

# SINGLE SIGN-ON INTEGRATION WITH PKI

*Dan Norris, norris (at) celeritas (dot) com, Celeritas Technologies, LLC*

## INTRODUCTION

This paper will briefly review what PKI is and how it works to provide authentication. The bulk of the paper will document how you might integrate a third-party (non-Oracle) PKI authentication into the 9iAS Single Sign-on (SSO) to provide user authentication without ever seeing the normal username/password dialog displayed by default in 9iAS SSO installations.

The steps and concepts documented here are relevant to 9iAS version 9.0.2 infrastructure. As long as the 9iAS Infrastructure is version 9.0.2, the solution shown in this paper should work regardless of what version of the middle tier is used.

## PUBLIC KEY INFRASTRUCTURE (PKI) OVERVIEW

### WHAT IS PKI?

Public Key Infrastructure (PKI) has many definitions depending on your background, role in the IT industry, and your use of or interest in PKI technology. The PKIX working group of the Internet Engineering Task Force (IETF) (<http://www.ietf.org/html.charters/pkix-charter.html>) has produced many RFCs related to PKI. The most general RFCs related to PKI are 2459, 2510, 2511, 2527, 2528, 2875, 3279, and 3280. The most broad and all-encompassing definition I've found is: "PKI is the set of hardware, software, people, policies, and procedures needed to create, store, manage, distribute, and revoke public-key certificates."<sup>1</sup> Other definitions are widely varied:

- PKI is the foundation for offering scaleable key and certificate life cycle management.
- PKI enables other security services including data confidentiality, data integrity, and key management.

Most importantly, just like you can't point to a single thing in your office and say "There's the network," you can't point to a single thing in your enterprise and call it "the PKI." PKI is comprised of many different components that, when configured properly together, can provide an infrastructure suitable for public-key creation, management, and distribution.

A PKI includes:

- End entities (generic term referring to end users, servers, devices, appliances, or routers)
- An optional registration authority (usually a person that assists with enrollment of new users to the CA)
- A certificate and CRL repository (usually a directory like X.500 or LDAP). This component is highly secure due to the sensitive nature of the data residing in the directory (public certificates and certificate revocation lists).
- One or more certification authority(ies) (CA) that issues certificates and "certifies" identities. This component is highly secure, guarded, and trusted by everyone to correctly identify end entities.

### HOW DOES PKI WORK?

Public-key cryptography is based on key *pairs*. These pairs of keys (one *private* key and one *public* key) are kept separate and both keys are required in order for communication to be carried out. The two keys are mathematically related to one other, but deriving one key from the other, while possible, is considered computationally infeasible since the amount of compute power, memory, and time necessary to compute one key given the other is formidable. A piece of data encrypted using one key from the pair can only be decrypted using the other key from the pair.

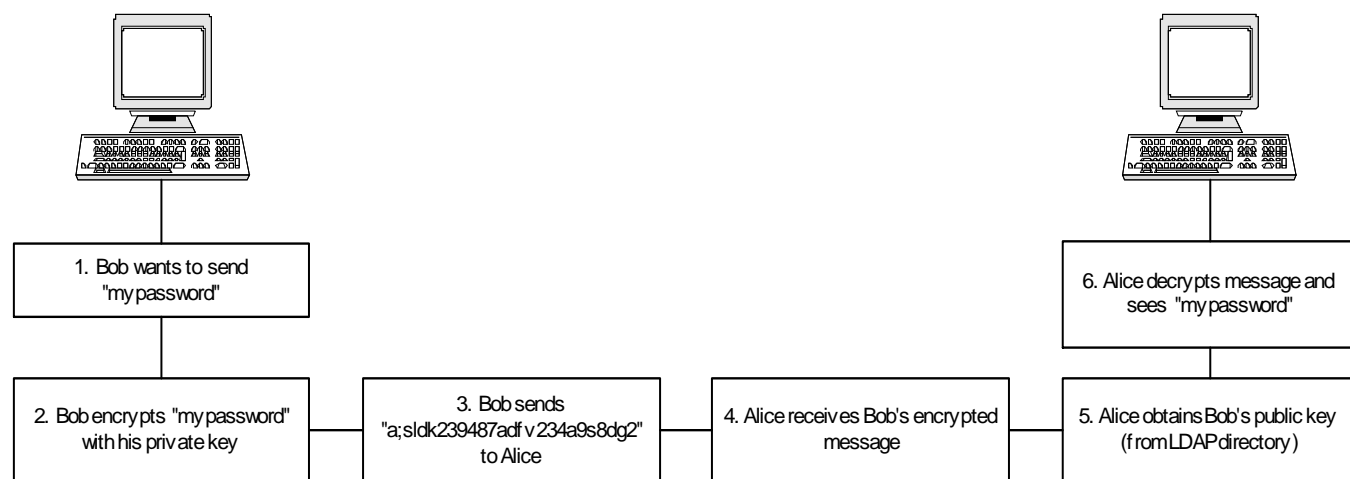
---

<sup>1</sup> S. Kiran, P. Lareau, S. Lloyd: PKI Basics - a Technical Perspective: November 2002; <http://www.pkiforum.org/>

The private key is held safely and securely by only one party and is never shared with anyone. An individual may use their private key to encrypt some data. When that encrypted message data is sent to another party, the receiver will know that the data must positively be a message from the sender since the sender's public key was able to decrypt the message. In the case of authentication, it may be common for the receiver to re-encrypt this clear text data using their own private key and send it back to the original sender. The original sender would then obtain the public key for the original receiver (now the sender) and decrypt the message. Once decrypted, it would be compared with the originally-sent message. If the two messages matched, then the authentication may be completed.

Instead of exchanging a password, PKI is more commonly used to exchange an encryption key (symmetric key) which may have been dynamically generated and be used to encrypt the rest of the "conversation" between the two parties. Symmetric ciphers (for example, Blowfish, AES, 3DES, RC4, RC5, IDEA) are typically much faster than asymmetric ciphers (for example, RSA) and it is common for asymmetric ciphers to be used to exchange symmetric cipher keys to "set up" an encrypted session.

A typical exchange is depicted in the following diagram:



**Figure 1: a PKI "conversation"**

In Figure 1, you can see that Alice now trusts Bob's identity because she was able to obtain Bob's public key and decrypt the message. If Bob's public key had not decrypted the message, then Alice would be certain that Bob did not send the message since only messages encrypted with Bob's private key can be decrypted with his public key.

However, after the exchange takes place, Bob still does not know who Alice is for certain. So, the same process would be repeated in reverse and if Bob was able to get the same message back from Alice, then he is assured that she encrypted that message with her private key because her public key decrypted the message. Therefore, he is sure that he was communicating with Alice and no one else.

It is important to note that all PKI systems assume that the private key is safe and protected. If someone's private key was compromised, it is common practice to revoke the entire key pair and generate a new one. One step not shown in the diagram above was that when obtaining the public key from the public key repository (LDAP, X.500 or other repository), there is usually a check done against the Certificate Revocation List (CRL), which is usually published in the same repository as the public keys to verify that the sender's keys have not been revoked. If the sender's key pair has been revoked (and they are on the CRL), the communication stops and the two parties cannot communicate until a valid public key is made available. The CRL checking feature is built in to most PKI software and is an invisible step to the parties involved. The same scenario occurs if the public key is stolen or replaced in the repository. You can see that the safety and integrity of the repository is critical because it can serve as the launching point for an attack on the entire trust pyramid.

While there are too many uses to include all of them here, the most popular uses for PKI technology are authentication, key exchange, and digital signatures. For each of these mechanisms, the start of the conversation is

usually the same. The two parties involved (may be two individuals or may be an individual and an application) exchange some information to ensure that they trust the other party's identity. In the case of authentication, this exchange is all that's required (since authentication is just ensuring that the other party has a certain identity).

### **SECURE SOCKETS LAYER (SSL) VERSUS PKI**

Secure Sockets Layer (SSL) is commonly used to secure HTTP traffic for web-based applications. SSL is based on PKI, but relies heavily on the user to properly authenticate the identity of the site they're visiting. The major web browser software vendors have incorporated a list of trusted certificate authorities (CA) into their browsers. By using a particular browser, you are implying that you also trust these CAs (IE trusts over 100 CAs by default, Mozilla over 60). This is because, by default, these browsers are configured to allow the sites that have certificates signed the CAs you trust to be displayed without prompting. This is usually safe and most users generally trust that the browser manufacturers have done a good job to verify the authenticity of the "root" CA certificates that they incorporate into the browser.

However, when a warning is presented in the browser stating that the authenticity of the certificate is not guaranteed by any of the CAs that your browser trusts, it is up to the end user to determine whether or not to proceed. If they do proceed, the session will still be encrypted, but it will not ensure that the site your browser is communicating with is actually the site you intended to visit. This could be due to the site being attacked and your browser being redirected (maliciously) to an alternate site that was unable to hijack the private key associated with the SSL certificate (public key) because that identity is guarded with a private key that only the site's rightful owner should know. When attempting to obtain certification from one of the "trusted" CAs, the CA will follow procedures to ensure that the applicant actually is the rightful owner of the site by checking DNS records and matching the DNS record ownerships with the name or company of the applicant. The exact procedures vary from CA to CA, but they generally follow the same guidelines. While there may not be heavy regulations place on the CA industry, most companies provide guarantees that their certificates are authentic by offering compensation for any compromise on their part.

While bidirectional authentication is possible with SSL, it is almost never used. That is, the end user (or browser) uses SSL to ensure that the site they're visiting is authentic, but the site does not attempt to authenticate the end user. Bidirectional authentication occurs when the web server requests a client certificate from the client visiting the server. This is handled by most browsers automatically. Once the server receives the client certificate, it will inspect the certification authority information in that certificate to see what CA has "guaranteed" the identity of the end user. If that CA is trusted by the web server, then the client's identity is deemed authentic and can be used by the web server or any web application to identify the user to the application. Typically, the distinguished name (DN) of the client's certificate is used as the client's username.

It should also be noted that SSL is not specific to HTTP, but can be used to generically provide secure communications for any protocol. It is commonly found in HTTP or LDAP servers.

While HTTPS requires server authentication, LDAPS does not and SSL can be used for encryption only without server authentication. It is not necessarily common practice to configure LDAPS in this manner, but it is possible and, in particular, the Oracle Internet Directory (OID) server allows this type of configuration. This can be useful when you wish to provide some encryption, but do not wish to go through the trouble of obtaining a signed certificate from a CA that is trusted by your clients/customers. In this type of configuration, an Oracle Wallet is still necessary, but the CA that signs the certificate for the LDAP server can be any CA since no one will be required to trust that CA's identity or its signed certificates. Many people use an OpenSSL CA for this purpose (because it's free). Oracle Metalink has published notes on how to configure OpenSSL for use as a standalone CA capable of signing certificates for testing internal development systems: 124689.1 and 117022.1 (may not apply directly to 9iAS 9.0.2, but can be used with any stock OpenSSL installation).

The normal unidirectional authentication used by SSL with HTTPS is shown in Figure 2.

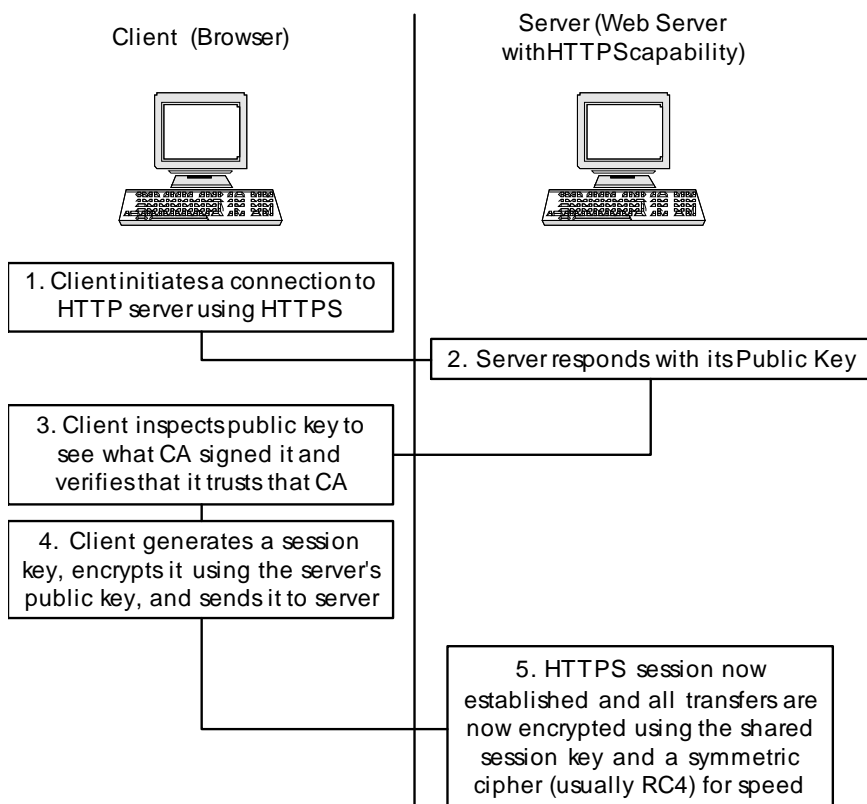


Figure 2: SSL Server Authentication

### **IMPLEMENTING PKI AUTHENTICATION WITH ORACLE 9iAS SINGLE SIGN-ON (SSO)**

With a basis for understanding PKI in place, we're ready to describe how one may integrate PKI into your 9iAS SSO environment. There are several partners that Oracle has worked with to provide integration of third-party SSO products into the 9iAS SSO environment. Some of these partners are listed at [http://otn.oracle.com/products/ias/9ias\\_partners.html#security](http://otn.oracle.com/products/ias/9ias_partners.html#security). Additionally, in the 9iAS 9.0.2 Single Sign-on Administrator's Guide, there is example code to implement integration with Netegrity's SiteMinder product. In addition to the partners documented in these locations, Oracle has also worked with Baltimore Technologies to integrate the SelectAccess application into the 9iAS environment and documentation for this integration is available in a whitepaper from Baltimore Technologies.

The information contained in the remainder of this paper was from experiences gained while integrating 9iAS SSO with a PKI product named Entrust TruePass. This product allows those with valid Entrust PKI credentials to achieve single sign-on to any TruePass-protected web resource. Unlike many of the other solutions that Oracle has chosen to partner with and certify, TruePass provides only authentication by default. Other products like Netegrity SiteMinder and Baltimore Technologies SelectAccess provide the authentication mechanisms as well as policy management including authorization to view certain pages on the site.

Due to choices made in the way that TruePass and Oracle 9iAS SSO implemented their procedures, some special considerations had to be made in order to make all facilities in 9iAS functional. In particular, any service using mod\_osso had to be deployed in a middle-tier server and could not be used directly without authenticating first to a true SSO partner application (like Oracle Portal). The reasons for these restrictions are beyond the scope of this paper (contact me if you would like more information).

The architecture of the Entrust TruePass solution with 9iAS SSO is depicted in Figure 3.

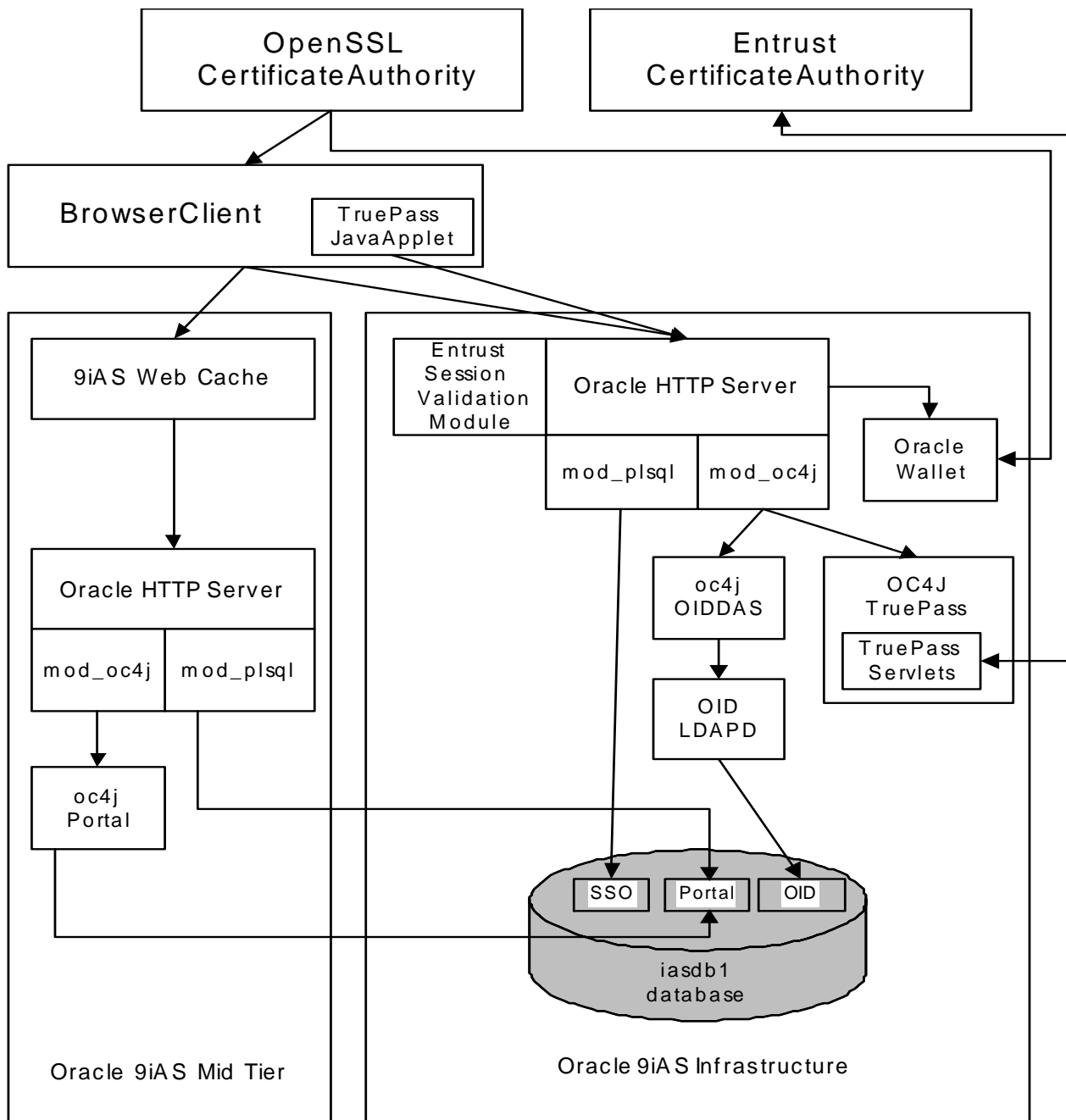


Figure 3: 9iAS Architecture with PKI Integration

### IMPLEMENTATION OVERVIEW

Oracle describes the third-party interface in chapter 5 of the 9iAS Single Sign-on Administrator's Guide (version 9.0.2, part number A96115-01). The main function that needs to be implemented is `ORASSO.WWSSO_AUTH_EXTERNAL.AUTHENTICATE_USER`. This function has one OUT parameter (`p_user`) which is the SSO username of the user. The username set in the `p_user` parameter must be found as an entry under the default LDAP context for the SSO server. Essentially, it is up to you to use some information you can gather from the client and match that up with an entry in the default LDAP context for the SSO server. If this function fails for any reason or fails to assign the `p_user` parameter, then the normal username/password dialog is displayed by the SSO server. So, it may be confusing at first if you don't implement everything correctly the first time

and you may assume that your code is simply being bypassed. I found it very helpful to use the UTL\_FILE package to write debug messages to a file while developing my customized package body.

Oracle provides a "blank" package specification skeleton for you to customize at `$INFRA_HOME/sso/admin/plsql/sso/ssoauthx.pks`. Based on this package specification, you need to create the corresponding package body.

The Oracle HTTP Server is usually customized to load a module from the vendor of the third-party software so that individual HTTP requests sent to the server can be intercepted by the third-party vendor's module and inspected to determine whether or not authentication is necessary to access the particular page requested. If authentication is required, the module will dynamically redirect the browser to a sign-on dialog that is specific to the third-party vendor. Once the authentication requirements are met, the client browser is sent back to the page they originally requested. In principle, this fits well with the 9iAS SSO architecture since it essentially works in the same manner and only requires authentication when the user attempts to access a protected resource.

The steps for integrating PKI (specifically, Entrust TruePass) with 9iAS SSO are:

1. Create an install the WWSSO\_AUTH\_EXTERNAL package in the ORASSO schema.
2. Deploy the TruePass servlets.
3. Add TruePass directives to the httpd.conf and dads.conf files.
4. Modify the SSO server URLs.
5. Re-register the mod\_osso module in each tier with the SSO server.
6. Re-register the Oracle Portal partner application with the SSO server.

Following are sections detailing each of the high-level steps above.

## **ENTRUST TRUEPASS SPECIFICS**

### *WWSSO\_AUTH\_EXTERNAL PACKAGE*

In the case of Entrust TruePass, the Apache module installed is called the Server Validation Module (SVM) and checks each request to see if it is protected with any "AuthType TruePass" or "require TruePass" directives in the httpd.conf file. If so, the user is sent to the TruePass sign-on screen (whose location is set as part of the SVM configuration). Once the user successfully authenticates to the SVM, then an additional HTTP header is set and sent from the client to the server. This header (HTTP\_ENTRUST\_CLIENT) contains the base64-encoded DN of the Entrust client certificate.

This header is easily decoded in the authenticate\_user function and is used to "map" an Entrust user to a 9iAS SSO user. In my case, I simply added a new attribute to the LDAP schema (celeritasentrustdn) to store the DN of the Entrust certificate as an attribute of a 9iAS SSO user. When a user was successfully authenticated with TruePass, the function decodes their DN, and does an LDAP query to see if the user has an LDAP entry already. If they do have an entry (there should be exactly one entry returned from the query), then the entry returned is their SSO identity and the p\_user parameter is set accordingly and the function is completed. If they do not already have an entry in the directory (no query results returned), another procedure in the same package (add\_oid\_user) is called to create the user's entry in OID. In this case, since all we know about the user is their DN (from the HTTP\_ENTRUST\_CLIENT header), an additional query is performed against the Entrust directory to retrieve some information about the user (their name, email address, et cetera), so that their entry in OID is created with some personal information already populated. A new OID RDN is generated using a database sequence (to ensure uniqueness) and the entry is in OID created using the DBMS\_LDAP package. One downside to this mechanism is that a username and password is required to be available to the procedure to bind to the directory for the purpose of creating this new entry. Once the new entry is created, the authenticate\_user function assigns this newly-created username to the p\_user parameter and execution proceeds.

The complete WWSSO\_AUTH\_EXTERNAL package body is included at the end of this paper as Appendix A.

In order for the deployment of the package body to be successful, you will need to meet the following prerequisites:

- The following SQL statements must be run in the iasdb database in the infrastructure:

```

SQL> conn somedba@iasdb
SQL> create user celeritas identified by somepass;
SQL> grant create session, create sequence, create public synonym to celeritas;
SQL> connect celeritas/somepass@iasdb
SQL> create sequence newuser_seq;
SQL> grant select on newuser_seq to orasso;
SQL> create public synonym newuser_seq for newuser_seq;
SQL> conn somedba@iasdb
SQL> alter user celeritas account lock;
SQL> revoke create session from celeritas;

```

- The `utl_file_dir` initialization parameter for the `iasdb` database must be set to the same location as the `gdebug_dir` variable in the package body. Alternatively, all references to `UTL_FILE` can be removed without harming the functionality of the package. Since user accounts are being created, the information reported with `UTL_FILE` may be used as an audit trail stored outside the database for some additional safety.
- The LDAP entry "cn=celeritascreatessouers" (or whatever DN is set for the `gv_oid_user` variable in the `WWSSO_AUTH_EXTERNAL`) must be created in the OID directory used by SSO. It should have objectclasses `top` and `person` and needs to have a password assigned. The password will appear in clear text in the package body. Alternatively, the password could be stored in a file and read in with `UTL_FILE` or the package body could be wrapped to add additional protection.
- The LDAP user `cn=celeritascreatessouers` (or whatever DN is chosen for `gv_oid_user`) needs to be a member of the group "cn=OracleDASCreateUser,cn=Groups,cn=OracleContext". That can be done by adding the DN of the user (`cn=celeritascreatessouers` in this case) to the `uniquemember` attribute of the group's entry (`cn=OracleDASCreateUser,cn=Groups,cn=OracleContext`).
- An LDAP attribute named `celeritasentrustdn` must be created. Its syntax is DN and it should be single-valued and indexed (so that it is searchable). The equality test should be `distinguishedNameMatch` and the usage, ordering, and substring can be left null. This is where the Entrust PKI DN is stored when a new entry is created for an Entrust user. If your company has their own base OID (object identifier, not Oracle Internet Directory) reserved, you should use it to generate the OIDs needed to create the new attribute and objectclass. If you do not have your own OID reserved, you may want to obtain one from ANSI at [http://www.ansi.org/other\\_services/registration\\_programs/reg\\_org.aspx?menuid=10](http://www.ansi.org/other_services/registration_programs/reg_org.aspx?menuid=10) (\$1000+) or from IANA at <http://www.iana.org/cgi-bin/enterprise.pl> (free). Otherwise, for testing purposes, you can just pick a number with a sufficiently large number of decimal points to avoid collisions with other OIDs that are "officially" assigned.
- An LDAP objectclass named `celeritasuserdn` must be created. The type of the objectclass is "None", its superclass is "orcluserV2" and it has exactly one mandatory attribute: `celeritasentrustdn`. Follow the same rules for the OID (it cannot be the same as any other OID already allocated to any other objectclass or attribute) as described in the previous bullet.
- The script listed in Appendix C must be located at `$(INFRA_HOME)/Apache/Apache/htdocs/cgi/getx500.cgi` and it should be owned by oracle with permissions of 700.
- The perl modules `Convert::ASN1` and `perl-ldap` must be obtained ([www.cpan.org](http://www.cpan.org)) and installed on the perl distribution used to run the `getx500.cgi` script (the path to perl is called out in the first line of the script).
- The following must be added to the `httpd.conf` for the infrastructure to support the CGI script used to pull information from the Entrust directory:

```

<Directory /u01/app/oracle/product/iasinf/Apache/Apache/htdocs/cgi>
    Options ExecCGI
    AddHandler cgi-script .cgi
    UnsetEnv PERL5LIB
</Directory>

```

## TRUEPASS SERVLETS DEPLOYMENT

As mentioned previously, the third-party vendor provides their own sign-on dialog at the appropriate time. For Entrust TruePass, this screen is implemented as a set of java servlets that must be deployed in a suitable servlet engine. For my deployment, we used an OC4J server for this. The TruePass installation provides a sample web.xml file and appropriate directory structures for creating a WAR file that can be deployed in most servlet engines (though they only support specific ones and OC4J isn't one of them). For a successful deployment, the web.xml file had to be modified slightly to deploy correctly and run in the OC4J container. The steps I followed for a successful deployment of the TruePass servlets in the 9iAS Infrastructure follow.

1. Login as the oracle software owner and set the environment (ORACLE\_HOME, PATH, ORACLE\_SID) for the 9iAS infrastructure. Also put the dcmctl and opmnctl commands in your PATH. If they aren't in the PATH, the following examples will have to be called slightly differently, but that's the only issue. I made a symbolic link from \$ORACLE\_HOME/bin/dcmctl to \$ORACLE\_HOME/dcm/bin/dcmctl and also from \$ORACLE\_HOME/bin/opmnctl to \$ORACLE\_HOME/opmn/bin/opmnctl.
2. Shutdown the OEM console to ensure that the following dcmctl commands don't conflict with OEM management activity (oemctl stop).
3. dcmctl createcomponent -ct oc4j -co OC4J\_TruePass
4. Edit \$ORACLE\_HOME/opmn/conf/opmn.xml to add the LD\_LIBRARY\_PATH to the OC4J\_TruePass startup environment. The modified section should look like this when finished (the **bold** text indicates the modifications that were necessary--use the correct paths for your environment):

```
<oc4j maxRetry="3" instanceName="OC4J_TruePass" gid="OC4J_TruePass" numProcs="1">
  <config-file path="/u01/app/oracle/product/iasinf/j2ee/OC4J_TruePass/config/server.xml"/>
  <oc4j-option value="-properties"/>
  <port ajp="3001-3100" jms="3201-3300" rmi="3101-3200"/>
  <environment>
    <prop name="LD_LIBRARY_PATH"
      value="/u01/app/oracle/product/iasinf/lib:/opt/entrust/truepass/lib52mt"/>
  </environment>
</oc4j>
```

5. dcmctl updateconfig -ct opmn
6. opmnctl reload
7. dcmctl restart -co OC4J\_TruePass
8. Then copy the java application to a temporary location (\$TRUEPASS\_HOME is the location to your installation, /opt/entrust/truepass by default):  
cp -r \$TRUEPASS\_HOME/samples/TruePassSample/TruePassSampleApp /tmp/.
9. Make the necessary changes to the /tmp/TruePassSampleApp/WEB-INF/web.xml for the truepass.ini file location and adding additional servlet mappings. My modified web.xml file is included as Appendix B. The only changes you will likely need to make from my example is for the location of the truepass.ini. The example has additional content (shown in bold) from the default file shipped with TruePass. Be sure to use the example I provided.
10. Create the TruePass WAR using these commands (change the pathnames if necessary):  
cd /tmp/TruePassSampleApp  
\$ORACLE\_HOME/jdk/bin/sparc/native\_threads/jar cvf /tmp/truepass.war \*
11. Assuming that you created the WAR file in /tmp/truepass.war, run this command to deploy the application to the newly-created OC4J\_TruePass (should all be on one line):

```
dcmctl deployapplication -a TruePassSampleApp -f /tmp/truepass.war -co OC4J_TruePass -rc /TruePassSampleApp
```

This command will likely cause a timeout to occur, but it will finish within a few minutes. The timeout warning is normal. It may take several minutes to finish deployment. Be patient and do not continue until it finishes. You can verify success by running:

```
dcmctl listapplications -co OC4J_TruePass
```

and see that the output shows the application has been deployed.

12. dcmctl restart -ct ohs -v

- Restart OEM console if desired (oemctl start).

With the necessary servlets deployed into an OC4J container, we can move ahead to the modifications to the HTTP server configuration. The servlets handle all communication with the other PKI components like the certificate repository and perform tasks such as CRL checking.

### *MODIFICATIONS TO ORACLE HTTP SERVER CONFIGURATION (HTTPD.CONF)*

In order for the proper integration between 9iAS SSO and PKI, some HTTP server directives must be added to tell the HTTP server that PKI authentication is required to access some resources. Specifically, the ORASSO mod\_plsql DAD must be protected with PKI directives so that in order to access the ORASSO DAD, you must first be authenticated by the third party sign-on. For Entrust TruePass, this authentication is performed by an Apache module called the Server Validation Module (SVM).

To install the Entrust TruePass Server Validation Module (SVM) into the Oracle HTTP server, we performed the following steps:

- Login as the oracle software owner and set your environment for the 9iAS infrastructure where you installed TruePass. Again, adding dcmctl and opmnctl to your PATH will be helpful, but not necessary.
- Shutdown the OEM console to ensure that the following dcmctl commands don't conflict with OEM management activity (oemctl stop).
- Add an additional environment variable to the \$ORACLE\_HOME/Apache/Apache/bin/apachectl script (modifying the LD\_LIBRARY\_PATH variable in that script). The resulting apachectl section looks like this (section I added is in **bold**):

```
if [ -z "$LD_LIBRARY_PATH" ]
then
    LD_LIBRARY_PATH=/u01/app/oracle/product/infrocs/lib:opt/entrust/truepass/lib52mt
    export LD_LIBRARY_PATH
else

LD_LIBRARY_PATH=/u01/app/oracle/product/infrocs/lib:opt/entrust/truepass/lib52mt:${LD_LIB
RARY_PATH}
    export LD_LIBRARY_PATH
fi
```

- dcmctl stop -ct ohs -v
- dcmctl start -ct ohs -v
- Edit \$ORACLE\_HOME/Apache/Apache/conf/httpd.conf and add the following between the LoadModule fastcgi\_module libexec/mod\_fastcgi.so and the <IfDefine SSL> directives:

```
#####
#####
#####
#### Add Entrust Server Validation Module (SVM)
#####
#####
#####
LoadModule session_validation /opt/entrust/truepass/sessionvalidationmodule/libEtSessionValidatio
n_IHS1319.so
AddModule ApacheApiGlue.cpp

<IfModule ApacheApiGlue.cpp>
TruePassIniFile /opt/entrust/truepass/config/truepass.ini
Alias /TruePassSample /opt/entrust/truepass/samples/TruePassSample/web/replaceclientapiframe

<Location /TruePassSample/secured>
    AuthType TruePass
    require TruePass
```

```

</Location>

<LocationMatch "(/CertificateServlet|/MessageServlet|/PrimaryAuthenticationServlet|/ProfileAc
cessServlet|/ProfileStorageServlet|/TransactionServlet|/receipt.jsp)">
    AuthType TruePass
    require TruePass
</LocationMatch>

</IfModule>

```

```

#####
#####
#####
#### Add Entrust Server Validation Module (SVM)
#####
#####
#####

```

7. Also in the httpd.conf, move the following line:

```
include "/u01/app/oracle/product/iasmt/Apache/Apache/conf/mod_osso.conf"
```

to be located just before the TruePass section you added in the previous step. By default, it is located near the end of the httpd.conf file. You should only include it once, so comment out or delete the original line when relocating it.

8. Now edit the \$ORACLE\_HOME/Apache/modplsql/conf/dads.conf file to add the lines that force TruePass authentication and allow the TruePass information to pass through to the WWSSO\_AUTH\_EXTERNAL package. The resulting orasso DAD looks like this (additions in **bold**):

```

<Location /pls/orasso>
    SetHandler pls_handler
    Order deny,allow
    Allow from all
    AllowOverride None
    ##### TruePass Directives
    AuthType TruePass
    require TruePass
    ##### TruePass Directives
    PlsqlDatabaseUsername orasso
    PlsqlDatabasePassword !b3Jhc3Nv
    PlsqlDatabaseConnectString orainfra.celeritas.com:1521:iasdb
    PlsqlDefaultPage orasso.home
    PlsqlDocumentTablename orasso.wwdoc_document
    PlsqlDocumentPath docs
    PlsqlDocumentProcedure orasso.wwdoc_process.process_download
    PlsqlAuthenticationMode SingleSignOn
    PlsqlPathAlias url
    PlsqlPathAliasProcedure orasso.wwpth_api_alias.process_download
    PlsqlSessionCookieName orasso
    PlsqlCGIEnvironmentList HTTP_ENTRUST_CLIENT
</Location>

```

9. dcmctl updateconfig -ct ohs  
10. dcmctl stop -ct ohs -v  
11. dcmctl start -ct ohs -v  
12. Restart OEM console if desired (oemctl start).

With the HTTP server modifications in place and the SVM loaded, we can now test the Entrust TruePass sign-on independently. The test URL should be <https://orainfra.celeritas.com:4443/TruePassSample/>. Once there, proceed

to the User Login link where you will then be presented with the login dialog. Assuming that the TruePass installation was conducted properly, you should be able to login to TruePass successfully.

With all server configuration performed, the next step is to modify the 9iAS SSO application and the 9iAS SSO partner applications to ensure that those applications are aware that we must now use HTTPS when communicating with the SSO server (HTTPS is a TruePass requirement and generally a good idea for authentication mechanisms).

### *MODIFY THE SSO SERVER URLS*

To support the secure nature of the TruePass sign-on, the SSO server must be configured to use HTTPS for sign-on. Additionally, all partner applications must be updated to redirect to the HTTPS URLs for the SSO server. The following steps were performed to update the SSO server URLs to use SSL (HTTPS).

#### *UPDATE THE SSO SERVER URL*

While logged in as the oracle user with your environment set for the 9iAS Infrastructure, run the following to update the SSO Server URLs that will be used when applications direct users to the SSO Server for authentication:

```
export LD_LIBRARY_PATH=$ORACLE_HOME/lib
$ORACLE_HOME/sso/bin/ssocfg.sh https orainfra.celeritas.com 443
```

#### *UPDATE THE SSO PARTNER APPLICATIONS TO USE NEW SSO URLS*

In each of the 9iAS tiers (infrastructure and each middle tier), set the ORACLE\_HOME and LD\_LIBRARY\_PATH (\$ORACLE\_HOME/lib) environment variables accordingly and then run the following command (substituting the appropriate hostnames and port numbers). The hostname and port numbers given should be the hostname and port number for the tier in which you're running the command--NOT for the SSO server since that information will be retrieved from the infrastructure database automatically. The URLs, therefore, may not use the HTTPS protocol, they may use standard HTTP protocol. If you find that you get segmentation fault messages in one of the HTTP servers following these commands, you likely used the wrong URLs.

With all the right substitutions for your environment, the following command will regenerate the configuration file for the mod\_osso module (\$ORACLE\_HOME/Apache/Apache/conf/osso/osso.conf) and re-register it with the SSO server.

```
$ORACLE_HOME/jdk/bin/java -jar $ORACLE_HOME/sso/lib/ossoreg.jar \
-oracle_home_path $ORACLE_HOME \
-host orainfra.celeritas.com \
-port 1521 \
-sid iasdb \
-site_name "some_appropriate_site_name_to_appear_in_SSO_partner_app_list" \
-success_url https://orainfra.celeritas.com/osso_login_success \
-logout_url https://orainfra.celeritas.com/osso_logout_success \
-cancel_url https://orainfra.celeritas.com/ \
-home_url https://orainfra.celeritas.com/ \
-config_mod_osso TRUE \
-u root \
-sso_server_version v1.2
```

#### *UPDATE THE PORTAL TO USE THE NEW SSO URLS*

With the ORACLE\_HOME set to your middle tier and substituting the correct passwords for each of the users specified, the correct hostname and port of the middle tier server, use the following syntax to update the Portal SSO URLs.

```
cd $ORACLE_HOME/assistants/opca
./ptlasst.csh -i typical -mode SSOPARTNERCONFIG -s portal -sp portalpasswd -c
orainfra.celeritas.com:1521:iasdb -sdad portal -o orasso -odad orasso -host
orainfra.celeritas.com -port 7778 -silent -verbose -sso_p orassopasswd -pa orasso_pa
-pap orasso_papasswd -ps orasso_ps -pp orasso_pspasswd
```

Following these changes, you should restart all processes in each tier (using "opmnctl stopall" and "opmnctl startall") and flush the mod\_plsql caches in each tier by removing the \$ORACLE\_HOME/Apache/mod\_plsql/cache directories from each tier.

## **CONCLUSION**

The 9iAS Infrastructure provides an open interface for overriding the default username/password authentication mechanism. While all vendors do not choose to test their products with 9iAS, the vendors that do have a lot in common in terms of the way in which they integrate with 9iAS. Therefore, by reviewing the tested configurations even if they are not the products you're trying to integrate, some valuable insights can be gained and those insights may be helpful when attempting integration for a set of products that have no clear support for integration with the other.

## **FOR ANY LAWYERS OUT THERE...**

The solution described in this paper is not guaranteed to work and is provided as a guideline with no further support implied or guaranteed. The modifications you make to your environment are your responsibility and neither the author of this paper nor any company mentioned herein will be held liable for any damages you may incur during your testing and/or implementation. That being said, you may wish to contact Oracle and lobby that they provide support for more integration partners.

**APPENDIX A: WSSO AUTH EXTERNAL PACKAGE BODY**

```

CREATE OR replace PACKAGE BODY wssso_auth_external AS

/*****
*** Copyright 2003 Celeritas Technologies, L.L.C.
*** This work was created by Celeritas Technologies, L.L.C. ("Creator").
*** This work and all rights therein and thereto, including copyright rights
*** and/or patent rights, are owned by Creator and/or another entity
*** (collectively, "Owner"). This shall serve as notice of such ownership as
*** of the date of this and associated files or subject matter, if any, as
*** depicted above and/or as depicted with an electronic file date stamp and/or
*** any versions thereof and their associated dates, if any. This work may not
*** be reproduced for any purpose, distributed, modified, reverse-engineered,
*** stored in a retrieval system, transmitted, used, made, offered for sale, or
*** sold, in whole or part, in any form or by any means, electronic,
*** mechanical, audio, photocopying, recording, or otherwise, without the prior
*** written permission of Owner. This work may not be exported unless in
*** compliance with the applicable technology export laws. While this
*** information is presented in good faith and believed to be accurate,
*** Creator does not guarantee satisfactory or any results from reliance upon
*** such information. Creator reserves the right, without notice, to alter or
*** improve the designs, specifications, creations, or works of the subject
*** matter herein. Nothing herein is to be construed as a warranty or
*** guarantee, express or implied, against infringement, or regarding
*** performance, merchantability, fitness, or any other matter with respect
*** to products, processes, or any other subject matter herein, and such
*** warranties and guaranties are expressly disclaimed. Nothing herein is to
*** be construed as a recommendation to use any product or process in conflict
*** with any third party rights in any intellectual property. All products,
*** languages, or trademarked names that are mentioned in this work
*** are acknowledged to be the proprietary property of the respective owner.
*****/
/*****/
/*****/
-- settings for local OID directory
gv_oid_host      CONSTANT VARCHAR2(100) := 'orainfra.celeritas.com';
gv_oid_port      CONSTANT PLS_INTEGER   := 389;
gv_oid_sslport   CONSTANT PLS_INTEGER   := 636;
gv_oid_basedn    CONSTANT VARCHAR2(256) := 'cn=Users,dc=celeritas,dc=com';
gv_oid_user      CONSTANT VARCHAR2(256) := 'cn=entrustcreatessousers';
gv_oid_pass      CONSTANT VARCHAR2(256) := 'password_for_gv_oid_user';
-- settings for Entrust LDAP (X.500) directory
gv_getx500_url   CONSTANT VARCHAR2(10000) := 'http://orainfra.celeritas.com:7777/cgi/getx500.cgi';

/*****/
/*****/
--setup for debugging
gdebug          BOOLEAN := TRUE;
gdebug_dir      VARCHAR2(2000) := '/u01/app/oracle/local/utl_file_dir';
gdebug_file     VARCHAR2(2000) := 'auth_debug.log';

```

```

gv_file          UTL_FILE.FILE_TYPE;

/*****
/*****
FUNCTION randpass
RETURN VARCHAR2 IS
    --BINARY_INTEGER can only go to 2^31 (2147483648)
    v_seed        BINARY_INTEGER := 2134567890;
    v_rand        BINARY_INTEGER;
BEGIN
    DBMS_RANDOM.INITIALIZE(v_seed);
    v_rand := DBMS_RANDOM.RANDOM;
    DBMS_RANDOM.SEED(v_rand);
    v_rand := DBMS_RANDOM.RANDOM;
    DBMS_RANDOM.TERMINATE;
    RETURN TO_CHAR(v_rand);
END randpass;
/*****
/*****
FUNCTION add_oid_user
(p_entrust_dn IN VARCHAR2
,p_fname IN VARCHAR2
,p_lname IN VARCHAR2
,p_email IN VARCHAR2
) RETURN VARCHAR2 IS
    v_array        DBMS_LDAP.MOD_ARRAY;
    v_vals         DBMS_LDAP.STRING_COLLECTION ;
    v_seqnum       NUMBER;
    v_cn           VARCHAR2(6);
    v_dn           VARCHAR2(2000);
    v_retval       PLS_INTEGER;
    v_session      DBMS_LDAP.session;
    v_randpass     VARCHAR2(20);
BEGIN
    DBMS_LDAP.USE_EXCEPTION := TRUE;
    v_session := DBMS_LDAP.init(gv_oid_host,gv_oid_sslport);
    --bind using SSL for encryption, but not for authentication
    --v_retval := DBMS_LDAP.bind_s(v_session, gv_oid_user, gv_oid_pass, 1);
    v_retval := DBMS_LDAP.open_ssl(v_session, '', '', 1);
    --v_retval := DBMS_LDAP.bind_s(v_session, gv_oid_user, gv_oid_pass, DBMS_LDAP.GSLC_SSL_NO_AUTH);
    v_retval := DBMS_LDAP.simple_bind_s(v_session, gv_oid_user, gv_oid_pass);
    --v_retval := DBMS_LDAP.bind_s(v_session, gv_oid_user, gv_oid_pass, DBMS_LDAP.AUTH_NONE);

    IF gdebug THEN
        UTL_FILE.PUT_LINE(gv_file,'*** Adding OID entry for '||p_entrust_dn);
    END IF;
    -- Create and setup attribute array for the New entry
    v_array := DBMS_LDAP.create_mod_array(14);
    --get a new username using a sequence-generated name
    SELECT newuser_seq.nextval INTO v_seqnum FROM dual;
    v_cn := 'x' || LPAD(TO_CHAR(v_seqnum),5,'0');

```

```

v_vals(1) := v_cn;
DBMS_LDAP.populate_mod_array(v_array,DBMS_LDAP.MOD_ADD,'cn',v_vals);
v_vals(1) := p_lname;
DBMS_LDAP.populate_mod_array(v_array,DBMS_LDAP.MOD_ADD,'sn',v_vals);
v_vals(1) := p_fname;
DBMS_LDAP.populate_mod_array(v_array,DBMS_LDAP.MOD_ADD,'givenname',v_vals);
v_vals(1) := p_email;
DBMS_LDAP.populate_mod_array(v_array,DBMS_LDAP.MOD_ADD,'mail',v_vals);
v_vals(1) := p_entrust_dn;
DBMS_LDAP.populate_mod_array(v_array,DBMS_LDAP.MOD_ADD,'celeritasentrustdn',v_vals);
v_vals(1) := 'top';
v_vals(2) := 'person';
v_vals(3) := 'organizationalPerson';
v_vals(4) := 'inetOrgPerson';
v_vals(5) := 'orcluser';
v_vals(6) := 'orcluserv2';
v_vals(7) := 'celeritasuserdn';
DBMS_LDAP.populate_mod_array(v_array,DBMS_LDAP.MOD_ADD,'objectclass',v_vals);
v_vals.DELETE;
--generate a password that is reasonably difficult and meets OID criteria
v_randpass := randpass;
v_vals(1) := p_fname || v_randpass;
DBMS_LDAP.populate_mod_array(v_array,DBMS_LDAP.MOD_ADD,'userpassword',v_vals);
--add displayname to be the "pretty" name, DAN, 7/22/2003
v_vals(1) := p_fname||' '||p_lname;
DBMS_LDAP.populate_mod_array(v_array,DBMS_LDAP.MOD_ADD,'displayname',v_vals);
-- DN for Entry to be Added under 'ldap_base'
v_dn := 'cn=' ||v_cn || ',' || gv_oid_basedn;
IF gdebug THEN
    UTL_FILE.PUT_LINE(gv_file,'*** Adding Entry for DN: ' || v_dn );
END IF;
-- Add new Entry to ldap directory
v_retval := DBMS_LDAP.add_s(v_session,v_dn,v_array);
IF gdebug THEN
    UTL_FILE.PUT_LINE(gv_file,'*** add_s Returns: ' || TO_CHAR(v_retval));
END IF;
-- Free attribute array (v_array)
DBMS_LDAP.free_mod_array(v_array);
v_retval := DBMS_LDAP.unbind_s(v_session);
RETURN v_cn;
END add_oid_user;

/*****
/*****
FUNCTION find_oid_user
(p_entrust_dn IN VARCHAR2
) RETURN VARCHAR2 IS
    v_session    DBMS_LDAP.session;
    v_attrs      DBMS_LDAP.string_collection;
    v_rdns       DBMS_LDAP.string_collection;
    v_retval     PLS_INTEGER;

```

```

v_entry      DBMS_LDAP.message;
v_message    DBMS_LDAP.message;
v_dn         VARCHAR2(2000);
BEGIN
  DBMS_LDAP.USE_EXCEPTION := TRUE;
  v_session := DBMS_LDAP.init(gv_oid_host,gv_oid_port);
  --No bind call needed to do an anonymous bind
  v_attrs(1) := 'dn';
  v_retval := DBMS_LDAP.search_s(v_session, gv_oid_basedn,
                                DBMS_LDAP.SCOPE_ONELEVEL,
                                'celeritasentrustdn='||p_entrust_dn,
                                v_attrs,
                                0,
                                v_message);

  v_retval := DBMS_LDAP.count_entries(v_session, v_message);
  --changed from != 1 to = 0 incase multiple entries are created somehow, DAN, 7/11/2003
  IF v_retval = 0 THEN
    RETURN NULL;
  END IF;
  v_entry := DBMS_LDAP.first_entry(v_session, v_message);
  v_dn := DBMS_LDAP.get_dn(v_session, v_entry);
  --get RDNs without the attribute tags
  v_rdns := DBMS_LDAP.explode_dn(v_dn, 1);
  v_retval := DBMS_LDAP.unbind_s(v_session);
  RETURN v_rdns(v_rdns.first);
END find_oid_user;

/*****
/*****
PROCEDURE get_x500_info
(p_entrust_dn IN VARCHAR2
,p_fname OUT VARCHAR2
,p_lname OUT VARCHAR2
,p_email OUT VARCHAR2
) IS
  v_givenname VARCHAR2(100);
  v_sn VARCHAR2(100);
  v_cn VARCHAR2(100);
  v_mail VARCHAR2(100);
  all_done BOOLEAN;
  req utl_http.req;
  resp utl_http.resp;
  value VARCHAR2(1024);
  url_raw VARCHAR2(10000);
  url_escaped VARCHAR2(10000);
BEGIN
  --DBMS_LDAP would not communicate with the Entrust LDAP directory, to we used
  -- a Perl CGI script that did the LDAP calls and returned the information in a simple format
  --The CGI script takes one parameter which was the Entrust DN for directory lookup

  --we must escape all chars including reserved chars (like "+")

```

```

url_escaped := gv_getx500_url || '?dn=' || utl_url.escape(p_entrust_dn,TRUE);

req := utl_http.begin_request(url_escaped);
resp := utl_http.get_response(req);
LOOP
  BEGIN
    utl_http.read_line(resp, value, TRUE);
  EXCEPTION
    WHEN utl_http.end_of_body THEN
      all_done := TRUE;
  END;
  EXIT WHEN all_done;
  --make sure we don't have too many entries
  IF TRIM(LOWER(SUBSTR(value,0,INSTR(value,':')-1))) = 'entries' THEN
    IF TRIM(SUBSTR(value,INSTR(value,':')+1)) != '1' THEN
      RAISE_APPLICATION_ERROR(-20000,'Too many entries, exiting!');
    END IF;
  END IF;

  IF TRIM(LOWER(SUBSTR(value,0,INSTR(value,':')-1))) = 'cn' THEN
    v_cn := TRIM(SUBSTR(value,INSTR(value,':')+1));
  ELSIF TRIM(LOWER(SUBSTR(value,0,INSTR(value,':')-1))) = 'sn' THEN
    v_sn := TRIM(SUBSTR(value,INSTR(value,':')+1));
  ELSIF TRIM(LOWER(SUBSTR(value,0,INSTR(value,':')-1))) = 'mail' THEN
    v_mail := TRIM(SUBSTR(value,INSTR(value,':')+1));
  ELSIF TRIM(LOWER(SUBSTR(value,0,INSTR(value,':')-1))) = 'givenname' THEN
    v_givenname := TRIM(SUBSTR(value,INSTR(value,':')+1));
  END IF;
END LOOP;
utl_http.end_response(resp);

-- Set return variables
p_email := v_mail;
p_lname := v_sn;
p_email := v_mail;
-- if they don't have a givenname, we'll just use the first part of cn
IF v_givenname IS NULL THEN
  p_fname := TRIM(SUBSTR(v_cn,0,INSTR(v_cn,' ')-1));
ELSE
  p_fname := v_givenname;
END IF;
EXCEPTION
  WHEN utl_http.end_of_body THEN
    utl_http.end_response(resp);
  WHEN others THEN
    RAISE;
END get_x500_info;

/*****/
/*****/
FUNCTION authenticate_user

```

```

(p_user OUT VARCHAR2
)
RETURN PLS_INTEGER
IS
  v_http_header      VARCHAR2(32000);
  v_ssouser          WWSEC_PERSON.USER_NAME%TYPE := NULL;
  v_dn               VARCHAR2(32000);
  v_fname            VARCHAR2(1000);
  v_lname            VARCHAR2(1000);
  v_email            VARCHAR2(1000);

BEGIN
  v_http_header := owa_util.get_cgi_env('HTTP_ENTRUST_CLIENT');
  IF gdebug THEN
    gv_file := UTL_FILE.FOPEN(gdebug_dir,gdebug_file,'a');
    UTL_FILE.PUT_LINE(gv_file,'*** Entrust HTTP_ENTRUST_CLIENT: ' || v_http_header);
  END IF;

  IF v_http_header IS NOT NULL THEN
    --decode DN from Base64 encoding
    v_dn := UTL_RAW.CAST_TO_VARCHAR2(
      UTL_ENCODE.BASE64_DECODE(
        UTL_RAW.CAST_TO_RAW(v_http_header)
      )
    );
    IF gdebug THEN
      UTL_FILE.PUT_LINE(gv_file,'*** Entrust DN: ' || v_dn);
    END IF;

    --do an LDAP search for the username in our OID
    v_ssouser := find_oid_user(v_dn);

    IF gdebug AND v_ssouser IS NOT NULL THEN
      UTL_FILE.PUT_LINE (gv_file,'*** found username ' ||v_ssouser||' for ' ||v_dn);
    END IF;

    --if we didn't find a user, create one
    IF v_ssouser IS NULL THEN
      --first get their name and email from the Entrust directory
      get_x500_info(v_dn,v_fname,v_lname,v_email);

      IF gdebug THEN
        UTL_FILE.PUT_LINE (gv_file,'*** got Entrust dir info: ' ||v_fname||' ' ||v_lname||' (' ||v_email||')
for ' ||v_dn);
      END IF;

      --then create a new OID entry for them with their name info
      v_ssouser := add_oid_user(v_dn,v_fname,v_lname,v_email);

      IF gdebug THEN
        UTL_FILE.PUT_LINE (gv_file,'*** created username ' ||v_ssouser||' based on DN ' ||v_dn);
      END IF;
    END IF;
  END IF;

```

```

END IF;

IF gdebug THEN
  UTL_FILE.PUT_LINE (gv_file,'** SUCCESS: Setting username to '||v_ssouser);
END IF;

--this is a hardcode hack to make sure we can test easily
--v_ssouser := 'portal';

--use this exception if necessary
--RAISE EXT_AUTH_FAILURE_EXCEPTION;

IF gdebug THEN
  UTL_FILE.PUT_LINE (gv_file,'** v_ssouser is: ' || v_ssouser);
END IF;
ELSE
  IF gdebug THEN
    UTL_FILE.PUT_LINE (gv_file,'** HTTP_ENTRUST_CLIENT is null.');
```

END IF;

```

  raise EXT_AUTH_FAILURE_EXCEPTION;
END IF;

p_user := NLS_UPPER(v_ssouser);
IF gdebug THEN
  UTL_FILE.FCLOSE(gv_file);
END IF;
RETURN 0;

EXCEPTION
  WHEN OTHERS THEN
    IF gdebug THEN
      UTL_FILE.PUT_LINE(gv_file,'** unknown exception in authenticate_user(p_user)' || sqlerrm);
      UTL_FILE.FCLOSE(gv_file);
    END IF;
    RAISE EXT_AUTH_FAILURE_EXCEPTION;

END authenticate_user;

/*****/
/*****/
FUNCTION get_authentication_name
RETURN VARCHAR2
AS
BEGIN

  RETURN 'Entrust TruePass';

END get_authentication_name;

/*****/
/*****/
```

```

/*****
PROCEDURE set_external_cookies
(
    p_username IN VARCHAR2
    ,p_password IN VARCHAR2
    ,p_cookie_list OUT wwssols_private.cookie_list
)
AS
BEGIN

    null;

END set_external_cookies;
*****/
/
END;
/
```

## **APPENDIX B: TRUEPASS SERVLETS WEB.XML**

The additions/changes from the default web.xml file shipped with TruePass are in bold below:

```
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.2//EN"
"http://java.sun.com/j2ee/dtds/web-app_2_2.dtd">
<web-app>
  <display-name>TruePass Sample</display-name>

  <servlet>
    <servlet-name>AuthenticationProfileAccessServlet</servlet-name>
    <display-name>AuthenticationProfileAccessServlet</display-name>
    <description></description>
    <servlet-class>EntrustTruePassAuthenticationProfileAccessServlet</servlet-class>
    <init-param>
      <param-name>EntrustTruePassIniFile</param-name>
      <param-value>/opt/entrust/truepass/config/truepass.ini</param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
  </servlet>

  <servlet>
    <servlet-name>AuthenticationServlet</servlet-name>
    <display-name>AuthenticationServlet</display-name>
    <description></description>
    <servlet-class>EntrustTruePassAuthenticationServlet</servlet-class>
    <init-param>
      <param-name>EntrustTruePassIniFile</param-name>
      <param-value>/opt/entrust/truepass/config/truepass.ini</param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
  </servlet>

  <servlet>
    <servlet-name>CertificateServlet</servlet-name>
    <display-name>CertificateServlet</display-name>
    <description></description>
    <servlet-class>EntrustTruePassCertificateServlet</servlet-class>
    <init-param>
      <param-name>EntrustTruePassIniFile</param-name>
      <param-value>/opt/entrust/truepass/config/truepass.ini</param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
  </servlet>

  <servlet>
    <servlet-name>MessageServlet</servlet-name>
    <display-name>MessageServlet</display-name>
    <description></description>
    <servlet-class>EntrustTruePassMessageServlet</servlet-class>
    <init-param>
```

```
    <param-name>EntrustTruePassIniFile</param-name>
    <param-value>/opt/entrust/truepass/config/truepass.ini</param-value>
</init-param>
<load-on-startup>1</load-on-startup>
</servlet>

<servlet>
  <servlet-name>ProfileAccessServlet</servlet-name>
  <display-name>ProfileAccessServlet</display-name>
  <description></description>
  <servlet-class>EntrustTruePassProfileAccessServlet</servlet-class>
  <init-param>
    <param-name>EntrustTruePassIniFile</param-name>
    <param-value>/opt/entrust/truepass/config/truepass.ini</param-value>
  </init-param>
  <load-on-startup>1</load-on-startup>
</servlet>

<servlet>
  <servlet-name>PrimaryAuthenticationServlet</servlet-name>
  <display-name>PrimaryAuthenticationServlet</display-name>
  <description></description>
  <servlet-class>EntrustTruePassPrimaryAuthenticationServlet</servlet-class>
  <init-param>
    <param-name>EntrustTruePassIniFile</param-name>
    <param-value>/opt/entrust/truepass/config/truepass.ini</param-value>
  </init-param>
</servlet>

<servlet>
  <servlet-name>ProfileStorageServlet</servlet-name>
  <display-name>ProfileStorageServlet</display-name>
  <description></description>
  <servlet-class>EntrustTruePassProfileStorageServlet</servlet-class>
  <init-param>
    <param-name>EntrustTruePassIniFile</param-name>
    <param-value>/opt/entrust/truepass/config/truepass.ini</param-value>
  </init-param>
  <load-on-startup>1</load-on-startup>
</servlet>

<servlet>
  <servlet-name>ProfileManagementServlet</servlet-name>
  <display-name>ProfileManagementServlet</display-name>
  <description></description>
  <servlet-class>EntrustTruePassProfileManagementServlet</servlet-class>
  <init-param>
    <param-name>EntrustTruePassIniFile</param-name>
    <param-value>/opt/entrust/truepass/config/truepass.ini</param-value>
  </init-param>
  <load-on-startup>1</load-on-startup>
```

```
</servlet>

<servlet>
  <servlet-name>SecondaryAuthenticationServlet</servlet-name>
  <display-name>SecondaryAuthenticationServlet</display-name>
  <description></description>
  <servlet-class>EntrustTruePassSecondaryAuthenticationServlet</servlet-class>
  <init-param>
    <param-name>EntrustTruePassIniFile</param-name>
    <param-value>/opt/entrust/truepass/config/truepass.ini</param-value>
  </init-param>
</servlet>

<servlet>
  <servlet-name>SecondaryLogoutServlet</servlet-name>
  <display-name>SecondaryLogoutServlet</display-name>
  <description></description>
  <servlet-class>EntrustTruePassSecondaryLogoutServlet</servlet-class>
  <init-param>
    <param-name>EntrustTruePassIniFile</param-name>
    <param-value>/opt/entrust/truepass/config/truepass.ini</param-value>
  </init-param>
</servlet>

<servlet>
  <servlet-name>TransactionServlet</servlet-name>
  <display-name>TransactionServlet</display-name>
  <description></description>
  <servlet-class>EntrustTruePassTransactionServlet</servlet-class>
  <init-param>
    <param-name>EntrustTruePassIniFile</param-name>
    <param-value>/opt/entrust/truepass/config/truepass.ini</param-value>
  </init-param>
  <load-on-startup>1</load-on-startup>
</servlet>

<servlet-mapping>
  <servlet-name>AuthenticationProfileAccessServlet</servlet-name>
  <url-pattern>/servlets/AuthenticationProfileAccessServlet/*</url-pattern>
</servlet-mapping>

<servlet-mapping>
  <servlet-name>AuthenticationServlet</servlet-name>
  <url-pattern>/servlets/AuthenticationServlet/*</url-pattern>
</servlet-mapping>

<servlet-mapping>
  <servlet-name>CertificateServlet</servlet-name>
  <url-pattern>/servlets/CertificateServlet/*</url-pattern>
</servlet-mapping>
```

```
<servlet-mapping>
  <servlet-name>MessageServlet</servlet-name>
  <url-pattern>/servlets/MessageServlet/*</url-pattern>
</servlet-mapping>

<servlet-mapping>
  <servlet-name>PrimaryAuthenticationServlet</servlet-name>
  <url-pattern>/servlets/PrimaryAuthenticationServlet/*</url-pattern>
</servlet-mapping>

<servlet-mapping>
  <servlet-name>ProfileAccessServlet</servlet-name>
  <url-pattern>/servlets/ProfileAccessServlet/*</url-pattern>
</servlet-mapping>

<servlet-mapping>
  <servlet-name>ProfileStorageServlet</servlet-name>
  <url-pattern>/servlets/ProfileStorageServlet/*</url-pattern>
</servlet-mapping>

<servlet-mapping>
  <servlet-name>ProfileManagementServlet</servlet-name>
  <url-pattern>/servlets/ProfileManagementServlet/*</url-pattern>
</servlet-mapping>

<servlet-mapping>
  <servlet-name>SecondaryAuthenticationServlet</servlet-name>
  <url-pattern>/servlets/SecondaryAuthenticationServlet/*</url-pattern>
</servlet-mapping>

<servlet-mapping>
  <servlet-name>SecondaryLogoutServlet</servlet-name>
  <url-pattern>/servlets/SecondaryLogoutServlet/*</url-pattern>
</servlet-mapping>

<servlet-mapping>
  <servlet-name>TransactionServlet</servlet-name>
  <url-pattern>/servlets/TransactionServlet/*</url-pattern>
</servlet-mapping>

<servlet-mapping>
  <servlet-name>AuthenticationProfileAccessServlet</servlet-name>
  <url-pattern>/servlets/AuthenticationProfileAccessServlet</url-pattern>
</servlet-mapping>

<servlet-mapping>
  <servlet-name>AuthenticationServlet</servlet-name>
  <url-pattern>/servlets/AuthenticationServlet</url-pattern>
</servlet-mapping>

<servlet-mapping>
```

```
<servlet-name>CertificateServlet</servlet-name>
<url-pattern>/servlets/CertificateServlet</url-pattern>
</servlet-mapping>

<servlet-mapping>
  <servlet-name>MessageServlet</servlet-name>
  <url-pattern>/servlets/MessageServlet</url-pattern>
</servlet-mapping>

<servlet-mapping>
  <servlet-name>PrimaryAuthenticationServlet</servlet-name>
  <url-pattern>/servlets/PrimaryAuthenticationServlet</url-pattern>
</servlet-mapping>

<servlet-mapping>
  <servlet-name>ProfileAccessServlet</servlet-name>
  <url-pattern>/servlets/ProfileAccessServlet</url-pattern>
</servlet-mapping>

<servlet-mapping>
  <servlet-name>ProfileStorageServlet</servlet-name>
  <url-pattern>/servlets/ProfileStorageServlet</url-pattern>
</servlet-mapping>

<servlet-mapping>
  <servlet-name>ProfileManagementServlet</servlet-name>
  <url-pattern>/servlets/ProfileManagementServlet</url-pattern>
</servlet-mapping>

<servlet-mapping>
  <servlet-name>SecondaryAuthenticationServlet</servlet-name>
  <url-pattern>/servlets/SecondaryAuthenticationServlet</url-pattern>
</servlet-mapping>

<servlet-mapping>
  <servlet-name>SecondaryLogoutServlet</servlet-name>
  <url-pattern>/servlets/SecondaryLogoutServlet</url-pattern>
</servlet-mapping>

<servlet-mapping>
  <servlet-name>TransactionServlet</servlet-name>
  <url-pattern>/servlets/TransactionServlet</url-pattern>
</servlet-mapping>

<session-config>
  <session-timeout>30</session-timeout>
</session-config>
<mime-mapping>
  <extension>txt</extension>
```

```
    <mime-type>text/plain</mime-type>
</mime-mapping>
<mime-mapping>
  <extension>html</extension>
  <mime-type>text/html</mime-type>
</mime-mapping>
<welcome-file-list><welcome-file>index.jsp</welcome-file></welcome-file-list>
</web-app>
```

**APPENDIX C: PERL CGI SCRIPT GETX500.CGI**

```
#!/usr/bin/perl

### Copyright (c) 2002 Celeritas Technologies, L.L.C.
### This work was created by Celeritas Technologies, L.L.C. ("Creator").
### This work and all rights therein and thereto, including copyright rights
### and/or patent rights, are owned by Creator and/or another entity
### (collectively, "Owner"). This shall serve as notice of such ownership as
### of the date of this and associated files or subject matter, if any, as
### depicted above and/or as depicted with an electronic file date stamp and/or
### any versions thereof and their associated dates, if any. This work may not
### be reproduced for any purpose, distributed, modified, reverse-engineered,
### stored in a retrieval system, transmitted, used, made, offered for sale, or
### sold, in whole or part, in any form or by any means, electronic,
### mechanical, audio, photocopying, recording, or otherwise, without the prior
### written permission of Owner. This work may not be exported unless in
### compliance with the applicable technology export laws. While this
### information is presented in good faith and believed to be accurate,
### Creator does not guarantee satisfactory or any results from reliance upon
### such information. Creator reserves the right, without notice, to alter or
### improve the designs, specifications, creations, or works of the subject
### matter herein. Nothing herein is to be construed as a warranty or
### guarantee, express or implied, against infringement, or regarding
### performance, merchantability, fitness, or any other matter with respect
### to products, processes, or any other subject matter herein, and such
### warranties and guaranties are expressly disclaimed. Nothing herein is to
### be construed as a recommendation to use any product or process in conflict
### with any third party rights in any intellectual property. All products,
### languages, or trademarked names that are mentioned in this work
### are acknowledged to be the proprietary property of the respective owner.

#####
#####
###
### getx500.cgi - a script to query Entrust directory for user information
###
### This script is called from the ORASSO.WSSO_AUTH_EXTERNAL.GET_X500_INFO
### function in the 9iAS infrastructure database (iasdb). DBMS_LDAP
### refused to talk to the directory used by Entrust.
###
### Dan Norris, Celeritas Technologies, LLC 3/4/2003
###
#####
#####

### LDAP host to query
$lldaphost = 'entrustdir.celeritas.com';
### attributes to retrieve (or attempt to retrieve)
$attrs = ['cn','sn','givenname','mail'];

#####
#####
### end of servicable parts
#####
#####

use CGI qw/:standard/;
### need to install perl-ldap and Convert::ASN1 for this to work
use Net::LDAP;

$DEBUG = 0;

print header('text/plain');
```

```

### get the DN from the parameters
$dn = param('dn');
if ( $dn =~ /^$/ ) {
    print "no argument passed, exiting\n";
    print STDERR "no argument passed, exiting\n";
    exit 1;
}
print "dn is: $dn\n" if $DEBUG;
$base = substr($dn,index($dn,',') + 1);
$cn = substr($dn,0,index($dn,','));
if ( $cn =~ /\+/ ) {
    @parts = split(/\+/, $cn);
    foreach $i (0 .. $#parts) {
        if ( $parts[$i] =~ /cn=/ ) {
            $cn=$parts[$i];
            break;
        }
    }
}

print "base is: $base\n" if $DEBUG;
print "cn is: $cn\n" if $DEBUG;

### do the LDAP query
$ldap = Net::LDAP->new($ldaphost) or die "$@";
$ldap->bind ; # an anonymous bind
$result = $ldap->search
    (base => "$base"
    ,filter => "($cn)"
    ,scope => "one"
    ,attrs => $attrs
);
$result->code && die $result->error;

if ($DEBUG) {
    foreach $entry ($result->all_entries) { $entry->dump; }
    exit;
}

### the code that reads this output will use this information to populate OID
print "entries: ", $result->count, "\n";
@entries = $result->all_entries;
foreach $entry (@entries) {
    print "dn: ", $entry->dn, "\n";
    foreach $attr (sort $entry->attributes) {
        next if ( $attr =~ /;binary$/ );
        print "$attr: ", $entry->get_value($attr), "\n";
    }
}
$ldap->unbind;

```