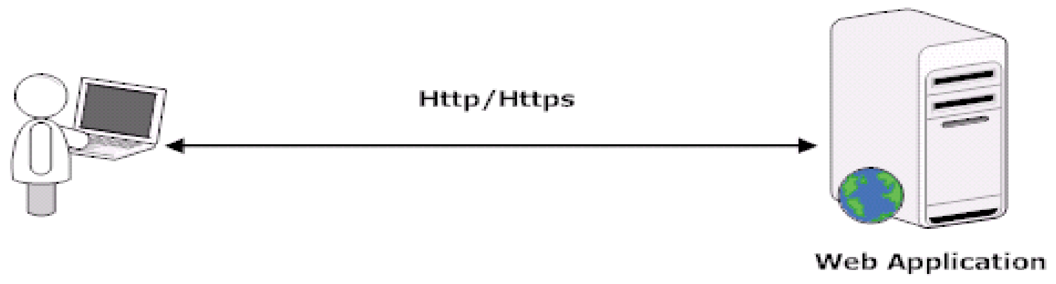
- The Oracle® Adaptive Risk Manager Online Installation Guide, which provides information on the installation of the administration user interface of Oracle Adaptive Access Manager. Adaptive Risk Manager Online is the administration user interface of Oracle Adaptive Access Manager, a set of web-based administration tools that provides sophisticated fraud monitoring, analysis, and tracking by user location, device, time of day, type of transaction, as well as a host of other factors, and evaluates these factors against a set of customizable rules.
- The Oracle® Adaptive Access Manager LDAP Configuration Guide, which provides information on how to configure the Oracle Adaptive Access Manager Application Server to allow a user to be authenticated via a user identifier and password. The intended audience of this manual are users of WebLogic and Tomcat who want to use LDAP to set up users instead of the functionality in WebLogic and Tomcat.
- The Oracle® Adaptive Access Manager Import/Export Manual, which provides information importing groups, rule templates, and models from the Adaptive Access Manager schema.
- The Oracle® Adaptive Risk Manager Online Customer Care API Guide, which provides information about the Adaptive Risk Manager Online Customer Care API and provides the XML definition for each of the APIs.
- The Oracle® Adaptive Access Manager Database Tables Archiving and Purging Procedure, which provides information on the purge and archive scripts in the Oracle Adaptive Access Manager Database Tables of Microsoft SQL Server 2005. The procedure to trigger the scripts and information on verification and validation of script results are also provided.
- The Oracle® Adaptive Access Manager SQL Server Maintenance Guide, which provides instructions to set up the Oracle Adaptive Access Manager Maintenance Plan to purge and archive scripts in the Oracle Adaptive Access Manager database tables of Microsoft SQL Server 2005. The manual also discusses in detail how to trigger the scripts and provides information on the verification and validation of script results.
- The Oracle® Adaptive Risk Manager™ Administrator's Guide, which provides step-by-step instructions for creating and managing groups, creating models that contain rules, and customizing and managing rules.
- The Oracle® Adaptive Risk Manager™ Dashboard and Reporting Guide, which provides detailed instructions on how to use the dashboard and reporting functionality within the Oracle® Adaptive Risk Manager Online. The Oracle® Adaptive Risk Manager Online includes a dashboard that provides a high-level overview of users and devices that have generated alerts and the alerts themselves, and it contains a comprehensive collection of reports on users, locations, devices, and security alerts.
- The Oracle® Adaptive Risk Manager™ Customer Care Administration Guide, which provides information on creating new customer cases and administering them.

Introduction

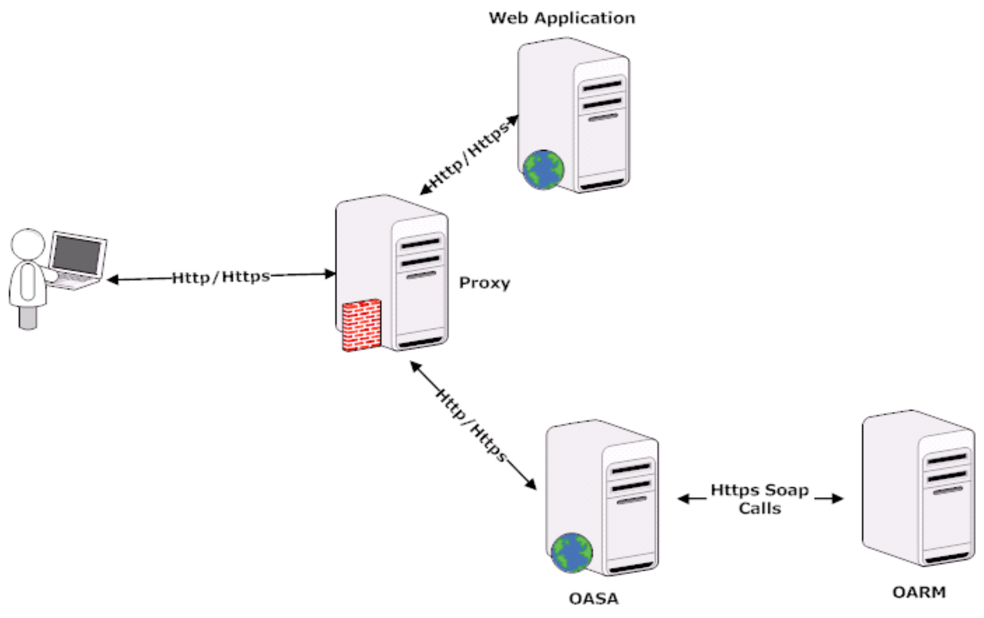
The Oracle Adaptive Access Manager UIO offers additional multi-factor authentication to Web applications without requiring any change to application code. The purpose of this document is to explain the Oracle Adaptive Access Manager proxy, one of the components in the Oracle Adaptive Access Manager UIO deployment. This document is intended for integrators who configure the Oracle Adaptive Access Manager proxy to add multi-factor authentication to Web applications. A very good understanding of HTTP request/response paradigm is required to understand the material presented in this document.

Architecture

The following diagrams show a typical Oracle Adaptive Access UIO deployment. The first diagram shows a Web application before the Oracle Adaptive Access UIO is deployed to provide multi-factor authentication. The second diagram shows various components after the Oracle Adaptive Access UIO deployment.



Before the Oracle Adaptive Access UIO



After the Oracle Adaptive Access UIO

The Adaptive Strong Authenticator proxy intercepts the HTTP traffic between the client (browser) and the server (web application) and performs appropriate actions, such as redirecting to the Adaptive Strong Authenticator, to provide multi-factor authentication and authorization. The Adaptive Strong Authenticator in turn communicates with Adaptive Risk Manager to assess the risk and takes the appropriate actions, such as permitting the login, challenging the user, blocking the user, and other actions.

The Oracle Adaptive Access Manager Proxy Installation

The Oracle Adaptive Access Manager Proxy uses the API provided by Microsoft ISA Server to monitor the HTTP traffic and perform various actions. Microsoft ISA Server 2006 or 2004 Standard edition should be installed before installing the Oracle Adaptive Access Proxy. Please refer to Microsoft ISA Server setup documentation for the details on installing and configuring ISA server.

The Oracle Adaptive Access Manager proxy is installed as a Web Filter in Microsoft ISA Server. Please follow the steps below to install the Oracle Adaptive Access Proxy:

1. Copy `BharosaProxy.dll` to Microsoft ISA Server installation directory, which is by default `%ProgramFiles%\Microsoft ISA Server`
2. Open command prompt and navigate to Microsoft ISA Server installation directory.
3. Register BharosaProxy.dll with the following command:
   regsvr32 .\BharosaProxy.dll

Settings

Various aspects of the Oracle Adaptive Access Proxy can be controlled using the registry values. All Oracle Adaptive Access Proxy settings are stored under HKLM\SOFTWARE\Bharosa\Proxy key. Changes to most of the registry values are picked up by proxy immediately without requiring a proxy restart.

Configuration files

During startup (and during config reload), proxy loads configuration from the files listed under `HKLM\SOFTWARE\Bharosa\Proxy\ConfigFiles` key.

- Type of each value under this key should be REG_DWORD.
- Name of each value under this key should be the filename containing proxy configuration.
- The filename can either be fully qualified or relative to the location of BharosaProxy.dll
- Proxy will load a configuration file only if the data has non-zero value. This can be used to dynamically load and unload proxy configuration files.
- The files will be loaded in the lexicographic order of the filenames in the registry.
- Changes under `ConfigFiles` key will not be effective until either proxy restart or `HKLM\SOFTWARE\Bharosa\Proxy\ReloadConfig` is set to 1.

Configuration reload

Proxy configuration can dynamically be changed while proxy is running; new configuration files can be added and currently loaded files can be updated or removed. But these changes will not be applied until `ReloadConfig` registry value is set to a non-zero value. Upon setting `ReloadConfig` to non-zero value, proxy will load configuration files. After loading the files, proxy will reset `ReloadConfig` value to 0.

Please note that the new configuration will be used only for new client (browser) connections. Clients already connected will continue to use the previous configuration.

Session Id Cookie

The Oracle Adaptive Access Manager proxy uses a cookie to relate between multiple requests from a client. The name of this cookie can be configured in registry value `SessionIdCookieName` (of type REG_SZ). If this value is not present or empty, the Oracle Adaptive Access proxy will use the cookie name, `BharosaProxy_SessionId`.

Session Inactive Interval

Sessions in the Oracle Adaptive Access proxy will be removed after certain period of inactivity. This period, in number of seconds, is specified in `MaxSessionInactiveInterval` registry value. If this value is not specified, the Oracle Adaptive Access Manager proxy will remove a session after 1200 seconds (20 minutes) of inactivity. This value should be set to at least few seconds higher than the Web application session timeout value.

Settings for Troubleshooting

Trace messages from the Oracle Adaptive Access proxy can be used for trouble shooting any issues with the proxy configuration and operation. Trace settings like trace level and destinations can be controlled using following registry values under HKLM\SOFTWARE\Bharosa\Proxy:

Name	**Type**	**Description**
TraceFilename	REG_SZ	Full path to the file in which the trace messages should be written to
TraceFileMaxLength	REG_DWORD	Maximum length of the trace file in bytes. Once the trace file reaches this size, proxy will rename the file by adding the timestamp to the filename and create a new trace file to write subsequent trace messages.
TraceToFile	REG_DWORD	Trace messages will be written to file only if this value is non-zero.
TraceToDebugTerminal	REG_DWORD	Trace messages will be written to debug terminal only if this value is non-zero. Tools like DbgView can be used to view these trace messages in real time.
TraceLevel	REG_DWORD	Each trace message is associated with a level – like error, warning, info, debug, etc. The Oracle Adaptive Access Manager proxy will output only the messages whose trace level is enabled in this registry value. Set this registry value to the addition of trace levels to be enabled. Various trace levels values are: FATAL 0x1, ERROR 0x2, WARN 0x4 INFO 0x8, DEBUG 0x10, HTML 0x80, FLOW 0x80000
IgnoreUrlMappings	REG_DWORD	If this value is non-zero, proxy will ignore all the interceptors defined in proxy configuration. Essentially this will put the Oracle Adaptive Access proxy in “pass-through” mode.
CaptureTraffic	REG_DWORD	Proxy does not handle (save, inspect) the HTTP traffic for URLs that don’t have interceptors defined in the configuration. But during application discovery process, it will be necessary to get a dump of all the HTTP traffic thorough the proxy. On such occasion, this registry value should be set to non-zero.

Proxy Configuration

The Oracle Adaptive Access Manager proxy intercepts all HTTP traffic between the client browser and the Web application and performs actions specified in the configuration files. The configuration files are in XML format that comply with the Oracle Adaptive Access Proxy config XML schema 2. The following sections describe various elements of the Oracle Adaptive Access Manager proxy configuration file.

Interceptors

Interceptors are the most important elements in the Oracle Adaptive Access Manager proxy configuration. You will see that authoring the Oracle Adaptive Access Manager proxy configuration file is all about defining interceptors.

There are two types of interceptors: request interceptors and response interceptors. As the names suggest, request interceptors are used when the Oracle Adaptive Access Manager proxy receives HTTP requests from the client browser and response interceptors are used when the Oracle Adaptive Access Manager proxy receives HTTP response from the server i.e. Web application or Web UIO.

There are four components to an interceptor and all of them are optional.

1. List of URLs – the interceptor will be evaluated if the interceptor URL list contains the current request URL or if the URL list is empty.
2. List of conditions – conditions can inspect the request/response contents, like check for presence of an HTTP header/parameter/cookie, etc or test whether a header/parameter/cookie has a specific value or not. Filters and action defined in the interceptor will be executed only if all the conditions specified are met or if no condition is specified.
3. List of filters – filters perform an action that might modify the request/response contents or modify some state information in the Oracle Adaptive Access Manager proxy. For example, a filter can add/remove HTTP headers, save HTTP header/parameter/cookie value in a proxy variable, etc.
4. Action – after executing the filters the interceptor will perform the action, if one is specified. Actions can be one of:
    a. redirect the client to a different URL
    b. send a saved response to the client
    c. perform a HTTP get on server
    d. perform a HTTP post on server
    e. send a saved request to the server

Conditions

Conditions are used in the Oracle Adaptive Access Manager proxy to inspect HTTP request/response or the state information saved in the proxy (variables). Each condition evaluates to either true or false. Conditions are evaluated in the order they are listed in the configuration file until a condition evaluates to false or all conditions are evaluated. Here is the list of conditions that can be defined in an interceptor:

Condition name	Attributes	Description
HeaderPresent	id, enabled, name	Checks the presence of the specified header in request/response
ParamPresent	id, enabled, name	Checks the presence of the specified parameter in request
QueryParamPresent	id, enabled, name	Checks the presence of the specified query parameter in the URL
VariablePresent	id, enabled, name	Checks whether the specified proxy variable has been set
RequestCookiePresent	id, enabled, name	Checks the presence of the specified cookie in request
ResponseCookiePresent	id, enabled, name	Checks the presence of the specified cookie in response
HeaderValue	id, enabled, name, value, mode, ignore-case	Checks whether the specified request/response header value matches the given value.
ParamValue	id, enabled, name, value, mode, ignore-case	Checks whether the specified request parameter value matches the given value.
QueryParamValue	id, enabled, name, value, mode, ignore-case	Checks whether the specified URL query parameter value matches the given value.
VariableValue	id, enabled, name, value, mode, ignore-case	Checks whether the specified proxy variable value matches the given value.
RequestCookieValue	id, enabled, name, value, mode, ignore-case	Checks whether the specified request cookie value matches the given value.
ResponseCookieValue	id, enabled, name, value, mode, ignore-case	Checks whether the specified response cookie value matches the given value.
HttpStatus	id, enabled, status	Checks whether the status code of the response matches the given value
HtmlElementPresent	id, enabled, name, attrib-name, attrib-value, attrib-name1, attrib-value1, … attrib-name9, attrib-value9,	Checks presence of a html element to match the specified conditions: <name attrib-name="attrib-value" attrib-name1="attrib-value1" …/>
PageContainsText	id, enabled, text	Checks whether the response contains the given text

NotVariableValue	id, enabled, name, value, mode, ignore-case	Checks whether the specified proxy variable value does not matche the given value.
And	id, enabled	Evaluates to true only if all the child conditions evaluate to true
Or	id, enabled	Evaluates to true if one of the child conditions evaluates to true
Not	id, enabled	Reverses the result of the child condition(s)

Attribute "id" is optional and is used only in trace messages. If no value is specified, the condition name (like HeaderPresent) will be used.

Attribute "enabled" is optional and the default value is "true". This attribute can be used to enable/disable a condition. The value of this attribute can be set to the name of a global variable – in such case the condition will be enabled or disabled according to the value of the global variable.

Attribute "value" can be set to the name of a proxy variable – in such case proxy will evaluate the variable at runtime and use that value in the condition.

Attribute "mode" can be set to one of the following: `begins-with, ends-with, contains`.

Attribute "ignore-case" can be set to one of the following: `true, false`.

Filters

Filters are used in the Oracle Adaptive Access Manager proxy to modify HTTP request/response contents or modify the state information saved in the proxy (variables). Filters are executed in the order they are listed in the configuration file. Here is the list of filters that can be defined in an interceptor:

Filter name	**Attributes**	**Description**
AddHeader	id, enabled, name, value	Adds the specified header with a given value to request/response
SaveHeader	id, enabled, name, variable	Saves the specified request/response header value in the given proxy variable.
RemoveHeader	id, enabled, name	Removes the specified header from request/response.
AddParam	id, enabled, name, value	Adds a request parameter with a specified name and value
SaveParam	id, enabled, name, variable	Saves the specified request parameter value in to the given proxy variable

AddRequestCookie	id, enabled, name, value	Adds the specified cookie with a given value to request
SaveRequestCookie	id, enabled, name	Saves the specified request cookie value in the given proxy variable
AddResponseCookie	id, enabled, name	Adds the specified cookie with a given value to response
SaveResponseCookie	id, enabled, name	Saves the specified response cookie value in the given proxy variable
SaveHiddenFields	id, enabled, form, variable, save-submit-fields	Saves all the hidden, submit fields value, in the given form – if form name is specified, to the given proxy variable. To not save submit fields, set save-submit-fields attribute to false.
AddHiddenFieldsParams	id, enabled, variable	Adds request parameters for each hidden field saved in the variable
SetVariable	id, enabled, name, value	Sets proxy variable with the given name to the specified value
UnsetVariable	id, enabled, name	Removes the proxy variable with the given name
ClearSession	id, enabled, name	Removes all session variables in the current session.
SaveQueryParam	id, enabled, name, variable	Saves the specified query parameter in the given proxy variable
SaveRequest	id, enabled, variable	Saves the entire request content in the given proxy variable
SaveResponse	id, enabled, variable	Saves the entire response content in the given proxy variable
ReplaceText	id, enabled, find, replace	Updates the response by replacing the text specified in “find” attribute with the value given in “replace” attribute
ProcessString	id, enabled, source, find, action, count, search-str, start-tag, end-tag, ignore-case, replace	This filter can be used to extract a sub-string from a string (like request, response contents) and save it to a proxy variable. This filter can also be used to dynamically format strings. Please see the examples below on how to use this filter.

Filter Examples - ProcessString

Find the `sub-string` between the given `start-tag` and `end-tag` in the `source` string, `extract` the sub-string found and save extracted sub-string in the given `variable`.

```
<ProcessString source="%RESPONSE_CONTENT"
    find="sub-string"
    start-tag="var traceID = '" end-tag="';"
    action="extract"
    variable="$TRACE_ID"/>
```

Find the given `search-string` in the `source` string, replace it with the `replace` string and save the updated string in the given `variable`.

```
<ProcessString
      source="/bfb/accounts/accounts.asp?TraceID=$TRACE_ID"
      find="string" search-str="$TRACE_ID"
      action="replace"
      replace="$TRACE_ID"
      variable="%POST_URL"/>
```

Find the `sub-string` between the given `start-tag` and `end-tag` in the `source` string, replace it (including the start and end tags) with the `eval`uated value of the sub-string found and save the updated string in the given `variable`.

```
<ProcessString
         source="/cgi-bin/mcw055.cgi?TRANEXIT[$UrlSuffix]"
         find="sub-string" start-tag="[" end-tag="]"
         action="eval"
         variable="%LogoffUrl"/>
```

Actions

An interceptor can optionally perform one of the following actions after executing all the filters. No further interceptors will be attempted after executing an action.

redirect-client

Often the proxy would need to redirect the client to load another URL – redirect-client is the action to use in such cases. Proxy will send a 302 HTTP response to request the client to load the specified URL.

If display-url attribute is specified in the interceptor, proxy will send a HTTP 302 response to the browser to load the URL specified in display-url attribute. When proxy receives this request, it will do a HTTP-GET on the server to get the URL specified in “url” attribute.

send-to-client

Often a response from the server would have to be saved in the proxy and sent to the client later after performing few other HTTP requests – send-to-client is the action to use in such cases. Proxy will send the client the contents of specified variable.

If display-url attribute is specified in the interceptor, proxy will send a HTTP 302 response to the browser to load the URL specified in display-url attribute. When proxy receives this request, it will send the response specified in the interceptor.

get-server

Sometimes proxy would need to get a URL from the server – get-server is the action to use in such cases. Proxy will send a HTTP-GET request for the specified URL to the server.

If display-url attribute is specified in the interceptor or if this action is specified in a response interceptor, proxy will send a HTTP 302 response to the browser. When proxy receives this request it will do a HTTP-GET on the server to get the URL specified in “url” attribute.

post-server

Sometimes proxy would need to post to a URL in the server – post-server is the action to use in such cases. Proxy will send a HTTP-POST request for the specified URL to the server.

If display-url attribute is specified in the interceptor or if this action is specified in a response interceptor, proxy will send a HTTP 302 response to the browser. When proxy receives this request it will do a HTTP-POST to the server to the URL specified in “url” attribute.

send-to-server

In certain situations the request from client needs to be saved in the proxy and sent to the server later after performing few other HTTP requests – send-to-server is the action to use in such cases. Proxy will send the contents of the specified variable to the server.

If display-url attribute is specified in the interceptor or if this action is specified in a response interceptor, proxy will send a HTTP 302 response to the browser. When proxy receives this request it will send the request specified in the interceptor to the server.

Variables

Proxy variables can store string data in proxy memory. Variables can be used in conditions, filters and actions. For example, SaveHeader filter can be used to save the value a specific header in the given proxy variable. This variable value could later be used, for example, to add a parameter to the request. Variables can also be used in conditions to determine whether to execute an interceptor or not.

Proxy variables are of 3 types, depending upon the lifespan of the variable. Type of variable is determined by the first letter of variable name – which can be one of: %, $, @.

All types of variables can be set using filters like SetVariable SaveHeader, SaveParam, SaveRespons, etc.

All types of variables can be unset/deleted by UnsetVariable filter. ClearSession filter can be used to remove all session variables.

Request variables

Request variables – these variable names start with %. These variables are associated with the current request and are deleted at the completion of the current request. Request variables are used where the value is not needed across requests.

Session variables

Session variables – these variable names start with $. These variables are associated with the current proxy session and are deleted when the proxy session is cleaned up. Session variables are used where the value should be preserved across requests from a client.

Global variables

Global variables – these variable names start with @. These variables associated with the current proxy configuration and are deleted when the proxy configuration is unloaded. Global variables are used where the value needs to be preserved across requests and across clients.

Global variables can be set at proxy configuration load time using SetGlobal in the configuration file. Global variables can also be set by adding registry values under key `HKLM\Software\Bharosa\Proxy\Globals`. Name of each entry under this key should be the variable name, starting with @. And the data of the entry should be the value of the variable. The registry-type of the value can be REG_DWORD, REG_SZ or REG_EXPAND_SZ.

Pre-defined variables

The Oracle Adaptive Access Manager proxy supports following pre-defined request variables:

Variable name	**Description**
%RESPONSE_CONTENT	This variable contains the contents of the entire response from the web server for the current request.
%REQUEST_CONTENT	This variable contains the contents of the entire request from the client.
%QUERY_STRING	This variable contains the query string, starting with ?, for the current request URL.
%REQUEST_METHOD	HTTP method verb for the request: GET, POST, etc
%REMOTE_HOST	Hostname of the client or agent of the client
%REMOTE_ADDR	IP address of the client or agent of the client
%HTTP_HOST	The content of HTTP Host header
%URL	URL for the current request

Application

A single Oracle Adaptive Access Manager proxy installation can be used to provide multi-factor authentication for multiple web application that run in one or more web servers. In the Oracle Adaptive Access Manager proxy configuration, an Application is a grouping of interceptors defined for a single web application.

Request and response interceptors can be defined outside of an application in the Oracle Adaptive Access Manager proxy configuration file. These interceptors are called “global” interceptors and will be evaluated and executed prior to interceptors defined in applications.

Interception process

When a request arrives, the Oracle Adaptive Access Manager proxy evaluates request interceptors defined for the URL in the order they are defined in the configuration file. Similarly when on receiving response from the web server, the Oracle Adaptive Access Manager proxy evaluates response interceptors defined for the URL in the order defined in the configuration file.

If the conditions in an interceptor evaluate to true, the Oracle Adaptive Access Manager proxy will execute that interceptor i.e. execute the filters and action. After executing an interceptor, the Oracle Adaptive Access Manager proxy will continue with the next interceptor only if the following conditions are met:

- no action is specified for the current interceptor
- post-exec-action attribute for the current interceptor is `continue`

Even if one of the above conditions is not met the Oracle Adaptive Access Manager proxy will stop evaluating subsequent interceptors.

It is highly recommended that “post-exec-action” attribute is specified for interceptors that don’t define an action. For global interceptors (for example – the interceptors defined outside of any application), the default value of “post-exec-action” attribute is `continue`. For non-global interceptors, the default value is `stop-intercept`.

As mentioned earlier the Oracle Adaptive Access Manager proxy configuration can contain multiple applications. While finding the list of interceptors to evaluate for a URL, only the following interceptors are considered:

- global interceptors that are defined outside of any application
- interceptors defined in the application associated with the current session

Each session will be associated with at most one application. If no application is associated with the current session (yet) when proxy finds an interceptor in an application for the URL, it will associate the application with the current session.

If the current session already has an application associated, and if no interceptor is found in that application for the URL, proxy will then look for intercepts in other applications. If an interceptor is found in another application for the URL, a new session will be created and the request will be associated with the new session.

Web UIO interface

The Oracle Adaptive Access Manager proxy redirects the user to Web UIO pages at appropriate times, for example to collect the password using the Adaptive Strong Authenticator, to run risk rules, etc. HTTP headers are used to exchange data between the Oracle Adaptive Access Manager proxy and Web UIO. The following table lists the Web UIO pages referenced in proxy configuration along with the details of HTTP headers used to pass data and. It also lists the expected action to be taken by proxy on the given conditions.

URL	**Condition**	**Action**
Any request to Web UIO page	On receiving request	Set header "BharosaAppId". Web UIO will use this header value to select appropriate customizations (UI, rules, etc).
loginPage.do	On receiving request to application login page	Redirect to this URL to use the Oracle Adaptive Access Manager login page instead of the application's login page.
password.do	Response contains headers userid, password (could be more depending upon the application)	Save the credentials from the response headers and post to the application
login.do	Phase-1 only: After validating the credentials entered by the user.	Redirect to this URL to update the status in the Adaptive Risk Manager and run appropriate risk rules.
login.do	Phase-1 only: On receiving the request.	Set "userid" header to the userid entered by the user. Set "Login-Status" header to one of the following: success, wrong_password, invalid_user, user_disabled, system_error. Set "WebUIOPhase" header to "one".
updateLoginStatus.do	Phase-2 only: After validating the credentials entered by the user.	Redirect to this URL to update the status in Adaptive Risk Manager and run appropriate risk rules.
updateLoginStatus.do	Phase-2 only: On receiving request	Set "Login-Status" header to one of the following: success, wrong_password, invalid_user, user_disabled, system_error.
updateLoginStatus.do challengeUser.do registerQuestions.do userPreferencesDone.do	Response header "Rules-Result" has value "allow"	The Oracle Adaptive Access Manager rules evaluated to permit the login. Proxy can permit access to the protected application URLs after this point.

updateLoginStatus.do challengeUser.do registerQuestions.do userPreferencesDone.do	Response header “Rules-Result” has value “block”	Either the application did not accept the login credentials or the Oracle Adaptive Access Manager rules evaluated to block the login. Proxy should logoff the session in the application, if login was successful. Then a login blocked message should be sent to the browser.
changePassword.do	Response contains headers “password”, “newpassword” and “confirmpassword”	Save the passwords from the response headers and post to the application
loginFail.do	To display error message in Web UIO page, like to display login blocked message	Redirect to this URL with appropriate “action” query parameter – like loginFail.do?action=block
logout.do	On completion of application session logout	Redirect to this URL to logout Web UIO session
logout.do	On receiving response	Redirect to application logout URL to logoff application session, if it is not done already
resetPassword.do	Response contains headers “newpassword” and “confirmpassword”	Save the passwords from the response headers and post to the application
getUserInput.do	Response contains headers “BH_UserInput”	Save the user input and take appropriate action (like post to application, etc)
changeUserId.do	On receiving request	Add “newUserId” header
changeUserId.do	On receiving response	Redirect to appropriate application page or send back saved application response
updateForgotPasswordStatus.do	Phase-2 only: After validating the forgot-password-credentials entered by the user.	Redirect to this URL to update the status in Adaptive Risk Manager and run appropriate risk rules.
updateForgotPasswordStatus.do	Phase-2 only: On receiving request	Set “BH_FP-Status” header to one of the following: success, wrong_password, invalid_user, user_disabled, system_error.
updateForgotPasswordStatus.do challengeForgotPasswordUser.do	Response header “BH_FP-Rules-Result” has value “allow”	The Oracle Adaptive Access Manager rules evaluated to permit the forgot-password flow. Proxy can permit continuation of to forgot-password flow, perhaps to reset the password or allow the user login, depending on the application.

updateForgotPasswordStatus.do challengeForgotPasswordUser.do	Response header “BH_FP-Rules-Result” has value “block”	Either the application did not accept the forgot-password credentials or the Oracle Adaptive Access Manager rules evaluated to block the forgot-password flow. A login blocked message should be sent to the browser.
Any request to Web UIO page	If proxy needs to get a property value from Web UIO. On receiving request	“BH_PropKeys” request header should be set to list of property names (separated by comma). Web UIO will return the values in multiple response headers, one for each property. The return response header names will be of format: “BH_Property-<name>”

Application Discovery

Application discovery is the process of studying an existing Web application to author proxy configuration to add multi-factor authentication using the Oracle Adaptive Access Manager UIO. Few logins attempts to the application would be made via the proxy to capture the HTTP traffic in each attempt. The captured HTTP traffic would be then be analyzed to author the proxy configuration. The Oracle Adaptive Access Manager proxy should be setup to dump all the HTTP traffic through it to a file. Then bunch of logins/login attempts to the application should be made via the proxy. The captured HTTP traffic should be then be analyzed to author the proxy configuration.

Application Information

For application discovery process it is preferable to work with the web application in customer's test environment, rather than the live application being used by users. If the test environment is not available for some reason, the live application can be used.

The following information is needed from the client for the application discovery process:

1. URL to login to the application.
2. Test user account credentials, including the data required in forgot password scenario. It will be best to get as many test accounts as possible, preferably at least 5 accounts, for uninterrupted discovery and testing. Please note that during discovery process some accounts could become disabled, perhaps due to multiple invalid login attempts.
3. Contact (phone, email) to enable/reset test accounts

Setting up proxy

The Oracle Adaptive Access Manager proxy settings (registry values under HKLM\SOFTWARE\Bharosa\Proxy key) should be set as given below for the proxy to capture the HTTP traffic to the specified file. This HTTP traffic captured will later be used for analysis to author proxy configuration.

Setting	**Value**
IgnoreUrlMappings	1
CaptureTraffic	1
TraceFilename	<filename>
TraceLevel	0x87
TraceToFile	1

It might be useful to capture the HTTP traffic for each scenario (like successful login attempt, wrong password, wrong username, disabled user, etc) in separate files. TraceFilename setting should be updated to the desired filename before start of the scenario.

After application discovery is done, proxy settings should be set as given below to restore the default the Oracle Adaptive Access Manager proxy behavior.

Setting	Value
IgnoreUrlMappings	0
CaptureTraffic	0
TraceFilename	<filename>
TraceLevel	0x7
TraceToFile	1

Setting up ISA

Microsoft ISA server should be setup to publish the Web application under discovery i.e. creating a website publishing rule with appropriate parameters. During application discovery process the application will be accessed via Microsoft ISA, which hosts the Oracle Adaptive Access Manager proxy. Please refer to Microsoft ISA configuration document for details of setting up Microsoft ISA.

Scenarios

Information should be collected for the following scenarios during the discovery process:

Login

1. URL that starts the login process
2. URL that contains the login form
3. Names of the input fields like username, password used to submit the credentials
4. URL to which the login form submits the credentials
5. Identifying successful login. The HTTP traffic dump needs to be studied carefully to derive this information. Here are few ways applications respond on successful login:
6. by setting a specific cookie in the credential submit response
7. by redirecting to a specific URL (like account summary, welcome page)
8. by responding with specific text
9. Identifying failure login with the reason for failure. This would often be derived by looking for certain text in the response to credential submit request.

Logout

1. URL that starts the logout process
2. URL that completes the logout process. In most cases the logout completes on receiving response to the logout start URL.

Change password

1. URL that starts the change password process
2. URL that contains the change password form
3. Names of the input fields like password, new-password, confirm-password used to submit the change password request
4. URL to which the change password form submits the passwords
5. Identifying the status (success/failure) of the change password request. This would often be derived by looking for certain text in the response.

Reset password

Follow the same process as above for Change password.

Change LoginId

1. URL to which the login-id change is posted to the application
2. Names of the input fields like new-login used to submit the change password request.
3. Identifying the status (success/failure) of the change login-id request. On successful change login-id request, changeUserId.do page in Web UIO should be called to update the login-id in the Oracle Adaptive Access Manager database.

Forgot password

Forgot-password options provided by the application should first be understood. Most applications ask for alternate ways to identity the user (account number/pin, SSN/pin, question/answer, etc); some applications provide more than one option. Some applications let the user reset the password on successfully entering alternate credentials; others send a new password to the user by mail/email; and some other applications would require the user to call customer care. For each of the supported scenarios, the following data should be captured:

1. URL that starts the forgot-password process
2. URL that contains the forgot-password form
3. Names of the input fields and URLs to submit the forgot-password request
4. Identifying the status (success/failure) of the forgot-password request.

Samples

The Oracle Adaptive Access Manager proxy configuration to add multi-factor authenticator to BigBank web application is listed here.

```
<?xml version="1.0" encoding="utf-8"?>
<BharosaProxyConfig xmlns="http://bharosa.com/"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:schemaLocation="http://bharosa.com/BharosaProxy.xsd">

  <Application id="BigBank">

    <RequestInterceptor id="AddAppIdTobharosauioRequests"
           desc="Add BharosaAppId header to each request to bharosauio"
           post-exec-action="continue">
      <Conditions>
        <VariableValue name="%URL"
                       value="/bharosauio/"
                       mode="begins-with"
                       ignore-case="true"/>
      </Conditions>

      <Filters>
        <AddHeader name="BharosaAppId:" value="BigBank"/>
      </Filters>
    </RequestInterceptor>

    <!-- Phase-1: Use BigBank login form to collect credentials -->
    <!—- Phase-2: Use BharosaUIO login forms to collect credentials -->

    <!-- Disable this interceptor after phase one is retired -->
    <RequestInterceptor id="Phase1BigBankLoginPostRequest"
                        desc="get the loginid from the post parameters"
                        post-exec-action="continue" enabled="true">
      <RequestUrl url="/bigbank/login.do"/>

      <Conditions>
        <VariableValue name="%REQUEST_METHOD" value="post"/>
      </Conditions>

      <Filters>
        <ClearSession/>
        <SetVariable name="$WebUIOPhase" value="one"/>
        <SaveParam   name="userid"       variable="$userid"/>
      </Filters>
    </RequestInterceptor>

    <!-- Enable this interceptor after phase one is retired -->
    <RequestInterceptor id="Phase2RedirectBigBankLoginPageRequest"
          desc="Redirect BigBank login page requests to UIO login page"
          enabled="false">
      <RequestUrl url="/bigbank"/>
      <RequestUrl url="/bigbank/"/>
      <RequestUrl url="/bigbank/loginPage.jsp"/>
      <Target action="redirect-client" url="/bharosauio/login.do"/>
    </RequestInterceptor>

    <RequestInterceptor id="Phase2BharosaLoginPageRequest"
                        desc="Phase-2 loginid post request"
                        post-exec-action="continue">
      <RequestUrl url="/bharosauio/login.do"/>

      <Conditions>
        <VariableValue name="%REQUEST_METHOD" value="post"/>
        <ParamPresent  name="userid"/>
        <Not>
          <ParamPresent name="password"/>
        </Not>
      </Conditions>

      <Filters>
        <ClearSession/>
        <SetVariable name="$WebUIOPhase" value="two"/>
      </Filters>
    </RequestInterceptor>

    <ResponseInterceptor id="Phase2PassowrdPageResponse"
           desc="Save userid, decoded password from Bharosa response">
      <ResponseUrl url="/bharosauio/password.do"/>

      <Conditions>
        <HeaderPresent name="userid:"/>
        <HeaderPresent name="password:"/>
      </Conditions>

      <Filters>
        <SaveHeader name="userid:"   variable="$userid"/>
        <SaveHeader name="password:" variable="$password"/>
      </Filters>

      <Target action="redirect-client"
              url="/bigbank/login.do"
              display-url="/bigbank/GetLoginPage"/>
    </ResponseInterceptor>

    <ResponseInterceptor id="GetBigBankLoginPageResponse"
             desc="Save hidden fields; then post login crdentials">
      <ResponseUrl url="/bigbank/GetLoginPage"/>

      <Filters>
        <SaveHiddenFields variable="%LoginPageHiddenParams"/>

        <AddHiddenFieldsParams variable="%LoginPageHiddenParams"/>
        <AddParam              name="userid"   value="$userid"/>
        <AddParam              name="password" value="$password"/>

        <UnsetVariable name="$password"/>
      </Filters>

      <Target action="post-server" url="/bigbank/login.do"/>
    </ResponseInterceptor>
```

```
<ResponseInterceptor id="InvalidLoginResponse"
                     desc="Invalid login response from BigBank">
  <ResponseUrl url="/bigbank/login.do"/>

  <Conditions>
    <PageContainsText text="You have entered an invalid Login Id"/>
  </Conditions>

  <Filters>
    <SetVariable  name="$Login-Credentials-Status"
                  value="invalid_user"/>
    <SetVariable  name="$Login-Continue-Url"
                  value="%URL"/>
    <SaveResponse variable="$Submit-Credentials-Response"/>
  </Filters>

  <Target action="redirect-client"
          url="/bharosauio/UpdateLoginStatusPage"/>
</ResponseInterceptor>

<ResponseInterceptor id="WrongPasswordResponse"
                     desc="Invalid login response from BigBank">
  <ResponseUrl url="/bigbank/login.do"/>

  <Conditions>
    <PageContainsText text="We do not recognize your password"/>
  </Conditions>

  <Filters>
    <SetVariable name="$Login-Credentials-Status"
                 value="wrong_password"/>
    <SetVariable name="$Login-Continue-Url"
                 value="%URL"/>
    <SaveResponse variable="$Submit-Credentials-Response"/>
  </Filters>

  <Target action="redirect-client"
          url="/bharosauio/UpdateLoginStatusPage"/>
</ResponseInterceptor>

<ResponseInterceptor id="LoginSuccessResponse"
                     desc="Login success response from BigBank">
  <ResponseUrl url="/bigbank/activity.do"/>
  <ResponseUrl url="/bigbank/login.do"/>

  <Conditions>
    <NotVariableValue name="$Login-Status" value="In-Session"/>
    <PageContainsText text="/bigbank/images/success.gif"/>
  </Conditions>

  <Filters>
    <SetVariable name="$Login-Credentials-Status" value="success"/>
    <SetVariable name="$Login-Continue-Url"       value="%URL"/>
    <SaveResponse variable="$Submit-Credentials-Response"/>
  </Filters>
```

```
      <Target action="redirect-client"
              url="/bharosauio/UpdateLoginStatusPage"/>
    </ResponseInterceptor>

    <RequestInterceptor id="Phase1UpdateLoginStatusPageRequest"
                   desc="Update Bharosa Tracker with the login status">
      <RequestUrl url="/bharosauio/UpdateLoginStatusPage"/>

      <Conditions>
        <VariableValue name="$WebUIOPhase" value="one"/>
      </Conditions>

      <Filters>
        <AddHeader name="WebUIOPhase:"  value="$WebUIOPhase"/>
        <AddHeader name="userid:"       value="$userid"/>
        <AddHeader name="Login-Status:"
                   value="$Login-Credentials-Status"/>
      </Filters>

      <!-- Any interceptors for /bigbank/login.do will not run because
we are doing get-server. -->
      <Target action="get-server" url="/bharosauio/login.do"/>
    </RequestInterceptor>

    <RequestInterceptor id="Phase2UpdateLoginStatusPageRequest"
                   desc="Update Bharosa Tracker with the login status">
      <RequestUrl url="/bharosauio/UpdateLoginStatusPage"/>

      <Filters>
        <AddHeader name="Login-Status:"
                   value="$Login-Credentials-Status"/>
      </Filters>

      <Target action="get-server"
              url="/bharosauio/updateLoginStatus.do"/>
    </RequestInterceptor>

    <ResponseInterceptor id="AllowLoginResponse"
                desc="Tracker returned 'allow' - continue with login">
      <ResponseUrl url="/bharosauio/UpdateLoginStatusPage"/>
      <ResponseUrl url="/bharosauio/updateLoginStatus.do"/>
      <ResponseUrl url="/bharosauio/challengeUser.do"/>
      <ResponseUrl url="/bharosauio/registerQuestions.do"/>
      <ResponseUrl url="/bharosauio/userPreferencesDone.do"/>

      <Conditions>
        <HeaderValue name="Rules-Result:" value="allow"/>
      </Conditions>

      <Filters>
        <SetVariable name="$Login-Status" value="In-Session"/>
      </Filters>

      <Target action="send-to-client"
              html="$Submit-Credentials-Response"
              display-url="$Login-Continue-Url"/>
    </ResponseInterceptor>
```

```
<ResponseInterceptor id="Phase1FailLoginResponse"
                     desc="BigBank failed the login">
  <ResponseUrl url="/bharosauio/UpdateLoginStatusPage"/>
  <ResponseUrl url="/bharosauio/updateLoginStatus.do"/>
  <ResponseUrl url="/bharosauio/challengeUser.do"/>
  <ResponseUrl url="/bharosauio/registerQuestions.do"/>
  <ResponseUrl url="/bharosauio/userPreferencesDone.do"/>

  <Conditions>
    <VariableValue name="$WebUIOPhase" value="one"/>
    <NotVariableValue name="$Login-Credentials-Status"
                      value="success"/>
    <HeaderValue name="Rules-Result:" value="block"/>
  </Conditions>

  <Filters>
    <UnsetVariable name="$Login-Status"/>
  </Filters>

  <Target action="send-to-client"
          html="$Submit-Credentials-Response"
          display-url="$Login-Continue-Url"/>
</ResponseInterceptor>

<ResponseInterceptor id="FailLoginResponse"
                     desc="BigBank failed the login">
  <ResponseUrl url="/bharosauio/UpdateLoginStatusPage"/>
  <ResponseUrl url="/bharosauio/updateLoginStatus.do"/>
  <ResponseUrl url="/bharosauio/challengeUser.do"/>
  <ResponseUrl url="/bharosauio/registerQuestions.do"/>
  <ResponseUrl url="/bharosauio/userPreferencesDone.do"/>

  <Conditions>
    <HeaderValue name="Rules-Result:" value="block"/>
    <NotVariableValue name="$Login-Credentials-Status"
                      value="success"/>
  </Conditions>

  <Filters>
    <UnsetVariable name="$Login-Status"/>
  </Filters>

  <Target action="redirect-client"
          url="/bharosauio/loginPage.jsp?action=invalid_user"/>
</ResponseInterceptor>

<ResponseInterceptor id="BlockLoginResponse"
          desc="BigBank passed login but tracker returned 'block'">
  <ResponseUrl url="/bharosauio/UpdateLoginStatusPage"/>
  <ResponseUrl url="/bharosauio/updateLoginStatus.do"/>
  <ResponseUrl url="/bharosauio/challengeUser.do"/>
  <ResponseUrl url="/bharosauio/registerQuestions.do"/>
  <ResponseUrl url="/bharosauio/userPreferencesDone.do"/>

  <Conditions>
    <HeaderValue name="Rules-Result:" value="block"/>
```

```
      </Conditions>

      <Filters>
        <UnsetVariable name="$Login-Status"/>
      </Filters>

      <!-- /bigbank/LoginBlockedPage isn't a real page.  The request
will be intercepted and redirected. -->
      <Target action="redirect-client"
url="/bigbank/LoginBlockedPage"/>
    </ResponseInterceptor>

    <RequestInterceptor id="LoginBlockedPageRequest"
                        desc="logoff the session in BigBank">
      <RequestUrl url="/bigbank/LoginBlockedPage"/>

      <Target action="get-server" url="/bigbank/logout.do"/>
    </RequestInterceptor>

    <ResponseInterceptor id="Phase1LoginBlockedPageResponse"
          desc="BigBank approved; but Bharosa blocked the login"
          post-exec-action="stop-intercept">
      <ResponseUrl url="/bigbank/LoginBlockedPage"/>

      <Conditions>
        <VariableValue name="$WebUIOPhase" value="one"/>
      </Conditions>

      <Filters>
        <ClearSession/>
      </Filters>

      <Target action="redirect-client"
              url="/bharosauio/loginFail.do?action=block"/>
    </ResponseInterceptor>

    <ResponseInterceptor id="Phase2LoginBlockedPageResponse"
         desc="BigBank approved; but Bharosa blocked the login">
      <ResponseUrl url="/bigbank/LoginBlockedPage"/>

      <Filters>
        <ClearSession/>
      </Filters>

      <Target action="redirect-client"
              url="/bharosauio/loginPage.jsp?action=block"/>
    </ResponseInterceptor>

    <ResponseInterceptor id="LogoutPageResponse"
          desc="Bharosa logout selected; logoff BigBank session ">
      <ResponseUrl url="/bharosauio/logout.do"/>

      <Target action="redirect-client" url="/bigbank/logout.do"/>
    </ResponseInterceptor>

    <ResponseInterceptor id="Phase1LogoffPageResponse"
                         desc="Logoff - clear Bharosa proxy session"
                         post-exec-action="stop-intercept">
      <ResponseUrl url="/bigbank/logout.do"/>

      <Conditions>
        <VariableValue name="$WebUIOPhase" value="one"/>
      </Conditions>

      <Filters>
        <ClearSession/>
      </Filters>
    </ResponseInterceptor>

    <ResponseInterceptor id="Phase2LogoffPageResponse"
                         desc="Logoff - clear Bharosa proxy session">
      <ResponseUrl url="/bigbank/logout.do"/>

      <Filters>
        <ClearSession/>
      </Filters>

      <Target action="redirect-client"
              url="/bharosauio/loginPage.jsp"/>
    </ResponseInterceptor>
  </Application>
</BharosaProxyConfig>
```

Troubleshooting

Enable tracing to file and set the trace level to 0x8008f. This will print detailed interceptor evaluation and execution information to the log file.

References

1. ISA server installation, configuration document
2. XML schema for the Oracle Adaptive Access Manager proxy configuration