ORNL/TM-2019/1291

Multi–LDAP Dynamic User Management Tool Utilizing Oracle Application Express and PL/SQL

A. Bengston

August 26, 2019

OAK RIDGE NATIONAL LABORATORY
MANAGED BY UT-BATTELLE FOR THE US DEPARTMENT OF ENERGY

ORNL/TM-2019/1291

Data System Sciences and Engineering
Center for Infrastructure Security Analysis

Multi–LDAP Dynamic User Management Tool Utilizing Oracle Application Express and PL/SQL

Adam Bengston

August 26, 2019

Prepared by
OAK RIDGE NATIONAL LABORATORY
Oak Ridge, TN 37831-6283
managed by
UT-BATTELLE, LLC
for the
US DEPARTMENT OF ENERGY
under contract DE-AC05-00OR22725

CONTENTS

ACRONYMS

AD	Microsoft Active Directory
ADLDS	Microsoft Active Directory Lightweight Directory Services
APEX	Oracle Application Express
API	application programming interface
ID	Identifier
CISA	Center for Infrastructure Security Analysis
LDAP	Lightweight Directory Access Protocol
OID	Oracle Internet Directory
OUMA	Operational User Management Application
PL/SQL	Oracle Procedural Language extension for Structured Query Language

ABSTRACT

A Center for Infrastructure Security Analysis (CISA) client system required support of multiple Lightweight Directory Access Protocol (LDAP) directories for multiple, varied user communities. The LDAP directories were all based upon different technologies, i.e., Microsoft Active Directory (AD), Microsoft Active Directory Lightweight Directory Services (ADLDS), and Oracle Internet Directory (OID), depending on system setup, security requirements, and legacy software. Research revealed that PL/SQL provided the capability to execute dynamic PL/SQL and SQL blocks. The dynamic capabilities were utilized along with several configuration tables to provide an object-oriented, polymorphic-like behavior. This architecture was chosen so that a top-level application programming interface (API) could be defined for all calls, no matter what type of LDAP was being manipulated, as each LDAP type has slightly different attributes and requirements. The dynamic nature of the user management system allowed for easy setup and utilization of any new LDAP directories, minimizing the time and effort expended on system changes.

1. SCOPE

This document is being provided to describe the integration of the following three major technologies at the Oak Ridge National Laboratory (ORNL) Center for Infrastructure Security Analysis (CISA): 1) Lightweight Directory Access Protocol (LDAP), 2) Oracle Application Express (APEX), and 3) Oracle Procedural Language extension for Structured Query Language (PL/SQL). This document is not intended to provide background information on those technologies or to provide an in-depth discussion on any one of those technologies.

2. INTRODUCTION

A CISA client system required support of multiple LDAP directories for multiple, varied user communities. The LDAP directories were all based upon different technologies, i.e., Microsoft Active Directory (AD), Microsoft Active Directory Lightweight Directory Services (ADLDS), and Oracle Internet Directory (OID), depending on system setup, security requirements, and legacy software (from which the CISA researchers initially chose not to migrate). With the requirement to support several different LDAP directories came the need to support user management activities from an application or applications.

3. SOLUTION

Appendix A provides an overview of the multi–LDAP dynamic user management tool architecture and the components detailed in this section.

3.1 PL/SQL

The CISA client project is completely Oracle-based, so, from the onset, CISA researchers sought a solution that involved utilizing Oracle PL/SQL language for writing code within the database. Research revealed that PL/SQL provided the capability to execute dynamic PL/SQL and SQL blocks. The dynamic capabilities were utilized along with several configuration tables to provide an object-oriented, polymorphic-like behavior. This architecture was chosen so that a top-level application programming interface (API) could be defined for all calls, no matter what type of LDAP was being manipulated, as each LDAP type has slightly different attributes and requirements.

Oracle also provides a core package, DBMS_LDAP, to interact with LDAP directories. This package was used at the core of all the PL/SQL that was written for this application.

3.2 TABLES

At the root of this package are a set of configuration tables that were created to store information on the connected LDAP directories, as well as information identifying the applications that would be utilizing each LDAP. Sections 3.2.1, *LDAP_SERVER_CONFIG*, and 3.2.2, *APPLICATION_REGISTRY*, discuss the tables created to store the necessary metadata.

3.2.1 LDAP_SERVER_CONFIG

The LDAP_SERVER_CONFIG table contains metadata for each LDAP the user management tool was required to support. This information includes the following:

1. ldap_server_id – Unique identifier for each LDAP entered into the system.
2. name – Text name for the LDAP to be displayed within the application.
3. host – Server that hosts the LDAP.
4. port – Port used to communicate with the LDAP on the host.
5. admin_user_name – Login of the administrative user used to execute commands within the LDAP.
6. admin_user_password – Encrypted password for the administrative user; keys stored elsewhere.
7. auth_base – Path utilized to authenticate the administrative user.
8. full_base – Root path.
9. user_base – Path used to access the users.
10. group_base – Path used to access the groups with which users are associated.
11. is_active – Flag indicating whether or not the LDAP should be listed within the application.
12. package_prefix – PL/SQL package prefix used within the dynamic PL/SQL code to determine which LDAP implementation code to utilize.
13. view_prefix – Prefix for the three distinct views that are built for each LDAP entered into the system.
14. replicate_to – Indicates a slave LDAP to which a parent LDAP is replicating. This requires the parent LDAP ID be set to an LDAP to which replication is not already being performed.
15. parent_ldap_id – LDAP that has user and group information that is being replicated to another LDAP.

Once parameters have been entered for an LDAP directory server, the user must then create the necessary views for the LDAP. Each LDAP registered within the user management application must have a User View, Group View, and User Group Junction View (labelled *_user, *_group, and *_junc_user_group) created for it. These views are used for querying data from the LDAP directory while operations are being run against the LDAP. These views also are used within the user management application for display purposes.

Depending upon the LDAP type being used (i.e., AD, ADLDS, or OID), a function can be used to create each view. Following are examples of the query used for each of the following three views:

User View

```
create or replace view oss_ad_user as
    select
      user_id
      ,cn
      ,is_enabled
      ,sn
      ,givenname
      ,mail
      ,rdn
      ,parentdn
      ,state
      ,access_list as state_access
      ,create_date
      ,last_login
      ,is_locked
      ,pwd_last_set
    from
      table
        (
          select cast (get_ad_users (LDAP_SERVER_ID) as ldap_user_table)
          from dual
        )
    where user_id is not null;
```

Group View

```
create or replace view oss_ad_group as
    select
      group_name
      ,rdn
      ,parentdn
    from
      table (select cast(get_ad_groups(LDAP_SERVER_ID) as ldap_group_table) from dual)
    where
      group_name is not null;
```

User Group Junction View

```
create or replace view oss_ad_junc_user_group as
    select
      group_name,
      user_name
    from
      table (select cast(get_ad_junctions(LDAP_SERVER_ID) as ldap_junc_table) from
dual)
    where
      group_name is not null;
```

In the aforementioned sample views, several functions are utilized to retrieve results from the LDAPs using specifically tailored functions for each type of LDAP.

3.2.2 APPLICATION_REGISTRY

The APPLICATION_REGISTRY table contains a list of applications and the corresponding LDAP that should be used for authentication and authorization. This information includes the following:

1. application_registry_id – Unique identifier (ID) for the record.
2. application_id – Oracle APEX or other identified ID representing the application when making calls to the top level API.
3. application_name – Text name of the application to display within the application.
4. ldap_server_id – Foreign key to the LDAP_SERVER_CONFIG table used to identify which LDAP the application should utilize for authentication and authorization.

3.3 API

Using native dynamic PL/SQL capabilities and the configuration tables discussed in Sections 3.2.1 and 3.2.2, CISA researchers developed a common top-level API with PL/SQL packages to which applications can make calls. The generic application API functions using a specific application ID that is tied to an LDAP via the APPLICATION_REGISTRY table. A package also was developed for the user management application API so that it could function using a specific LDAP ID instead of an application ID. These packages are described in Sections 3.3.1 through 3.3.4.

3.3.1 OPS_UM_LDAP

OPS_UM_LDAP is the base package utilized for retrieving LDAP metadata, as well as for establishing connections to an LDAP. This package provides the following procedures:

1. get_admin_session – Establishes an administrative LDAP session for manipulating the LDAP or retrieving user or group information for the given application.
2. get_user_session – Establishes an LDAP session as a specific user for a given application. Can be used to authenticate a user as well.
3. close_session – Closes a given LDAP session.
4. get_ldap_server – Retrieves the LDAP Server ID associated with a given Application ID.
5. get_ldap_host – Retrieves the LDAP host for a given LDAP ID.
6. get_ldap_port – Retrieves the LDAP port for a given LDAP ID.
7. get_ldap_auth_base – Retrieves the authentication path for the administrative user for a given LDAP ID.
8. get_ldap_full_base – Retrieves the root path for a given LDAP ID.
9. get_ldap_user_base – Retrieves the path to the users for a given LDAP ID.
10. get_ldap_group_base – Retrieves the path to the groups for a given LDAP ID.
11. get_ldap_package_prefix – Retrieves the package prefix used to determine which implementation to execute for a given LDAP ID.

12. get_ldap_view_prefix – Retrieves the view prefix used to query users and groups for a given LDAP ID.
13. get_real_login – Retrieves the exact login name for each user in cases where the common name (CN) differs from the user's login name, such as when the user CN returned is not the ID utilized to manipulate the user in Microsoft AD.

3.3.2 OPS_UM_LDAP_GROUP

The OPS_UM_LDAP_GROUP package is used to administer groups within an LDAP based on interaction with a specific application ID that is listed in the APPLICATION_REGISTRY table. This package provides the following procedures:

1. add_user_to_group – Adds the given user to the given group for a given application.
2. remove_user_from_group – Removes the given user from the given group for a given application.
3. create_group – Creates a new group with the given name and description for a given application.
4. delete_group – Removes a group identified by the passed-in string.

3.3.3 OPS_UM_LDAP_USER

The OPS_UM_LDAP_USER package is used to administer users within an LDAP, based upon interaction with a specific application ID that is listed in the APPLICATION_REGISTRY table. This package provides the following procedures:

1. authenticate_user – Attempts to authenticate the given user with the given password against the given application.
2. get_user_attribute – Retrieves the specified user attribute for a given user from the given application.
3. get_user_email – Retrieves the mail attribute for the given user from the given application.
4. create_user – Creates a user with the specified attributes for a given application.
5. reset_user_password – Uses a password generator to create a random password and resets the user's account attributes as necessary for the given application.
6. change_user_password – Allows users to change their password for a given application.
7. modify_user – Sets the given attribute to the given value for a given user for a given application. There are overloaded versions of this procedure for text and numeric attributes.
8. clear_user_attribute – Resets/clears a given attribute for a given user for a given application.
9. enable_user – Enables a user for the given application.
10. disable_user – Disables a user for the given application.
11. unlock_user – Unlocks a user for the given application.
12. check_group_membership – Checks to see if the given user is a member of the given group for the given application.
13. is_locked – Indicates if the given user account is locked for the given application.

14. is_password_expired – Indicates if the given user account has an expired password for the given application.
15. get_password_expiration_days – Retrieves the number of days until the given user account password expires for the given application.

3.3.4 OUMA_FACADE

The OUMA_FACADE package is utilized by the user management application as its interface to an LDAP directory by specifying an LDAP ID instead of an application ID.

1. get_admin_session – Establishes an administrative session for manipulating the given LDAP.
2. add_user_to_group – Adds a given user to the given group in the given LDAP.
3. remove_user_from_group – Removes the given user from the given group in the given LDAP.
4. create_group – Creates a group with the given name and description in the given LDAP.
5. delete_group – Deletes the given group from the given LDAP.
6. authenticate_user – Attempts to authenticate the given user against the given LDAP.
7. get_user_attribute – Retrieves the specified attribute for the given user from the given LDAP.
8. get_user_email – Retrieves the mail attribute for the given user from the given LDAP.
9. get_user_attrs – Retrieves a set of attributes for the given user from the given LDAP.
10. does_user_exist – Indicates if the given user exists in the given LDAP.
11. create_user – Creates a user with the specified information in the given LDAP.
12. reset_user_password – Resets a user password, and sends the user an email with the “forgot password” link for the application menu associated with the given LDAP.
13. change_user_password – Changes a user’s password for the given LDAP.
14. modify_user – Updates the given attribute with the given value for the given user in the given LDAP.
15. clear_user_attribute – Resets/clears the given attribute for the given user in the given LDAP.
16. enable_user – Enables the given user in the given LDAP.
17. disable_user – Disables the given user in the given LDAP.
18. check_group_membership – Indicates if the given user is a member of the given group in the given LDAP.
19. unlock_user – Unlocks the given user account in the given LDAP.
20. is_locked – Indicates if the given user’s account is locked in the given LDAP.
21. is_enabled – Indicates if the given user’s account is enabled in the given LDAP.
22. is_password_expired – Indicates if the given user’s account has an expired password in the given LDAP.
23. delete_user – Deletes the given user account from the given LDAP.

3.4 LDAP–SPECIFIC PACKAGES

Below the top-level API, packages that are specific to each LDAP directory technology supported are provided. CISA researchers utilized the object-oriented, polymorphic-like behavior of dynamic PL/SQL, along with the configuration data stored in the database, to call these specific LDAP implementations. For each LDAP implementation, three packages (i.e., user, group, and session management packages) are provided. Each of these packages must implement the top-level interface defined by the OPS_UM_LDAP_* API packages.

Currently, CISA researchers have implemented packages to support the following technologies:

- Microsoft Active Directory
- Microsoft Active Directory Lightweight Directory Services
- Oracle Internet Directory
- Weblogic Lightweight Directory Access Protocol
- Amazon Simple Active Directory

Each of these implementations can be utilized by entering configuration data into the LDAP_SERVER_CONFIG data.

3.5 UTILIZING THE API IN THE USER MANAGEMENT APPLICATION

The user management application was developed in Oracle's Application Express (APEX) platform and was built on the OUMA_FACADE API. Leveraging APEX, CISA researchers were able to tightly integrate the application to the API. The interface provides user management functionalities, such as creating users and groups, modifying existing users, managing group membership, and an administrative area. The administrative area is used to view logs and to manage application and LDAP configuration data.

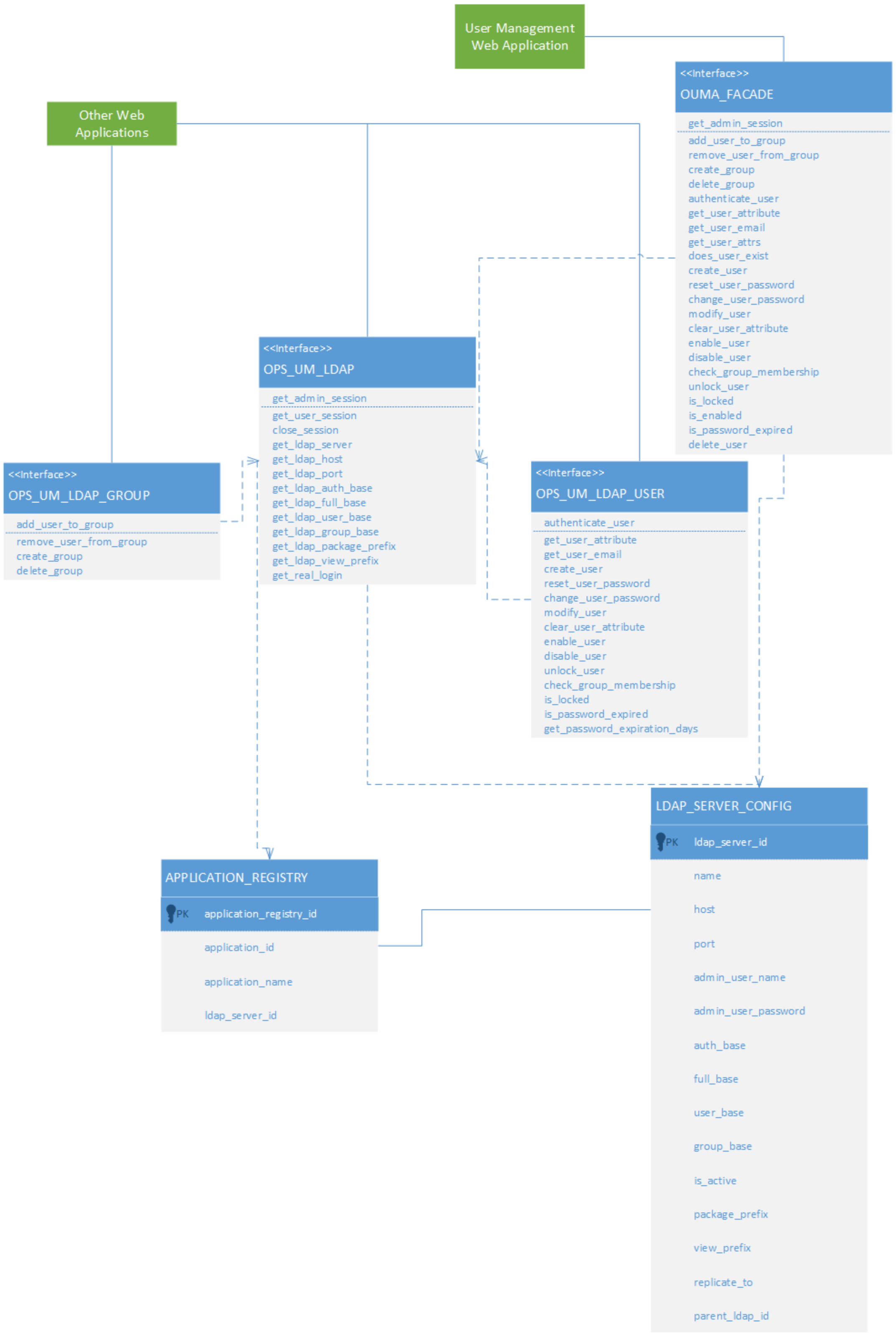
The key component to making this a flexible application is an LDAP selection utility that allows the application to manage the selected LDAP simply by applying the selected LDAP parameters to the application's metadata. With this capability, application users can efficiently maneuver between the LDAPs required by the client system.

The inclusion of the management utilities to the application also allows a user with administrative privileges to add additional LDAP servers to the configuration. If the client requirements change, or if a new technology is utilized, the additional LDAP servers can be added and managed by this single application.

4. CONCLUSION

The Oracle PL/SQL programming language provides a flexible platform for developing core application logic that CISA researchers were able to utilize to create an application used by the client for more than 10 years. Over the years, CISA has added new LDAP directory configurations to the production environment as client requirements and resources have changed. The dynamic nature of the user management system allowed for easy setup and utilization of any new LDAP directories, minimizing the time and effort expended on system changes.

APPENDIX A
MULTI–LDAP USER MANAGEMENT TOOL OVERVIEW


User Management Web Application
Other Web Applications
<<Interface>>
OUMA_FACADE
get_admin_session
add_user_to_group
remove_user_from_group
create_group
delete_group
authenticate_user
get_user_attribute
get_user_email
get_user_attrs
does_user_exist
create_user
reset_user_password
change_user_password
modify_user
clear_user_attribute
enable_user
disable_user
check_group_membership
unlock_user
is_locked
is_enabled
is_password_expired
delete_user
<<Interface>>
OPS_UM_LDAP
get_admin_session
get_user_session
close_session
get_ldap_server
get_ldap_host
get_ldap_port
get_ldap_auth_base
get_ldap_full_base
get_ldap_user_base
get_ldap_group_base
get_ldap_package_prefix
get_ldap_view_prefix
get_real_login
<<Interface>>
OPS_UM_LDAP_GROUP
add_user_to_group
remove_user_from_group
create_group
delete_group
<<Interface>>
OPS_UM_LDAP_USER
authenticate_user
get_user_attribute
get_user_email
create_user
reset_user_password
change_user_password
modify_user
clear_user_attribute
enable_user
disable_user
unlock_user
check_group_membership
is_locked
is_password_expired
get_password_expiration_days
APPLICATION_REGISTRY
PK application_registry_id
application_id
application_name
ldap_server_id
LDAP_SERVER_CONFIG
PK ldap_server_id
name
host
port
admin_user_name
admin_user_password
auth_base
full_base
user_base
group_base
is_active
package_prefix
view_prefix
replicate_to
parent_ldap_id