



Patron Directory Services Guide

Versions 1.3 & 2.x

CONFIDENTIAL INFORMATION

The information herein is the property of Ex Libris Ltd. or its affiliates and any misuse or abuse will result in economic loss. DO NOT COPY UNLESS YOU HAVE BEEN GIVEN SPECIFIC WRITTEN AUTHORIZATION FROM EX LIBRIS LTD.

This document is provided for limited and restricted purposes in accordance with a binding contract with Ex Libris Ltd. or an affiliate. The information herein includes trade secrets and is confidential.

DISCLAIMER

The information in this document will be subject to periodic change and updating. Please confirm that you have the most current documentation. There are no warranties of any kind, express or implied, provided in this documentation, other than those expressly agreed upon in the applicable Ex Libris contract. This information is provided AS IS. Unless otherwise agreed, Ex Libris shall not be liable for any damages for use of this document, including, without limitation, consequential, punitive, indirect or direct damages.

Any references in this document to third-party material (including third-party Web sites) are provided for convenience only and do not in any manner serve as an endorsement of that third-party material or those Web sites. The third-party materials are not part of the materials for this Ex Libris product and Ex Libris has no liability for such materials.

TRADEMARKS

"Ex Libris," the Ex Libris bridge, Primo, Alma, Aleph, Alephino, Voyager, SFX, MetaLib, Verde, DigiTool, Preservation, Leganto, Esploro, URM, Voyager, ENCompass, Endeavor eZConnect, WebVoyage, Citation Server, LinkFinder and LinkFinder Plus, and other marks are trademarks or registered trademarks of Ex Libris Ltd. or its affiliates.

The absence of a name or logo in this list does not constitute a waiver of any and all intellectual property rights that Ex Libris Ltd. or its affiliates have established in any of its products, features, or service names or logos.

Trademarks of various third-party products, which may include the following, are referenced in this documentation. Ex Libris does not claim any rights in these trademarks. Use of these marks does not imply endorsement by Ex Libris of these third-party products, or endorsement by these third parties of Ex Libris products.

Oracle is a registered trademark of Oracle Corporation.

UNIX is a registered trademark in the United States and other countries, licensed exclusively through X/Open Company Ltd.

Microsoft, the Microsoft logo, MS, MS-DOS, Microsoft PowerPoint, Visual Basic, Visual C++, Win32, Microsoft Windows, the Windows logo, Microsoft Notepad, Microsoft Windows Explorer, Microsoft Internet Explorer, and Windows NT are registered trademarks and ActiveX is a trademark of the Microsoft Corporation in the United States and/or other countries.

Unicode and the Unicode logo are registered trademarks of Unicode, Inc.

Google is a registered trademark of Google, Inc.

Copyright Ex Libris Limited, 2018. All rights reserved.

Document released: October 2018

Web address: <http://www.exlibrisgroup.com>

Table of Contents

Part I Introduction

Chapter 1	Overview of This Guide and the PDS	11
	About This Guide	11
	Overview of the PDS.....	12
	<i>Institutions</i>	14
	<i>Securing the PDS</i>	15
	<i>Debugging the PDS</i>	15
	<i>Working with SaaS</i>	15

Part II SaaS Support

Chapter 2	Working in a SaaS Environment.....	19
	PDSDefinitions File	19
	Customer Institution Identification and Configuration.....	19

Part III Institution and Calling Application Configuration

Chapter 3	Institution Configuration	25
	Overview of Institutions.....	25
	Configuring Institutions Using the PDS Configuration Wizard	26
	Configuring Institutions Manually	30
	<i>Adding Institutions</i>	32
	<i>Configuring an Institution Display Name</i>	32
	<i>Configuring Character Conversion</i>	33
	<i>Configuring the Display Order of Institutions</i>	34
	<i>Configuring an Override for the Default Institution</i>	34
Chapter 4	Calling Application Configuration	37
	MetaLib Configuration	37
	DigiTool Configuration	38
	Aleph Configuration	39

Part IV PDS Component Configuration

Chapter 5	Single Sign-On (SSO) Configuration	43
	Overview of SSO Configuration for Ex Libris Products	43

	Configuring SSO Using the PDS Configuration Wizard	45
	Configuring SSO Manually	47
	<i>Configuring Single Sign-On – [LOGON] Section</i>	47
	<i>Configuring Single Sign-Off – [LOGOUT] Section</i>	48
	<i>Callback URL Security – [ALLOWED_HOSTS] Section</i>	48
	<i>Persistent Cookies – [PERSISTENT_COOKIE] Section</i>	49
	Synchronizing Subdomains for SSO	50
	Remote SSO	51
	Configuring Remote SSO	52
	<i>Configuring Remote SSO Using the PDS Configuration Wizard</i>	52
	<i>Configuring Remote SSO Manually</i>	53
	Configuring a Remote PDS SSO Check.....	57
Chapter 6	Login Configuration.....	59
	Overview of Login Page Handling	59
	Configuring Remote Login	61
	<i>Configuring Remote Login Using the PDS Configuration Wizard</i>	62
	<i>Configuring Remote Login Manually</i>	63
	Configuring Local Login	67
	<i>Configuring Authentication Methods Using the PDS Configuration Wizard</i>	68
	<i>Configuring Authentication Methods Manually</i>	76
	<i>LDAP Services – Manual Configuration</i>	81
	<i>Remote CGI Hook – Manual Configuration</i>	86
	Local Login Page Display	89
	<i>Customizing the Institution Login Page</i>	93
	<i>Language-Specific Login Pages</i>	93
	<i>Checking the Displayed Login Page File</i>	94
Chapter 7	User Attribute Retrieval and Attribute Mapping.....	97
	Overview of User Attribute Retrieval	97
	Configuring User Attribute Retrieval Methods Using the PDS Configuration Wizard.....	98
	<i>Testing User Attribute Retrieval Methods</i>	101
	Configuring User Attributes Manually	101
	Overview of Attribute Mapping	102
	Mapping User Attributes Using the PDS Configuration Wizard ...	102

	Mapping User Attributes Manually	105
	User Attribute Retrieval – Requests by Calling Applications.....	107
	User Attribute Retrieval with Voyager	108
Chapter 8	Logout Configuration.....	111
	Overview of Logout Configuration	111
	Configuring Logout Using the PDS Configuration Wizard.....	112
	Configuring Logout Manually.....	113
	<i>iChain Logout</i>	114
	<i>CAS Logout</i>	114
	<i>Shibboleth Logout</i>	115

Part V Remote Authentication Programs

Chapter 9	CAS and iChain Authentication	119
	CAS Authentication	119
	<i>EZproxy Authentication</i>	124
	iChain Authentication.....	125
Chapter 10	Shibboleth	129
	Overview of Shibboleth	129
	Prerequisites for Working with Shibboleth	130
	<i>Tips for Installing the Shibboleth Service Provider</i>	130
	<i>Verifying That Prerequisites Are Met</i>	131
	Additional Shibboleth Apache Configuration	131
	<i>Adding a Symbolic Link in the /apache/htdocs Directory</i>	132
	<i>Editing the Apache Configuration File</i>	132
	<i>Modifying <application name>_start</i>	133
	Configuring Shibboleth to Work with the PDS.....	133
	<i>Configuring Shibboleth Using the Configuration Wizard</i>	134
	<i>Configuring Shibboleth Manually</i>	137
	The Shibboleth Login Workflow	139
	The Shibboleth Logout Workflow	140
	The Shibboleth SSO Workflow	141
	Configuring Shibboleth as a WAYF.....	141

Part VI Security Configuration

Chapter 11	Configuring Security for the PDS	145
	Configuring the PDS to Use SSL	145
	Securing the Configuration Wizard.....	146
	Optional Security Enhancement for REMOTE_LOGIN and REMOTE_SSO.....	146
	X-Server Security Patch	148
	Securing the PDS_HANDLE Cookie	149

Part VII Debugging

Chapter 12	Debugging the PDS.....	153
	PDS Direct Access.....	153
	<i>The Remote and Local List Page</i>	<i>154</i>
	<i>Main Menu – Logged Out Page</i>	<i>154</i>
	<i>Main Menu – Logged On Page</i>	<i>156</i>
	Error Messages from the PDS.....	156
	PDS Debug Mode	157

Part VIII Appendixes

Appendix A	Utilities for Manual Configuration.....	161
	PDS Setup Validity Testing Utility	161
	<i>Explanation of the PDS Check Utility Options</i>	<i>162</i>
	Encrypt Password Utility	163
Appendix B	tab_service.<institute> Configurations.....	167
	Aleph Standard Configuration.....	168
	MetaLib Standard Configuration.....	169
	DigiTool Standard Configuration	169
	Alma Standard Configuration	170
	Voyager Standard Configuration.....	171
	LDAP Configuration	171
	Aleph X-Server Configuration.....	172
	Remote CGI Configuration	173
Appendix C	Ex Libris Calling Application Target Attributes	175
	Overview	175
	MetaLib Target Attributes.....	175

	Primo Target Attributes.....	176
	DigiTool Target Attributes.....	176
Appendix D	The bor_info.tags Attribute Mapping Table.....	179
	Overview.....	179
	Aleph Attributes.....	180
	Voyager Attributes.....	181
	DigiTool Attributes.....	182
	MetaLib Attributes.....	182
Appendix E	Example of the PDSDefinitions File.....	187
Index	189

Part I

Introduction

This part contains the following section:

- **Section 1: Overview of This Guide and the PDS** on page 11

1

Overview of This Guide and the PDS

This section includes:

- **About This Guide** on page 11
- **Overview of the PDS** on page 12

About This Guide

This guide describes the interaction between the following Ex Libris products and the Patron Directory Service (PDS), a component that facilitates user authentication and login to the product (or **calling application**, as it is referred to in this document):

- Aleph
- MetaLib
- DigiTool
- Primo
- Voyager

NOTES:

This guide describes how to work with PDS versions 1.3, 2.0, and 2.1. For information about the PDS 1.3 and 2.x topologies, refer to the *Patron Directory Services Upgrade Guide* (also located under Cross-Product > Technical Documentation > PDS in the Documentation Center).

- PDS 2.1 can be used with the following Ex Libris applications: Primo 3.1.1 and later; Aleph version 21.1 and later; MetaLib version 4.4.3 and later; Voyager version 8.1 and later.
- PDS 2.0 can be used with the following Ex Libris applications: Primo version 3.0.2 and later; Aleph version 20.2.2 and later; MetaLib version 4.4 and later; DigiTool version 3.3 and later; Voyager version 8.0 and later

- PDS 1.3 can be used with the following Ex Libris applications: all installations of Primo; MetaLib versions 3 and 4; Aleph versions 18.01 and later
 - Earlier versions of Aleph and DigiTool can be configured to use PDS 1.2. Note that there may be certain features described in this document that are not available for PDS 1.2.
-

The guide provides an in-depth technical understanding of the PDS and how it can be configured to suit different authentication needs. Given its technical content, the primary audience for this guide is likely to be system administrators or staff with technical knowledge of authentication arrangements at the institution or consortium.

NOTE:

The screen capture examples in the document use the Ex Libris PDS pages. MetaLib, DigiTool, and Aleph all have their own customized pages that contain their logos.

Overview of the PDS

The PDS is a “back-end” Web component that facilitates user authentication and login to an Ex Libris application. The PDS is part of the standard application package for each of the Ex Libris products listed above, but it is a distinct and separate component.

The following diagram illustrates the components of the PDS system:

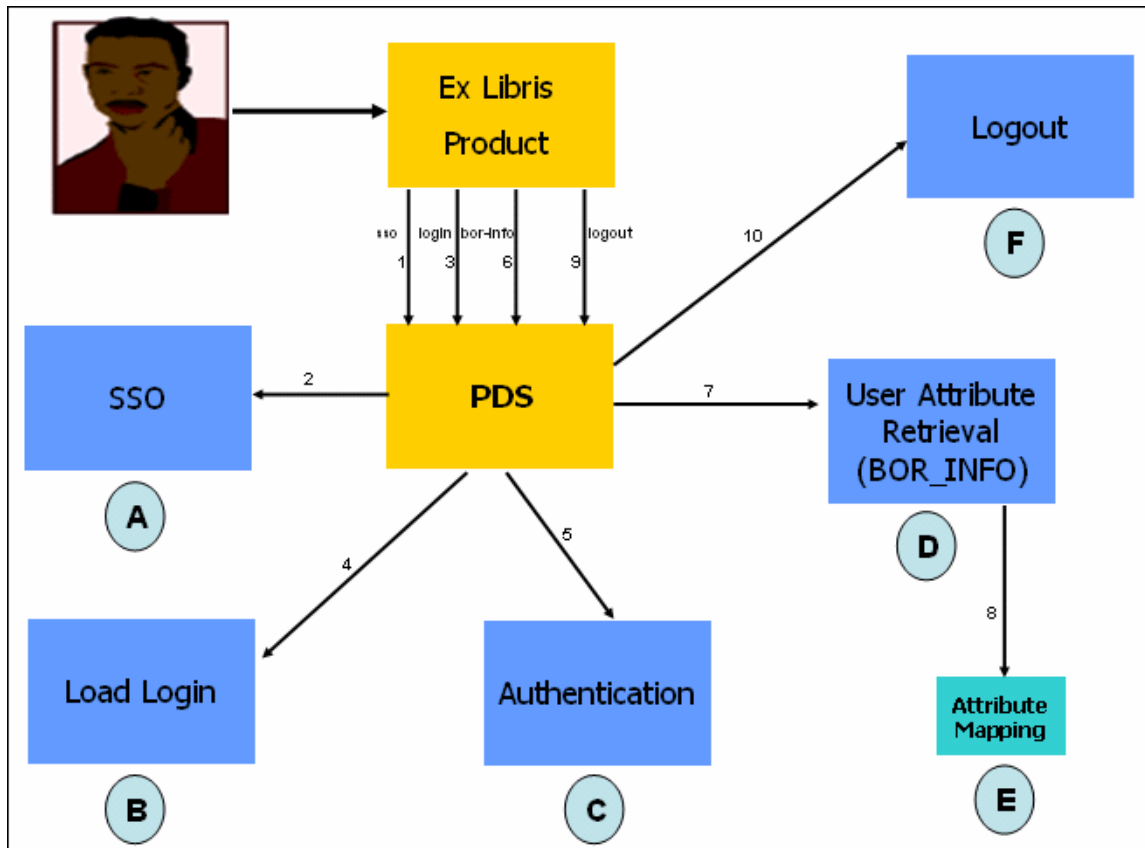


Figure 1: PDS System Components

The PDS system operates in the following manner:

- 1 A user starts a new session in an Ex Libris calling application.
- 2 If the calling application sends a Single Sign-On (SSO) request to the PDS, the PDS checks whether SSO services are configured between this calling application and other Ex Libris calling applications (step A in the above figure). If SSO or remote SSO is configured, the PDS attempts to authenticate the user this way. If the user is authenticated, the calling application proceeds to request user attributes (see step 4 below). For information on configuring SSO or remote SSO, see [Single Sign-On \(SSO\) Configuration](#) on page 43.
- 3 If SSO is not configured or failed to authenticate a user and the user attempts to log in to the calling application, a load-login request is sent from the calling application to the PDS (step B in the above figure). The PDS then attempts to authenticate the user via a remote authentication system (such as CAS, iChain, or Shibboleth) or locally (step C in the above figure), via the calling application's own user database, an LDAP server, or a custom

remote CGI hook. For information on configuring local or remote login, see [Login Configuration](#) on page 59.

- 4 Following authentication, the calling application requests user attributes (step **D** in the above figure). The PDS obtains user attributes either from the calling application's user database or from external user directories. Attributes retrieved are handled in an XML format, mapped, and normalized (step **E** in the above figure). For information on configuring user attribute retrieval methods and mapping attributes, see [User Attribute Retrieval and Attribute Mapping](#) on page 97.

NOTE:

The PDS can accommodate an authentication check against one server and the retrieval of user attributes from another.

- 5 The PDS sends the normalized user attributes to the calling application and the user is logged in to the application.
- 6 When the user attempts to log out of the application, a logout request is sent from the calling application to the PDS. The logout process (step **F** in the above figure) expires the user's session. If you want to direct the user to a specific URL after logout or log the user out of a remote authentication server, you can configure these logout options as described in [Logout Configuration](#) on page 111.

IMPORTANT:

To work with an Ex Libris calling application in conjunction with the PDS, you must first configure the calling application. For information on configuring the Ex Libris calling applications, see [Calling Application Configuration](#) on page 37.

Institutions

A key PDS concept is the **institution** or **institute** (used interchangeably in this document). In a multi-institution site (multiple Primo or MetaLib institutions, multiple Aleph ADMs, or multiple Voyager databases), each institution usually has its own PDS configuration, referred to as a PDS **institution/institute**. For information on configuring institutions, see [Institution Configuration](#) on page 25.

NOTES:

- The PDS can be configured to work for a consortium in which several institutions share the same Ex Libris application, but each has its own authentication and user database.
 - For a SaaS product, you cannot create or delete institutions. You can only edit existing institutions that were created for you by Ex Libris.
-

Securing the PDS

You can choose to work with the PDS within a secure environment. For information on configuring security for the PDS, see [Configuring Security for the PDS](#) on page 145.

Debugging the PDS

For information on debugging the PDS system, see [Debugging the PDS](#) on page 153.

Working with SaaS

For version 2.1, the PDS was enhanced to work in a SaaS environment. For information on SaaS support, see [Working in a SaaS Environment](#) on page 19.

Part II

SaaS Support

This part contains the following section:

- **Section 2: Working in a SaaS Environment** on page 19

2

Working in a SaaS Environment

This section includes:

- **PDSDefinitions File** on page 19
- **Customer Institution Identification and Configuration** on page 19

PDSDefinitions File

To support working with the PDS in a SaaS environment, the following parameter must be added to the PDSDefinitions file:

```
our ($saas) = "Y";
```

When this parameter is activated (that is, it is set to Y), the PDS uses a separate session for each institution and users can therefore log in to several institutions in parallel. Note that in order for this to work, the calling application must send the institution name to the PDS for each operation performed.

Alternatively, you can use the `pds_saas_on` and `pds_saas_off` scripts to facilitate the use of the `saas` parameter. To activate these scripts, run the following:

```
pdsroot
cd program
./pds_saas_on
./pds_saas_off
```

Customer Institution Identification and Configuration

To enable working with the PDS in a SaaS environment, Ex Libris has defined the following parameters: `customerid` and `institutionid`. The values of these

parameters are mapped to the value of the `institute` parameter received from the calling application. If not found, the default values of `customerid` and `institutionid` are 0 (zero).

NOTE:

All requests to the PDS, including those in a SaaS environment, should use only the `inst_code` value for the `institute` parameter. For example, `University_A`.

When working in a SaaS environment, the PDS operates as follows:

- When accessing the login page, customers are identified by the `institute` parameter. (The `load-login` request to the PDS contains the `institute` parameter to identify the customer.) The PDS compares this parameter with the `customerid` parameter values in the configuration and displays only the institutions that belong to the customer on the login page. If the `institute` parameter is missing, the PDS displays an error page.
- The PDS was enhanced to support customer partitioning in user attribute and SSO requests. In these requests, the PDS looks for the session record (local or remote) using the `pds-handle` parameter. If the SaaS mode is activated (the `saas` parameter is set to `Y` in the `PDSDefinitions` file), the `institute` parameter is also received. If the `customerid` of the current `pds-handle` session is the same as the `institute` parameter value received from the calling application, the PDS returns an XML with user attributes. If the calling application's `institute` and `customerid` values are different, the request fails.
- The PDS configuration wizard was enhanced to support the SaaS mode of operation by displaying only the relevant institutions for the customer. Based on the `institute` parameter that should be received by the PDS in SaaS mode, the PDS locates the `customerid` and displays only the relevant institutions.
- When working in SaaS mode, the PDS wizard works at two possible levels: `super_user` and `institution`.

At the `super_user` level, all institutions are open for creation, deletion, and editing.

At the `institution` level, only the relevant institution is open for editing. Deletion/creation is not allowed.
- Calling the PDS Wizard should be handled as follows:
 - `super_user` level: `http://<PDS_URL>/pdsadmin/general_configuration.cgi?level=super_user`
 - `institution` level: `http://<PDS_URL>/pdsadmin/general_configuration.cgi?institute=REEF&level=institution`

- If both institution and super_user are sent ~V, the super_user level is activated: `http://<PDS_URL>/pdsadmin/general_configuration.cgi?institute=DEMO&level=super_user`

NOTE:

The customer can edit only an existing institution. The create and delete options are disabled.

- The `pds_create_inst` script was added to the PDS to facilitate the addition of a new customer and institution for SaaS-based applications.

Run the following to activate the script:

```
pdsroot
cd program
pds_create_inst <application> <customerid> <institutionid>
<inst_code> <domain for authentication> <port for
authentication> <secure [Y/N]>
```

For example:

```
pds_create_inst   urm 32 11
UNIV1univ1.prod.saas.exlibrisgroup.com 1801 N
```


Part III

Institution and Calling Application Configuration

This part contains the following sections:

- **Section 3: Institution Configuration** on page 25
- **Section 4: Calling Application Configuration** on page 37

3

Institution Configuration

This section includes:

- **Overview of Institutions** on page 25
- **Configuring Institutions Using the PDS Configuration Wizard** on page 26
- **Configuring Institutions Manually** on page 30

Overview of Institutions

The **institution** is an administrative concept and denotes the organization to which end users and/or resources belong. In terms of the PDS, the important aspect of an institution is user affiliation.

The following is a list of Ex Libris calling applications and an explanation of the way in which each one handles the concept of the institution or its equivalent:

- **Aleph** – In Aleph, each institution within a consortium generally has its own ADM library and in most cases, an ADM is the equivalent of a PDS institution. There are some cases in which one or more sublibraries represent a single institution.
- **MetaLib** – In MetaLib, each institution within a consortium generally has its own MetaLib institution, which is the equivalent of a PDS institution. There are some MetaLib consortia that share a single institution and refer to the individual institutions as MetaLib **user groups**.
- **Primo** – In Primo, each institution within a consortium generally has its own Primo institution, which is the equivalent of a PDS institution.
- **DigiTool** – DigiTool does not have the equivalent of an institution. Consortia can use the **user group** concept to indicate individual institutions.
- **Voyager** – In Voyager, each institution within a consortium generally has its own Voyager database, which is the equivalent of a PDS institution.

NOTES:

- You must define the institution or institution-equivalent in your calling application before you can configure the institution in the PDS.
 - For SaaS products, Ex Libris provides a new PDS configuration for each customer. This configuration contains the number of institutions required by the customer. The customer can edit the configuration of these institutions, but cannot create additional institutions or delete existing institutions.
-

To configure institutions:

- If you are working with the local disk PDS topology, you can configure institutions via the configuration wizard, using the instructions in **Configuring Institutions Using the PDS Configuration Wizard** below, or manually, using the instructions in **Configuring Institutions Manually** on page 30.
- If you are working with the shared Oracle database PDS topology, you must configure institutions via the configuration wizard, using the instructions in **Configuring Institutions Using the PDS Configuration Wizard** below.

Configuring Institutions Using the PDS Configuration Wizard

If you are working with the PDS configuration wizard, you configure institutions using the PDS - Institution Configuration page.

NOTE:

For SaaS products, you can edit the configuration of existing institutions, but cannot create additional institutions or delete existing institutions.

To configure an institution using the configuration wizard:

- 1 Click **Save and Continue** on the PDS - SSO Configuration page (the first page of the wizard, which you access from the Ex Libris calling application). The PDS - Institution Configuration page opens.

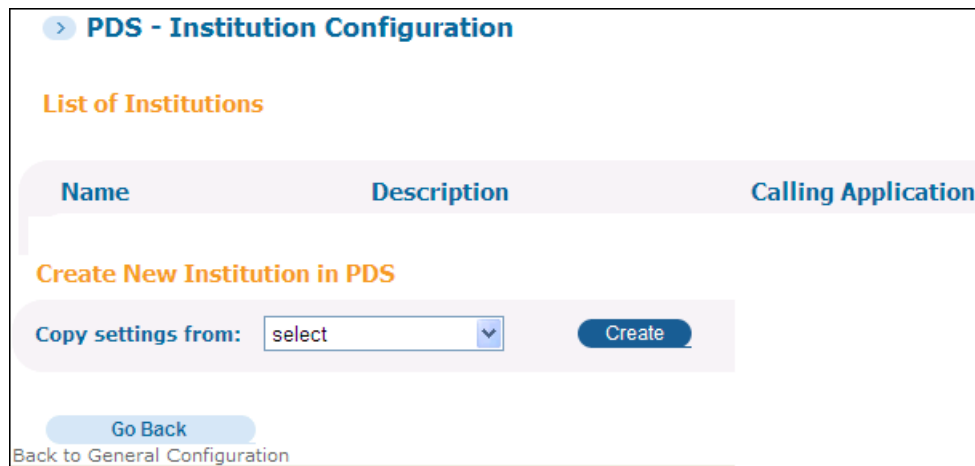


Figure 2: PDS - Institution Configuration Page

- 2 In the Create New Institution in PDS section, click **Create**. The PDS General Attributes page displays.

Figure 3: PDS General Attributes Page

3 Fill in the fields according to the following table:

Table 1. PDS General Attributes Page Details

Field Name	Description
Code (mandatory field)	The institution code.
Description (mandatory field)	A description of the institution name. This is how the institution name will appear in the PDS login menu.
Character conversion	Institution names can be displayed in many languages by using extended characters. When using non-Latin characters, you must add a character conversion definition. The PDS uses the appropriate character conversion routines to convert the local character set to UTF8.
Calling application and override code	Some calling applications (for example, Primo) require an override code for the institution. The override code defines the institution that is sent to the calling application instead of the institution code. This is useful in a consortium in the following situations: <ul style="list-style-type: none"> ■ Different authentication methods are needed for the same institution ■ There is an SSO scenario between two calling applications (for example, Aleph and MetaLib) in which a single configuration needs to return information for both applications (for example, both the MetaLib institution and the Aleph admin library).
Remote login	This field indicates whether the institution uses local login (Do not use remote login) or remote login (Use remote login). For information on defining remote login, see Configuring Remote Login on page 61.
Logout from PDS	For information on defining logout options, see Logout Configuration on page 111.
Application logout service credentials (PDS 2.x only)	For information on defining the application logout service credentials, see Configuring SSO Using the PDS Configuration Wizard on page 45 (end of section).
Remote SSO	For information on defining remote SSO, see Configuring Remote SSO Using the PDS Configuration Wizard on page 52.

Table 1. PDS General Attributes Page Details

Field Name	Description
EZproxy	For information on this option, see EZproxy Authentication on page 124.
Use LDAP attributes for Voyager BOR_INFO	For information on defining this option, see User Attribute Retrieval with Voyager on page 108.
Change institution display order	The default order of the institution names in the drop-down list of the PDS sign-on page is alphabetic. You may customize this by specifying which institution should be displayed first, second, and so forth. For details of this page, see Overview of Login Page Handling on page 59. NOTE: In most cases, the calling application forwards the user's institution (based on the IP address) to the PDS. In this case, the user's institution defaults to the first in the list.

- 4 Click **Save & Continue** to save your settings.

Configuring Institutions Manually

The `tab_service.<institute>` file in the `./pds/conf_table` directory defines the services required from the PDS for an institution. There is one file for each institution.

Each file is composed of sections and each section generally contains the following four lines, to which others can be added:

```
[SERVICE NAME]
program      =
params      =
[END]
```

The following is an explanation of each of the above lines.

Table 2. tab_service.<institute> File Description

File Line	Description
[SERVICE NAME]	<p>The service requested from the PDS. The following services are available:</p> <ul style="list-style-type: none"> ■ AUTHENTICATE – performs user authentication ■ BOR_INFO – gets user attributes (name, affiliation, and so forth) ■ REDIRECT_LOGOUT – redirects to a different page after logout ■ LOAD_LOGIN – redirects the user to a remote login page ■ REMOTE_LOGIN – handles the remote login response ■ LOAD_SSO – redirects the user to a remote SSO system ■ REMOTE_SSO – handles the remote SSO callback ■ BOR_VERIFICATION – gets the user password (used for EZPROXY SSO) ■ REMOTE_LOGOUT – redirects the browser to an SSO logout URL and then back to the regular PDS LOGOUT (CAS and SHIB) ■ LOGOFF_ID – provides credentials for single sign-off from Aleph/Digitool. ■ BOR_ID_FROM_LDAP – saves the user’s LDAP credentials for later use (used for Voyager). <hr/> <p>NOTE: Each of the above services is described in detail in the appropriate section of this guide.</p>
Program	<p>The program used. Specifies the appropriate PDS program to use for the task at hand. The programs reside in the <code>./pds/service_proc</code> directory.</p>
Params	<p>Parameters for the program or the name of a configuration file that is used to store parameters.</p>
[END]	<p>Added to each section to confirm the end of the service section.</p>

NOTE:

For examples of typical `tab_service.<institute>` file setups, see **Appendix B: `tab_service.<institute>` Configurations**.

This section describes:

- **Adding Institutions** on page 32
- **Configuring an Institution Display Name** on page 32
- **Configuring Character Conversion** on page 33
- **Configuring the Display Order of Institutions** on page 34
- **Configuring an Override for the Default Institution** on page 34

Adding Institutions

You can define new institutions by creating new `tab_service.<institute>` files.

To add an institution:

Open a new `tab_service.<institute>` file in the `./pds/conf_table` directory, using the following commands:

```
pdsroot
cd conf_table
vi tab_service.<institute>
```

NOTE:

The name of the institution is not case-sensitive—that is, `tab_service.<institute>` and `tab_service.<INSTITUTE>` are the same. However, the accepted convention is to use lowercase letters for the names of institutions.

Configuring an Institution Display Name

You can configure the names of the institutions in the drop-down list of the PDS login page.

To configure an institution's display name:

Modify the `desc` field in the `[INSTITUTE_DISPLAY]` section of the institution's `tab_service.<institute>` file.

For example:

```
[ INSTITUTE_DISPLAY ]
code      = CITYUNIV
lang      = ENG
desc      = City University
[ END ]
```

In this example, the PDS is configured to display **City University** as the institution name in the drop-down list of the login page. The drop-down list is sorted alphabetically, according to the values entered in the `desc` field of all `tab_service.<institute>` files.

NOTE:

You can display the institution name in only one language at a time.

Configuring Character Conversion

Institution names can be displayed in many languages using extended characters.

To configure character conversion:

When using non-Latin characters, add a character conversion definition below the string to be converted. The PDS will use the appropriate character conversion routines to convert the local character set to UTF8.

The following is an example of a character conversion definition and the corresponding page output:

```
[ INSTITUTE_DISPLAY ]
code      = HEB
desc      = מכללת אקטליבוריס
lang      = ENG
CHARACTER_CONVERSION = 8859_8_TO_UTF
[ END ]
```

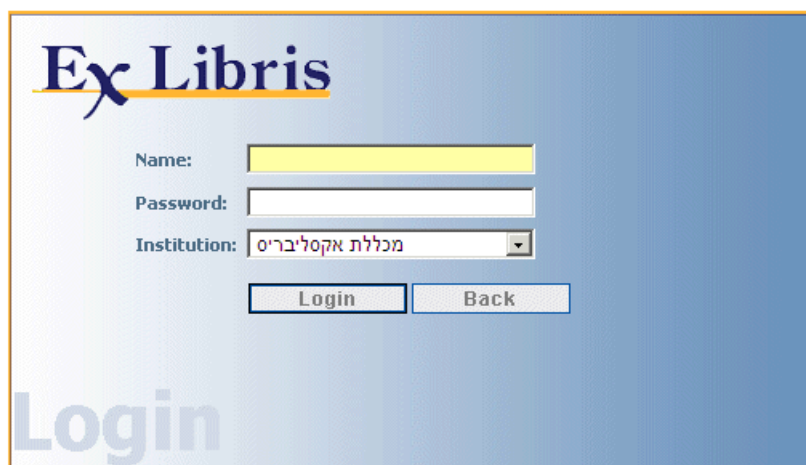


Figure 4: Example of an Institute Name in Hebrew

The character conversion uses `Unicode::MapUTF8::to_utf8` and supports the 8859-n family of character sets as input.

An alternative to using the character conversion is to code the strings in the `tab_service.<institute>` file in `&string;` or `&#Unicode;` HTML notation is then properly displayed without any conversion—for example, `desc = Åbo Akademi` displays as `?bo Akademi`.

Configuring the Display Order of Institutions

The default order of the institution names in the drop-down list is alphabetic.

To change the default display order:

Use the optional `sort_key = rank` parameter.

For example, to display **City University** first in the drop-down list box, add a `sort_key = 1` parameter to the `[INSTITUTE_DISPLAY]` section of the institution's `tab_service.<institute>` file:

```
[ INSTITUTE_DISPLAY ]
code      = CITYUNIV
lang      = ENG
desc      = City University
sort_key  = 1
[END]
```

Configuring an Override for the Default Institution

Usually there is a single `tab_service.<institute>` file corresponding to each calling application institution. By default, the calling application institution is the code from the `tab_service.<institute>` file. However, there is an

optional parameter to override the default institution. This is used in the following cases:

- A consortia in which different authentication methods are needed for the same institute.
- An SSO scenario between two calling applications, in which one `tab_service.<institute>` file must return both institutes (for example, an SSO scenario between MetaLib and Aleph, in which the `tab_service.<institute>` file must return both the MetaLib institute and the Aleph admin library).

To override the default institution:

Add a separate institution code for each calling application in the `[INSTITUTE_DISPLAY]` section of the institution's `tab_service.<institute>` file.

For example:

```
[INSTITUTE_DISPLAY]
code      = CITYUNIV
lang      = ENG
desc      = City University
aleph    = CTY50
metalib   = LAW
[END]
```

In this example, the `tab_service.cityuniv` returns `institute=LAW` to MetaLib, and `admin library=CTY50` to Aleph.

4

Calling Application Configuration

To work with an Ex Libris calling application in conjunction with the PDS, you must configure the calling application.

This section includes:

- **MetaLib Configuration** on page 37
- **DigiTool Configuration** on page 38
- **Aleph Configuration** on page 39

IMPORTANT:

For SSO configuration to work, the calling applications need to build a shared cookie. This means that the values of the `PDS_HOST` variable (mentioned in each section below) must consistently be either an IP address (for example, 10.1.235.47) or a host name (for example, RAM47), but not a combination of both.

NOTES:

- If you are working with Primo, refer to the *Primo Back Office Guide* for information on configuring Primo to call the PDS.
 - If you are working with Voyager, contact the Voyager installation team to configure the parameters that must be configured in the Voyager server XML in order for Voyager to call the PDS.
-

MetaLib Configuration

To work with MetaLib in conjunction with the PDS, you must configure MetaLib to call the PDS.

To configure Metalib to call the PDS:

- 1 Open the `metalib_start` file, located in the `$metalib_conf` directory.
- 2 Edit the `PDS_HOST` and `PDS_PORT` variables so that they point to the appropriate PDS instance.

The following is an example of the `PDS_HOST` variable:

```
setenv PDS_HOST 10.1.235.47
setenv PDS_HOST_IN 10.1.235.47
```

The following is an example of the `PDS_PORT` variable:

```
setenv PDS_PORT 8339
setenv PDS_PORT_IN 8339
```

- 3 Save the `metalib_start` file and restart the MetaLib servers (`start_w`).
- 4 Run `metalib_conf_create`.
- 5 Open the `www_server.conf` file, located in the `$metalib_conf` directory.
- 6 Ensure that the following lines exist:

```
#####
# PDS definitions:
#####
setenv server_pds "http://${PDS_HOST}:${PDS_PORT}/pds"
setenv server_pds_in "http://${PDS_HOST_IN}:${PDS_PORT_IN}/
pds"
```

NOTE:

If you are using SSL, replace HTTP with HTTPS.

DigiTool Configuration

To work with DigiTool in conjunction with the PDS, you must configure DigiTool to call the PDS.

To configure DigiTool to call the PDS:

- 1 Open the `dtl_start` file, located in the `$dtle_root` directory.
- 2 Edit the `PDS_HOST` and `PDS_PORT` variables so that they point to the appropriate PDS instance.

The following is an example of the way in which these variables should look:

```
setenv PDS_HOST 10.1.235.47
setenv PDS_PORT 8339
```

- 3 Open the `www_server.conf` file, located in the `$dtle_root` directory and locate the `server_pds` parameter. If it is hard-coded, edit the file so that it uses variables from the `dtl_start` file. The result is the following:

```
#####
# PDS definitions:
#####
setenv server_pds "http://${PDS_HOST}:${PDS_PORT}/pds"
```

NOTE:

If you are using SSL, replace HTTP with HTTPS.

- 4 Save the `dtl_start` file and run UTIL W/3/1 to start the servers.

Aleph Configuration

To work with Aleph in conjunction with the PDS, you must configure Aleph to call the PDS.

To configure Aleph to call the PDS:

- 1 Open the `tab100` table, located in the `$alephe_tab` directory.
- 2 Configure the `PDS-AWARE` and `PDS-KEY-TYPE` variables as follows:

```
PDS-AWARE=Y
PDS-KEY-TYPE=nn
```

Note that `nn` is determined according to the first two digits of the `z308_key_type`, which match the `ex1_id` value in the `z311.session_id` file, located in the `pds_files` directory. This is the key used by Aleph to access `BOR_INFO` attributes.

NOTE:

If you are working with LDAP, Shibboleth, or CAS, ensure that the system is properly configured to work with Aleph.

- 3 Open the `aleph_start` file, located in the `$alephe_root` directory.
- 4 Edit the `PDS_HOST` and `PDS_PORT` variables so that they point to the appropriate PDS instance.

The following is an example of the way in which these variables should look:

```
setenv PDS_HOST 10.1.235.47
setenv PDS_PORT 8339
```

- 5 Open the `www_server.conf` file, located in the `$alephe_tab` directory and locate the `server_pds` parameter. If it is hard-coded, edit the file so that it uses variables from the `aleph_start` file. The result is the following:

```
#####
# PDS definitions:
#####
setenv server_pds "http://${PDS_HOST}:${PDS_PORT}/pds"
```

NOTE:

If you are using SSL, replace HTTP with HTTPS.

- 6 Save the `aleph_start` file and run UTIL W/3/1 to start the servers.

Part IV

PDS Component Configuration

This part contains the following sections:

- **Section 5: Single Sign-On (SSO) Configuration** on page 43
- **Section 6: Login Configuration** on page 59
- **Section 7: User Attribute Retrieval and Attribute Mapping** on page 97
- **Section 8: Logout Configuration** on page 111

5

Single Sign-On (SSO) Configuration

This section includes:

- [Overview of SSO Configuration for Ex Libris Products](#) on page 43
- [Configuring SSO Using the PDS Configuration Wizard](#) on page 45
- [Configuring SSO Manually](#) on page 47
- [Synchronizing Subdomains for SSO](#) on page 50
- [Remote SSO](#) on page 51
- [Configuring Remote SSO](#) on page 52
- [Configuring a Remote PDS SSO Check](#) on page 57

Overview of SSO Configuration for Ex Libris Products

The PDS can be configured to provide Single Sign-On (SSO) services between the following Ex Libris products: MetaLib, DigiTool, Aleph, Primo, and Voyager. This chapter describes the configuration steps necessary to implement SSO.

When a user of one of the Ex Libris applications is authenticated via SSO, the PDS sets a cookie with a session key in the user's browser. The cookie is named PDS_HANDLE. This cookie is deleted from the user's browser when the user logs out of one of the applications, or when the user closes the browser.

NOTE:

Although the default cookie is a session cookie, the PDS can be configured to use a persistent cookie. For details, see the appropriate SSO configuration section below.

If you are working with the local disk PDS topology (see the *Patron Directory Services Upgrade Guide* for an explanation of the available PDS topologies), all the

Ex Libris calling applications share the same PDS by using the same PDS host name and PDS port number definitions. If you are working with the shared Oracle database PDS topology, each application uses its own PDS instance and each PDS instance writes the PDS cookie under its own subdomain (for example, <organization name>.edu.us). Generally, the subdomains are the same and each application's PDS can access the PDS cookies of the other applications. However, if the subdomains are not the same, SSO will not work unless you apply one of the solutions described in [Synchronizing Subdomains for SSO](#) on page 50.

You can also integrate your calling applications with external Single Sign-On servers using remote SSO. Remote SSO addresses scenarios in which users log in to a university's portal or another central login system and then move to a calling application. It is configured per institution, rather than per installation or application (as is the case with non-remote SSO). For more information regarding remote SSO and details of remote SSO configuration, see [Remote SSO](#) on page 51 and [Configuring Remote SSO](#) on page 52.

To configure SSO:

- If you are working with the local disk PDS topology, you can configure SSO via the configuration wizard, using the instructions in [Configuring SSO Using the PDS Configuration Wizard](#) on page 45, or manually, using the instructions in [Configuring SSO Manually](#) on page 47.
- If you are working with the shared Oracle database PDS topology, you must configure SSO via the configuration wizard, using the instructions in [Configuring SSO Using the PDS Configuration Wizard](#) on page 45.

NOTES:

- If you are working with the local disk PDS topology and setting up SSO between different IP addresses, the `hosts-allow` table in the Apache `conf` directory must be updated (see [X-Server Security Patch](#) on page 148).
 - If you are working with an Ex Libris SaaS product and a local product (for example, a local Primo installation and Alma), each product must be aware of the other's location. You must therefore define the PDS URL for each product as described in [Configuring a Remote PDS SSO Check](#) on page 57.
-

Configuring SSO Using the PDS Configuration Wizard

This section describes how to use the PDS configuration wizard to configure SSO.

To configure SSO using the configuration wizard:

- 1 On the first page of the configuration wizard, the PDS - SSO Configuration page, select the **Enabled** check box next to each of the calling applications that you want to enable for SSO.

Enabled	Calling Application	Default Institution
<input checked="" type="checkbox"/>	MetaLib	<input type="text"/>
<input checked="" type="checkbox"/>	Aleph	<input type="text"/>
<input checked="" type="checkbox"/>	DigiTool	<input type="text"/>
<input checked="" type="checkbox"/>	Primo	<input type="text"/>
<input checked="" type="checkbox"/>	Rosetta	<input type="text"/>
<input type="checkbox"/>	Voyager	<input type="text"/>
<input type="checkbox"/>	Shibboleth	<input type="text"/>

Persistent Cookie Configuration

Remember Me	Persistent Cookie Duration
<input checked="" type="checkbox"/>	<input type="text" value="30"/>

Manage Allowed Domains/IPs for Callback

Allow only the following domains/IPs for callback after PDS operations:

Full Domain Cookie

Use full domain for PDS session cookie

[Cancel & Go Back](#)
To Home Page

[Save & Continue](#)
To Institution Configuration

Figure 5: PDS - SSO Configuration Page

NOTE:

For information on the **Default Institution** field, see the first step of the procedure in **Configuring Remote SSO Using the PDS Configuration Wizard** on page 52.

- 2 In the Persistent Cookie Configuration section, define the following if you want the PDS to use a persistent cookie instead of a session cookie:
 - **Remember me**—Ensure that this option is selected (the default) if you want the **Remember me** option to display on the PDS login page. If the user selects the **Remember me** option, the PDS will create a persistent cookie.

The following section is added to the default login page and should be added to any locally modified pages:

```
<START_SECTION><!-- login-41 -->
  <SPAN class=msg><br><br>
</SPAN>
<input type="checkbox" name="term" value="long" checked>
  <SPAN class=msg>Remember me on this computer?
</SPAN>
<END_SECTION><! end change #F040b selfreg -- login-41 -->
```

NOTE:

The above is relevant only for local login. If you use remote login, you will need to add it to your remote login page.

- **Persistent Cookie Duration**—Indicates the lifespan of the persistent cookie. The default is 30 days.
- 3 In the Manage Allowed Domains/IPs for Callback section, select the **Allow only the following domains/IPs for callback after PDS operations** check box to enable the PDS to perform a back-link URL check to verify that a supplied back-link URL belongs to a certified user. If you select this option, click the **Add New Domain/IP** button to define each approved domain/IP (in the **Domain/IP** box).
 - 4 (For the shared Oracle database PDS topology only) In the Full Domain Cookie section, select the **Use full domain for PDS session** cookie check box to instruct the PDS to write the `PDS_HANDLE` cookie under the full domain of the PDS server, including the prefix. This is relevant for domains such as **bl.uk**, which cannot be used as subdomains.
 - 5 Click **Save & Continue** to save your updates.

NOTE:

When logging out of Aleph or DigiTool, a user name and password must be sent from the PDS to the application. The default values are `PDS/PDS`. To configure a different user name/password, access the PDS General Attributes page for the institution for which you want to configure a user name and password (see **Configuring Institutions Using the PDS Configuration Wizard** on page 26 for instructions on creating institutions) and enter the user name and password to be used in the **Service user name** and **Service password** fields under **Credentials for application logout service**. Click **Save & Continue** to save your settings. Note that

you can configure this option via the wizard only if you are working with PDS 2.x. This option is not available in the PDS 1.3 configuration wizard and must be configured manually if you are working with PDS 1.3.

Configuring SSO Manually

To configure SSO manually, you must configure the `general_conf` (or `sso_conf`) file, which exists by default in the `./pds/conf_table/` directory. This file defines the PDS' behavior upon login and logout in an SSO environment.

NOTE:

Both the `general_conf` and `sso_conf` files are supported. The PDS first looks for the `general_conf` file. If it cannot locate this file, it looks for the `sso_conf` file instead.

This section describes the following aspects of the `general_conf` (`sso_conf`) file:

- [Configuring Single Sign-On – \[LOGON\] Section](#)
- [Configuring Single Sign-Off – \[LOGOUT\] Section](#) on page 48
- [Callback URL Security – \[ALLOWED_HOSTS\] Section](#) on page 48
- [Persistent Cookies – \[PERSISTENT_COOKIE\] Section](#) on page 49

NOTE:

For an explanation of the `[DEFAULT_INSTITUTE]` section, see [Configuring Remote SSO Manually](#) on page 53.

Configuring Single Sign-On – [LOGON] Section

The `[LOGON]` section of the `general_conf` (or `sso_conf`) file contains the configuration of Single Sign-On across Ex Libris products. There is one configuration option:

- **TYPE 1: Enable automatic sign-on** – Enables a shared session between the various TYPE 1 applications listed.

For example, if a user logs in to MetaLib and then logs in to DigiTool, DigiTool queries the PDS to establish whether the user has a valid PDS cookie. If the user has a valid PDS cookie, the user will automatically be logged in to DigiTool.

In the following example, Single Sign-On is enabled for the following applications defined as TYPE 1:

```
[LOGON]
TYPE1 = digitool,metalib,aleph,dps,dep,voyager,primo
[END]
```

Configuring Single Sign-Off – [LOGOUT] Section

The [LOGOUT] section `general_conf` (or `sso_conf`) file contains the configuration of Single Sign-Off across Ex Libris products. There is one configuration option:

- **TYPE 1: Enable automatic sign-off** – When logging out of one application, the user is automatically logged out of all other applications defined as TYPE 1.

For example, if a user logs out of MetaLib, the user is logged out of DigiTool and Aleph as well.

In the following example, Single Sign-Off is enabled for the following applications defined as TYPE 1:

```
[LOGOUT]
TYPE1 = digitool,metalib,aleph,dps,dep,voyager,primo
[END]
```

When logging out of Aleph or DigiTool, a user name and password must be sent from the PDS to the application. The default values are `PDS/PDS`. To configure a different user name/password, add the following section to the relevant `./pds/conf_table/tab_service.<institute>` file:

```
[LOGOFF_ID]
params = <user name>,<password>
[END]
```

For example:

```
[LOGOFF_ID]
params = WWW-X,WWW-X
[END]
```

Callback URL Security – [ALLOWED_HOSTS] Section

The PDS can be configured not to redirect automatically to any URL it receives in the back-link URL parameter and to verify that the back-link URL belongs to a certified user. In this case, a callback is performed only if the URL appears in a list of approved domains. To ensure that a callback is performed to each domain used, ensure that all used domains are included in the list of approved domains.

The list of all approved domains is maintained in the [ALLOWED_HOSTS] section of the `general_conf` (or `sso_conf`) file. Domain names/IP addresses are separated by commas. The list is case-sensitive.

In addition, this section contains a flag that indicates whether the PDS should check the back-link URL against the list of approved domains.

For example:

```
[ALLOWED_HOSTS]
HOSTS = il-qalab06,primo.bl.uk,aleph.oxford,10.1.235.56
CHECK = Y
[END]
```

The PDS checks for a full match of the domain's name or IP.

If `CHECK = N` or there is no back-link URL in the request, the PDS proceeds with the redirection.

If `CHECK = Y` and the HOSTS list is empty or the back-link URL does not appear in the HOSTS list, the PDS displays an error page and stops.

The default PDS configuration allows the callback without checking.

Persistent Cookies – [PERSISTENT_COOKIE] Section

Although the default cookie is a session cookie, the PDS can be configured to use a persistent cookie. There are three configuration elements for persistent cookies:

- `pds_cookie_exp` – This parameter defines the duration of the persistent cookie. The default duration is 30 days.
- `&term` – A calling application can request a persistent cookie by sending the parameter `&term=long` in the `func=load-login`.
- `remember_me` – This parameter defines whether a **Remember me** check box will appear on the local login page.

The `pds_cookie_exp` and `remember_me` parameters are defined in the [PERSISTENT_COOKIE] section of the `general_conf` (or `sso_conf`) file.

```
[PERSISTENT_COOKIE]
pds_cookie_exp = 30
remember_me = Y
[END]
```

If `remember_me=Y`, the **Remember me** check box displays on the local login page and persistent cookie creation is determined by the check box selection. If the check box is selected, a persistent cookie is created and if it is not, a session cookie is created.

If `remember_me=N`, the **Remember me** check box does not display on the local login page and persistent cookie creation is determined by the configuration of the `term` parameter.

If the check box is not displayed and:

- `term = short` -> a session cookie is used
- `term = long` -> a persistent cookie is used
- `term` does not exist or is not short or long -> a session cookie is used

Synchronizing Subdomains for SSO

If you are working with the shared Oracle database PDS topology and the subdomains to which the various PDS instances write their PDS cookies differ, SSO will not work because one application does not have access to the PDS cookie written by another application.

The following is an illustration of such a situation:

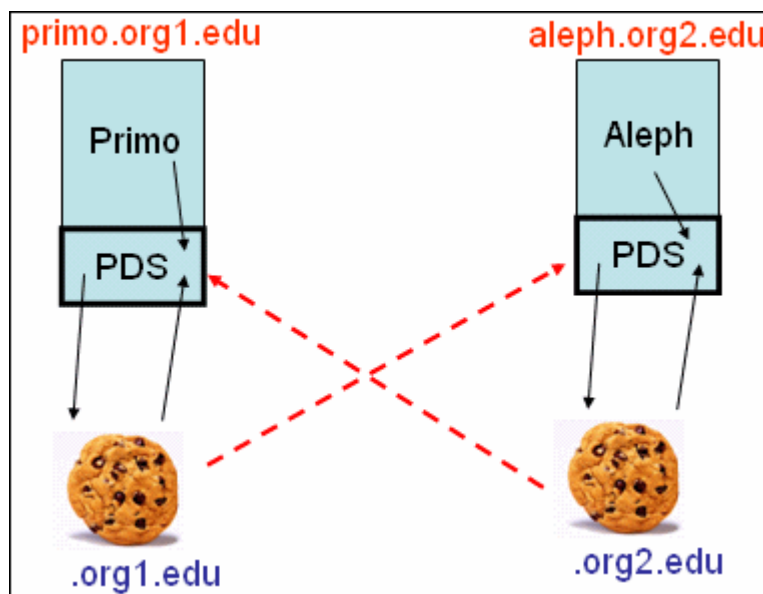


Figure 6: Different Subdomains

To enable SSO in such a situation:

- 1 For each application machine, create a new domain name. For example, for the Primo machine illustrated above, create the domain

`primo.newdomain.com` and for the Aleph machine, create the domain name `aleph.newdomain.com`.

- 2 Assign the new domain to the machine. For example, define the Primo machine illustrated above as `primo.newdomain.com` and the Aleph machine as `aleph.newdomain.com`.
- 3 Configure the application on each machine to redirect to the new PDS domain name. For example, configure Primo to redirect to `primo.newdomain.com` and Aleph to redirect to `aleph.newdomain.com`.
- 4 Configure the `server_pds` variable in the `PDSDefinitions` file of each PDS instance to point to the new domain.

The PDS cookie is written to the newly created, shared subdomain. All the applications can therefore access the `PDS_HANDLE` cookie.

NOTES:

- The new domain name is used only for the PDS and is transparent to users. Applications are accessed via their original domains.
 - If it is not possible for you to change the domain names, you will need to work with the local disk PDS topology—that is, one active PDS located on one of the application machines, with all the other applications linking to it. Note that this topology does not support failover between Ex Libris applications. If the active PDS is down, all applications are affected. (Failover between multiple Primo FEs, however, is supported.)
-

Remote SSO

If you are working with an environment in which users log in to a university's portal or another central login system and then move to a calling application, you can integrate your systems using remote Single Sign-On. Through the remote SSO service, users who are authenticated by the SSO system and then access the calling application are seamlessly redirected to the remote SSO system to determine whether they are already logged in. The remote SSO system then validates users and redirects them back to the PDS with the correct callback parameters, enabling login to the calling application without referral to a login page.

The remote SSO process is similar to the remote login scenario. However, unlike the remote login scenario, a user accessing an Ex Libris application with a remote SSO configuration is automatically logged in to the calling application, provided the user logged in to the university system prior to accessing the calling application. Whereas the remote login process is activated only when a user chooses to log in from a calling application, the remote SSO process is always activated at the start of a new calling application session. Even if the user

has not yet logged in, the user is redirected to the calling application as a GUEST user.

The differentiation between remote SSO and remote login allows for greater flexibility, as not all external systems can support SSO capabilities.

NOTES:

- You can configure both remote login and remote SSO services to cover two scenarios: one involving users first accessing a calling application and then initiating a login request and the other involving users first logging in to a central SSO system and then moving to a calling application.
 - Because remote SSO is defined per institution rather than per application (as is the case with non-remote SSO), the remote SSO service will work only if the PDS can identify the user's institution—for example, if there is only one defined institution for a calling application, if a calling application sends the user's institution to the PDS, or if a default SSO institution is defined for a calling application.
-

Configuring Remote SSO

If you are working with the local disk PDS topology, you can configure remote SSO via the configuration wizard, using the instructions in [Configuring Remote SSO Using the PDS Configuration Wizard](#) on page 52, or manually, using the instructions in [Configuring Remote SSO Manually](#) on page 53.

If you are working with the shared Oracle database PDS topology, you must configure remote SSO via the configuration wizard, using the instructions in [Configuring Remote SSO Using the PDS Configuration Wizard](#) on page 52.

Configuring Remote SSO Using the PDS Configuration Wizard

If you are working with the PDS configuration wizard, you configure remote SSO using two separate pages of the wizard.

To configure remote SSO using the configuration wizard:

- 1 On the first page of the configuration wizard, the PDS - SSO Configuration page, in the **Default Institution** field next to each of the enabled calling applications, enter the default institution to be used as the SSO institution if the institution parameter is not sent by the calling application.
- 2 Click **Save & Continue**. The PDS - Institution Configuration page opens. (See [Configuring Institutions Using the PDS Configuration Wizard](#) on page 26 for instructions on creating institutions.)

- 3 Access the PDS General Attributes page for the institution for which you want to configure remote SSO.
- 4 Under **Remote SSO**, select **Use remote SSO** and choose one of the following methods from the **Method** drop-down list:
 - **Remote** – In the **Path** field, enter the name of the HTML file you created for the remote SSO URL redirection (for example, `sso-remote-1`), indicate whether to use a secure ID, and enter the number of seconds for which you want the login to be valid. For an example of an HTML file that can be created and a list of the parameters to be included in this file, see page 55.
 - **CAS** – Use the CAS method. For details on CAS, see [CAS Authentication](#) on page 119.
 - **Shibboleth** – Use the Shibboleth method. For details on Shibboleth, see [Shibboleth](#) on page 129.
 - **iChain** – Use the iChain method. For details on iChain, see [iChain Authentication](#) on page 125.

Remote SSO: Use remote SSO Do not use remote SSO Method: Choose method
EZproxy: Use EZproxy Do not use EZproxy
Use LDAP attributes for Voyager BOR_INFO: Yes No
Change institution display order: select
Cancel & Go Back To Institution Configuration Save & Continue To Authentication Methods

Figure 7: Remote SSO Configuration

- 5 Click **Save & Continue**.

Configuring Remote SSO Manually

To configure remote SSO manually, you must define the following two services:

- **LOAD_SSO** – handles the redirection of the user’s browser to the external system. For details, see [Configuring LOAD_SSO](#) on page 54.
- **REMOTE_SSO** – handles the callback from the external system. For details, see [Configuring REMOTE_SSO](#) on page 56.

LOAD_SSO calls an HTML file pointing to the remote script to retrieve the user's ID. REMOTE_SSO handles the return call and adds the received user ID to the PDS session.

NOTES:

- For information on configuring an optional security enhancement for REMOTE_SSO, see **Optional Security Enhancement for REMOTE_LOGIN and REMOTE_SSO** on page 146.
- When there is more than one `tab_service.<institute>` file in the `./pds/conf_table` directory, the `general_conf` (or `sso_conf`) file needs to contain a `[DEFAULT_INSTITUTE]` section to define a default SSO institute to be used if the institution parameter is not sent by the calling application. In the following example, the default institute **LAW** is defined for the MetaLib calling application:

```
[DEFAULT_INSTITUTE]
LAW=metalib
[END]
```

Configuring LOAD_SSO

LOAD_SSO calls an HTML file pointing to the remote script in order to retrieve the user's ID.

```
[LOAD_SSO]
program      = remote_sso.pl
params       = sso-remote-1
[END]
```

The LOAD_SSO program receives the HTML file name in the `params` line and places standard parameters in the file to create the redirect request. In the above example, the HTML file is called `sso-remote-1`. The file specified must be placed in the `/pds/html_form/global` directory.

The following is an example of an `sso-remote-1` file:

```
<!-- remote sso -redirect -->
<html>
<head>
<title>Identification</title>
<include>meta-tags

<script language=Javascript>
  function redirect()
  {
    var url =  "http://hostname:port/cgi/mysso?" +
              "pds_handle=$0100&" +
              "calling_system=$0200&" +
              "institute=$0300&" +
              "url=$0400";
    top.location = url;
  }
</script>

</head>

<body onload="javascript:redirect()">
</html>
```

The following are the standard parameters included in the HTML redirection:

- **Redirect address** – This is the address of the remote SSO system and needs to be changed to the location of the remote SSO script.

For example:

```
var url="https://www.hostname.com/pin/default.asp?" +
```

- **PDS handle** – Use the following syntax: "pds_handle=\$0100&" +
- **Calling system** – Use the following syntax: "calling_system=\$0200&" +
- **Institution code** – Use the following syntax: "institute=\$0300&" +
- **Back-link URL** – This parameter is sent to the calling application (the Web page that the user came from) and must be the last parameter sent. It may be sent in either an enabled or disabled state. To enable the encoding of the back-link URL, use the following format: "url=\$0400"; To disable the encoding of the back-link URL, use the following format: "url=\$0410";

The following is an example of a request sent to the remote SSO system:

```
http://extserver/sso?
calling_system=exlibris&
institute=cityuniv&
pds_handle=&
URL= http://exlserver:8999/Z/7VNSL9REEEIPD-00001?func=quick-8
```

This file and link format can be edited to suit the link format expected by the remote SSO system. The PDS parameters and parameter names must be included as in the above example.

NOTE:

The parameters sent by the PDS to the remote system should be sent back as is and must not be changed during the processing of the remote script.

Configuring REMOTE_SSO

After the `LOAD_SSO` request is sent to the remote SSO system, the remote system sends a response back to the PDS as a URL. The PDS takes the ID provided in the URL and updates the PDS session accordingly. The PDS then retrieves user attributes as defined in the `[BOR_INFO]` section and redirects the browser back to the calling application using the returned back-link URL.

The following is an example of `REMOTE_SSO` configuration:

```
[REMOTE_SSO]
program      = remote_sso_gen_1.pl
params      =
[END]
```

The response of the remote SSO system needs to be a URL sent back to the PDS using the following URL format:

```
http://<server>:<port>/pds?func=remote-ss0
```

This URL must include the following five standard parameters (and only these parameters):

- **Institute code** – The returned parameter is `institute` and it contains the institute code sent in the `LOAD_SSO` URL.
- **PDS handle** – The returned parameter is `pds_handle` and it is empty.
- **Calling application/system** – The returned parameter is `calling_system` and it contains the name of the calling application sent in the `LOAD_SSO` URL.
- **User ID** – The returned parameter is `id`. If the user has previously logged in, the parameter contains the user ID. Otherwise, this parameter is empty.
- **Back-link URL to calling application** – The returned parameter is `url`, with the value as it was sent by the PDS. This must be the last parameter sent.

The following is an example of a remote SSO URL:

```
http://exlserver:8999/pds?func=remote-sso&
calling_system=exlibris&
institute=TESTINST&
pds_handle=&
id=04523&
url= http://exlserver:8999/z/7VNSL9REEEIPD-00001?func=quick-8
```

Configuring a Remote PDS SSO Check

If you are working with an Ex Libris SaaS product and a local product (for example, a local Primo installation and Alma), each product must be aware of the other's location. You must therefore define the PDS URL for each product in the Remote PDS SSO Check section of the PDS General Attributes page.

To configure a remote PDS SSO check:

- 1 Click **Save and Continue** on the PDS - SSO Configuration page (the first page of the wizard, which you access from the Ex Libris calling application). The PDS - Institution Configuration page opens.
- 2 Access the PDS General Attributes page for the institution for which you want to configure a remote PDS SSO check.
- 3 In the **Remote PDS SSO Check** section:
 - a Select the **Check for SSO in the following PDS** check box.



Figure 8: Remote PDS SSO Check – Check Box

- b For each product you are using, click the **Add New URL** button and enter the URL of the product's PDS in the **Domain/IP** box.

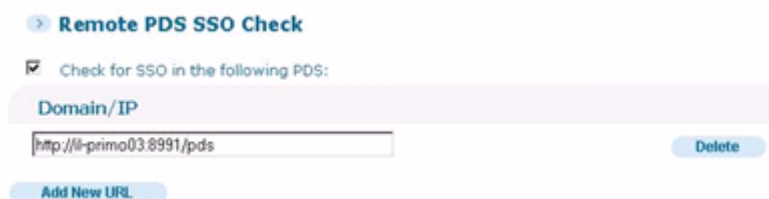


Figure 9: Remote PDS SSO Check – Add New URL

- 4 Click **Save & Continue** at the bottom of the page.

6

Login Configuration

This section includes:

- **Overview of Login Page Handling** on page 59
- **Configuring Remote Login** on page 61
- **Configuring Local Login** on page 67
- **Local Login Page Display** on page 89

Overview of Login Page Handling

If SSO is not being used or failed to authenticate a user, a load-login request (`pds?func=load-login`) is sent from the calling application to the PDS when a user chooses to log in. The PDS is then responsible to display the appropriate login page. A PDS installation can present a variety of login pages to the user, according to the information sent from the calling application and your PDS configuration.

Each institution can have one of two types of login pages:

- **Local login page** – The page is presented by the PDS based on the calling application parameter in the URL (for example, `&calling_system=aleph`). This page contains a drop-down list of all the local institutions. If there is an

institution parameter in the URL (for example, `&institute=Law School`), it is automatically selected from the drop-down list.



Figure 10: Ex Libris Local Login Page

NOTE:

The displayed local login page can be either a global login page or a customized institution login page. For more information on the local login page display, see [Local Login Page Display](#) on page 89.

-
- **Remote login page** – The PDS redirects the user to a login page that is not part of the PDS and can be located on any server.

The remote option is used only when the user's institution has configured the use of a remote login page. For information on configuring remote login, see [Configuring Remote Login](#) on page 61.

If remote login is not configured for the user's institution, the local login process is triggered when a load-login request is sent from the calling application to the PDS and the PDS seeks to authenticate the user, using either the calling application's own user database or an external authentication source (such as an LDAP server or a custom remote CGI hook). For information on the local login authentication process, see [Configuring Local Login](#) on page 67.

When the PDS works with several institutions, some may use local login pages and others may use remote login pages. If no institution parameter is included in the URL, the user is presented with a list of institutions from which to select.

When the user selects an institution from the list, the appropriate login page is displayed.

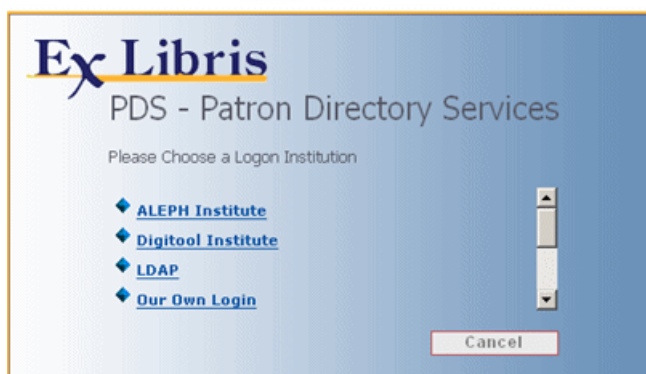


Figure 11: An Institution Selection Page

Configuring Remote Login

The PDS supports a remote login option, in which the PDS redirects the login request to a remote login page, typically presented by a central authentication server that the institution wants to use instead of the PDS.

In this scenario, the PDS must identify where to direct the user and must pass parameters to the remote system. Following authentication, a PDS session and `pds_handle` are created for the user. Upon request from the calling application, the PDS then proceeds to fetch user attributes from the remote system, as described in [User Attribute Retrieval and Attribute Mapping](#) on page 97, so that it can log the user in to the calling application.

To configure remote login:

- If you are working with the local disk PDS topology, you can configure remote login via the configuration wizard, using the instructions in [Configuring Remote Login Using the PDS Configuration Wizard](#) below, or manually, using the instructions in [Configuring Remote Login Manually](#) on page 63.
- If you are working with the shared Oracle database PDS topology, you must configure remote login via the configuration wizard, using the instructions in [Configuring Remote Login Using the PDS Configuration Wizard](#) below.

Configuring Remote Login Using the PDS Configuration Wizard

If you are working with the PDS configuration wizard, you configure remote login using the PDS General Attributes page.

To configure remote login using the configuration wizard:

- 1 Access the PDS General Attributes page for the institution for which you want to configure remote login. (See [Configuring Institutions Using the PDS Configuration Wizard](#) on page 26 for instructions on creating institutions.)
- 2 Under **Remote Login**, select **Use remote login** and choose one of the following methods from the **Method** drop-down list:
 - **Remote** – In the **Path** field, enter the name of the HTML file you created for the remote login URL redirection (for example, `load-login-cityuniv`), indicate whether to use a secure ID, and enter the number of seconds for which you want the login to be valid. For an example of an HTML file that can be created and a list of the parameters to be included in this file, see page 65.
 - **CAS** – Use the CAS method. For details on CAS, see [CAS Authentication](#) on page 119.
 - **Shibboleth** – Use the Shibboleth method. For details on Shibboleth, see [Shibboleth](#) on page 129.
 - **iChain** – Use the iChain method. For details on iChain, see [iChain Authentication](#) on page 125.

The screenshot shows the 'PDS General Attributes' configuration page. It contains several sections:

- Code:** A text input field containing 'USM50'.
- Description:** A text input field containing 'ALEPH Institute'.
- Character conversion:** A dropdown menu set to 'NONE'.
- Calling application & override code:** A list of checkboxes for 'Primo', 'MetaLib', 'Aleph', 'Rosetta', 'Voyager', and 'DigiTool', each followed by an empty text input field.
- Remote login:** Two radio buttons: 'Use remote login' (selected) and 'Do not use remote login'. To the right is a 'Method:' dropdown menu with a list of options: 'Choose method', 'Remote', 'CAS', 'Shibboleth', and 'iChain'.
- Logout from PDS:** Two checkboxes: 'Redirect logout' and 'Remote logout', each followed by an empty text input field.

Figure 12: Remote Login Configuration

3 Click **Save & Continue**.

Configuring Remote Login Manually

If you are configuring remote login manually, it is necessary to configure the following two separate services in the `./pds/conf_table/tab_service.<institute>` file:

- **LOAD_LOGIN** service – handles the communication from the PDS when redirecting the login request to the remote login program. For details, see [Configuring LOAD_LOGIN](#) on page 63.
- **REMOTE_LOGIN** service – handles the communication from the remote login program when it sends the required callback parameters to the PDS. For details, see [Configuring REMOTE_LOGIN](#) on page 65.

Configuring LOAD_LOGIN

The `LOAD_LOGIN` functionality is called from the calling application when a user requests a login page and the PDS is configured to redirect to a remote login page.

The redirect will take place in either one of the following circumstances:

- An institution parameter has been passed to the PDS when `LOAD_LOGIN` is requested and the relevant `tab_service.<institute>` file is configured with a `LOAD_LOGIN` service.
- There is only one `tab_service.<institute>` file in the `./pds/conf_table` directory and this file is configured with a `LOAD_LOGIN` service.

Following is an example of `LOAD_LOGIN` configuration in the `tab_service.<institute>` file:

```
[LOAD_LOGIN]
program = remote_load_login_encoded.pl
params  = load-login-cityuniv,Y
[END]
```

NOTE:

In the `./pds/conf_table/tab_service.<institute>` file, the `LOAD_LOGIN` service needs to be defined instead of the `AUTHENTICATE` service.

The `remote_load_login_encoded.pl` program used is a generic program that places standard parameters in an HTML page with the redirect request. The `load-login-cityuniv` on the `params` line points to the HTML file that contains the actual redirection to the URL of the remote login page. The `,Y` on the `params` line activates UTF-8 encoding. The Web page (HTML file) is the value of the `params` parameter (in the above example, `load-login-cityuniv`) and should be placed either in the `./pds/html_form` global directory, or in the appropriate `./pds/html-form/calling_system-xxx` directory.

In order to pass the user interface language to the remote authentication program, customize separate HTML files per language, for example `load-login-aub.ger`, `load-login-aub.fra`, etc. Place the files in the `institute-<INST_CODE>` directory under `html_form`. Each file redirects to the remote login system and send the relevant language parameter.

The following is an example of an HTML file configured to support LOAD_LOGIN:

```
<!-- login-redirect -->
<html>
<head>
<title>Identification</title>
<include>meta-tags

<script language=JavaScript>
function redirect()
{
var url = "https://authserver/cgi/login.asp?" +
"fixed_par1=F1&" +
"fixed_par2=F2&" +
"pds_handle=$0100&" +
"calling_system =$0200&" +
"institute=$0300&" +
"url=$0400";

top.location = url;
}
</script>

</head>

<body onload=" javascript:redirect()">
</body>
</html>
```

Include the following standard parameters in the Web page with the redirect:

- Institution code ("institute")
- PDS handle (which remains unpopulated)
- Calling application ("calling_system")
- Back-link URL (the URL from which the user originated). This must be the last parameter. ("url")

NOTE:

Instead of using the `remote_load_login_encoded.pl` program, you can choose to use CAS, iChain, or Shibboleth authentication. For information on these remote authentication programs, see [CAS and iChain Authentication](#) on page 119 and [Shibboleth](#) on page 129.

Configuring REMOTE_LOGIN

This service supports the asynchronous callback of the remote login system to the PDS. It is mandatory to have a callback function in the PDS to handle the return of the remote login. The section in the `tab_service.<institute>` file is called [REMOTE_LOGIN] and the standard program is called `remote_login_gen_1.pl`.

NOTE:

Instead of using the `remote_login_gen_1.pl` program, you can choose to use CAS, iChain, or Shibboleth authentication. For information on these remote authentication programs, see **CAS and iChain Authentication** on page 119 and **Shibboleth** on page 129.

The program retrieves the parameters passed back from the server, analyzes them, updates the PDS session, and then redirects the user back to the calling application.

For example:

```
[REMOTE_LOGIN]
program = remote_login_gen_1.pl
params = check,1800
[END]
```

The following table lists the standard parameters that must be passed back with the REMOTE_LOGIN:

Table 3. REMOTE_LOGIN Parameters

Parameter Name	Example
User ID	ID=0123456
Calling system	metalib, aleph, digitool, primo, or voyager
Institution code	institute=HUJI
PDS handle	pds_handle (currently empty; reserved for future development)
Back-link URL to calling application	url=http://...

The back-link URL must remain the last parameter, as everything that follows it will be considered part of the back-link.

The following is an example of REMOTE_LOGIN input:

```
http://il-metalsfx01.corp.exlibrisgroup.com:13006/pds?func=remote-
login&
calling_system=metalib&
institute=REMOTE&
id=DEMO&
url=http://il-metalsfx01.corp.exlibrisgroup.com:8331/V
```

In the above example, the remote CGI script returns a parameter, `id=DEMO`, in addition to the calling application, `institute`, and URL parameters which it received from the PDS. This means that the user was authenticated and the

`remote_login_gen_1.pl` program will create a `z311` file on the disk and return a `pds_handle` to the calling application.

The following is an example of a full remote login configuration that shows the addition of a new institution named **business** (`file=tab_service.business`). This institution uses the remote login, whereby PDS redirects the login request to a remote authentication server where it authenticates the user. The `BOR_INFO` services are defined via a remote CGI script.

```
[LOAD_LOGIN]
program = remote_load_login_encoded.pl
params = load-login-business-pin
[END]
[REMOTE_LOGIN]
program = remote_login_business.pl
params = check,1800
[END]
[BOR_INFO]
program = remote_cgi_hook.pl
params = GET,host:port,cgi-bin/rp_business.pl
[END]
```

NOTE:

For information on configuring an optional security enhancement for `REMOTE_LOGIN`, see [Optional Security Enhancement for `REMOTE_LOGIN` and `REMOTE_SSO`](#) on page 146.

Configuring Local Login

If remote login is not configured and the local login process is triggered when a `load-login` request is sent from the calling application to the PDS, the PDS seeks to authenticate the user performing the login.

You can configure one or more authentication methods for each institution. The PDS attempts to authenticate the user with the first-listed authentication method. If the authentication fails, it will try the next authentication method. If authentication fails with all of the methods defined, the user will be redirected back to the login page. If authentication succeeds, a PDS session and `pds_handle` are created for the user. Upon request from the calling application, the PDS then proceeds to fetch user attributes, as described in [User Attribute Retrieval and Attribute Mapping](#) on page 97.

To configure authentication methods:

- If you are working with the local disk PDS topology, you can configure authentication methods via the configuration wizard, using the instructions in [Configuring Authentication Methods Using the PDS Configuration](#)

Wizard below, or manually, using the instructions in **Configuring Authentication Methods Manually** on page 76.

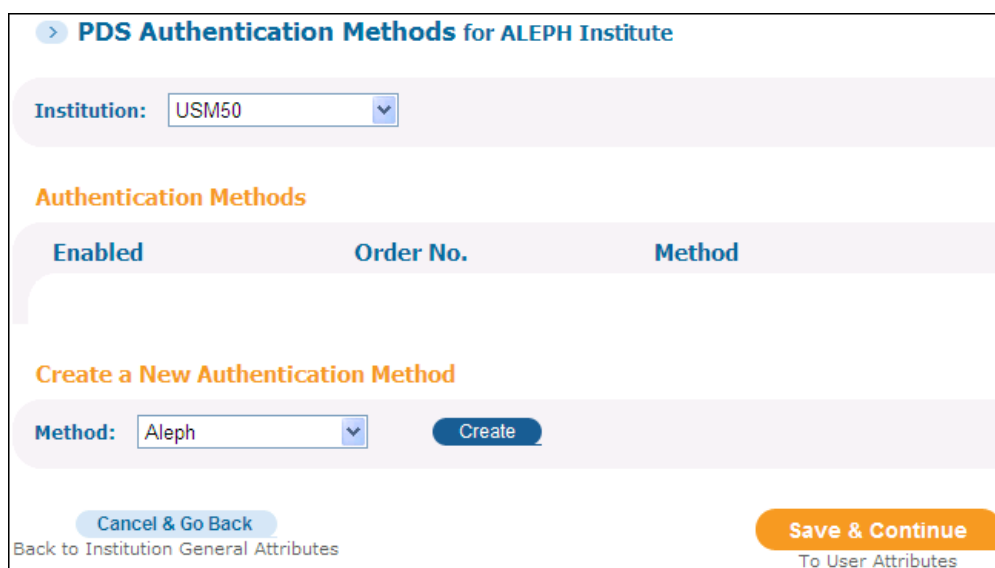
- If you are working with the shared Oracle database PDS topology, you must configure authentication methods via the configuration wizard, using the instructions in **Configuring Authentication Methods Using the PDS Configuration Wizard** below.

Configuring Authentication Methods Using the PDS Configuration Wizard

If you are working with the PDS configuration wizard, you configure authentication methods using the PDS Authentication Methods page.

To create an authentication method using the configuration wizard:

- 1 Click **Save & Continue** on the PDS General Attributes page.
The PDS Authentication Methods page opens.



The screenshot shows the 'PDS Authentication Methods for ALEPH Institute' page. At the top, there is a breadcrumb '> PDS Authentication Methods for ALEPH Institute'. Below that is a form with 'Institution: USM50' and a dropdown arrow. The main section is titled 'Authentication Methods' and contains a table with columns 'Enabled', 'Order No.', and 'Method'. Below the table is a section 'Create a New Authentication Method' with a 'Method:' dropdown set to 'Aleph' and a 'Create' button. At the bottom, there are two buttons: 'Cancel & Go Back' (with a link 'Back to Institution General Attributes') and 'Save & Continue' (with a link 'To User Attributes').

Figure 13: PDS Authentication Methods

- 2 From the **Method** drop-down list, select one of the following authentication methods: **Aleph**, **MetaLib**, **Voyager**, **DigiTool**, **LDAP**, **CGI Hook**, or **CGI Hook SSL-1**.

NOTE:

z312 is not an available option.

- 3 Click **Create**. The new authentication method displays in the list of authentication methods.

- 4 Click **Edit** to customize the authentication method to suit your system’s needs. The PDS - Configure page displays the selected authentication method.

Configuration of Aleph Authentication Method

Host name/IP:

Port:

Operation code:

Admin library:

Use secure: Do not use secure
 Use secure

Service name:

Service password:

Encrypt service password: Yes
 No

Cancel & Go Back
 To Authentication Methods

Save & Continue
 To Authentication Methods
Note: Will be saved in file only when Authentication Methods is saved

Figure 14: PDS Configure Page - Aleph Example

- 5 Edit the fields according to the method on which your authentication method is based:
 - If the authentication method is based on an internal authentication method, such as Aleph, MetaLib, DigiTool, or Voyager (see the above figure for an example), enter the information according to the following table:

Table 4. PDS - Configure Internal Authentication Details

Field Name	Description
Host name/IP	Enter the host name or IP address of the calling application’s X-Server.
Port	Enter the Web port of the calling application’s X-Server.

Table 4. PDS - Configure Internal Authentication Details

Field Name	Description
Operation code	<p>Select the X-Server service with which you want to query the calling application. The following two options are available:</p> <ul style="list-style-type: none"> ■ BOR_AUTH ■ BOR_INFO <hr/> <p>NOTE: It is recommended that you use BOR_AUTH and not BOR_INFO for Aleph.</p>
Admin library (Aleph only)	Specify the administrative library in Aleph in which you want to search for users in order to authenticate them. For example, USM50 .
Use secure (not available for Voyager)	Specify whether or not to use an HTTPS call (Use secure or Do not use secure)
User schema (DigiTool only)	Enter the code of the user schema in DigiTool in which you want to search for users in order to authenticate them. For example, DAT01 .
Service name (not applicable for Voyager)	Enter the user name used by the PDS to obtain permission to use X-Server services.
Service password (not applicable for Voyager)	Enter the password used by the PDS to obtain permission to use X-Server services.
Encrypt service password (not applicable for Voyager)	This field indicates whether the service password is encrypted in a file on the server. Select Yes to use an encrypted password.

- If the authentication method is based on the LDAP external authentication method, enter the information according to the following table (see the figure below the table for an example):

Table 5. PDS - Configure LDAP Details

Field Name	Description
LDAP name (Note: Relevant for shared Oracle DB topology only)	Enter a name for the current LDAP configuration (mandatory). This field enables multiple LDAP configurations for a single institution.
Secure LDAP	This field indicates whether secure or non-secure communication should be established. If you select the secure method, the LDAPS protocol is used to communicate with the LDAP server.
Host name/IP	Enter the host name or IP address of the LDAP server.
Port	Enter the port of the LDAP server.
LDAP version	Select the relevant LDAP version (2 or 3).
Initial bind name and password	Many LDAP servers enable anonymous login, in which case no user name or password is necessary. If a user name and password are required for initial binding, enter the user name and password assigned to the institution for accessing the LDAP server.
Encrypt initial bind password	This field indicates whether the initial bind password is encrypted in a file on the server. Select Yes to use an encrypted password.

Table 5. PDS - Configure LDAP Details

Field Name	Description
Bind and search	<p>Select one of the following bind and search options:</p> <ul style="list-style-type: none"> ■ Default – This option instructs the PDS to perform the LDAP bind step (using the full distinguished name together with the password provided by the user) initially and then search the LDAP server using the search bases and filters (returning <code>bor_info</code>). ■ Do not use bind – This option instructs the PDS to skip the bind step and return <code>bor_info</code>. ■ Do not use search – This option instructs the PDS to perform the LDAP bind step (using the full distinguished name together with the password provided by the user) and return <code>bor_auth</code>. ■ Bind before search – This option instructs the PDS to perform the LDAP bind step (using the full distinguished name together with the password provided by the user) prior to each search of the LDAP server (using the search bases and filters and returning <code>bor_info</code>). ■ Full distinguished name – Enter the full distinguished name, which is then bound with the user’s password to authenticate using the LDAP server. Note that the full distinguished name is required only if you selected the Bind before search option.
Search base	<p>Specify the full path of the search as it appears in the LDAP directory tree. To add additional search bases, click Add New Search Base.</p> <hr/> <p>NOTE: If the Do not use search option is set, this field is not used.</p> <hr/>
Search filter	<p>Specify the parameter that is used to filter the search results. Select the filtering parameter that returns only one object.</p> <hr/> <p>NOTE: If the Do not use search option is set, this field is not used.</p> <hr/>

Table 5. PDS - Configure LDAP Details

Field Name	Description
Switch to TLS	Select Yes to convert the existing connection using Transport Layer Security (TLS), which provides an encrypted connection. This is only possible if the connection uses LDAP version 3.
UTF to character set	Select ISO-8859-1 to convert the password entered by the user from UTF8 to ISO-8859-1 before sending it to LDAP.
LDAP response convert to	This value is used to set the encoding of the LDAP response before sending it back to the calling application. If you want to encode the LDAP response, select UTF8 . Otherwise, the response is not encoded.
LDAP Attribute Mapping	<p>The LDAP tag names need to be mapped to names that the calling application recognizes. For each LDAP attribute that you want to map:</p> <ol style="list-style-type: none"> 1 Click the Add New Mapping button. 2 In the LDAP System Attribute field, enter the name of the LDAP attribute you want to map (for example, PortalName). 3 (Optional) In the LDAP System Value field, enter a specific value of the LDAP attribute that you want to map to a calling application attribute or attribute value (for example, EBSCO). 4 From the PDS Attribute drop-down list, select the calling application attribute to which you want to map the LDAP attribute you defined (for example, portal_name to map to PortalName). 5 (Optional) In the PDS Value field, enter a specific value of the calling application attribute to which you want to map the LDAP attribute or LDAP attribute + value that you defined (for example, portal_name, EBSCO to map to PortalName, EBSCO). 6 To enable the mapping you defined, ensure that the Enabled check box is selected.

Configuration of LDAP Authentication Method

LDAP name:

Secure LDAP: Use a secure LDAP
 Do not use a secure LDAP

Host name/IP:

Port:

LDAP version: V3
 V2

Initial bind name:

Initial bind password:

Encrypt initial bind password: Yes
 No

Bind and search: Default
 Do not use bind
 Do not use search
 Bind before search

Full distinguished name:

Search base: [Delete](#)

Search filter:

[Add New Search Base](#)

Switch to TLS: Yes
 No

UTF to character set:

LDAP response convert to:

LDAP Attribute Mapping

Enable	LDAP System Attribute	LDAP System Value	PDS Attribute	PDS Value
<input checked="" type="checkbox"/>	<input type="text" value="cn"/>	<input type="text"/>	<input type="text" value="user_name"/>	<input type="text"/>
<input checked="" type="checkbox"/>	<input type="text" value="sn"/>	<input type="text"/>	<input type="text" value="profile-id"/>	<input type="text"/>
<input checked="" type="checkbox"/>	<input type="text" value="mail"/>	<input type="text"/>	<input type="text" value="email_address"/>	<input type="text"/>
<input checked="" type="checkbox"/>	<input type="text"/>	<input type="text"/>	<input type="text" value="group"/>	<input type="text" value="demo"/>

[Add New Mapping](#)

[Cancel & Go Back](#)
To Authentication Methods

[Save & Continue](#)
To Authentication Methods
 Note: Will be saved in file only
 when Authentication Methods is saved

Figure 15: PDS Configure Page - LDAP Example

- If the authentication method is based on CGI hook or CGI hook SSL, enter the information according to the following table (see the figure below the table for an example):

Table 6. PDS - Configure CGI Hook/CGI Hook SSL Details

Field Name	Description
Host name/IP	Enter the host name or IP address of the remote CGI script.
Port	Enter the Web port of the remote CGI script.
User default CGI script	Enter the CGI parameters for the remote CGI script (for example, <code>cgi-bin/auth-ng/primo-bor-info.cgi</code> or <code>cgi-bin/remote_hasharon.pl</code> . For information on the target attributes to be used in this script, see Appendix C: Ex Libris Calling Application Target Attributes .
HTTP command	Select the syntax that you want to use: either GET or POST .

Configuration of CGI-Hook Authentication Method

Host name/IP:

Port:

User default CGI script:

HTTP command:

Cancel & Go Back
To Authentication Methods

Save & Continue
To Authentication Methods
Note: Will be saved in file only when Authentication Methods is saved

Figure 16: PDS Configure Page - CGI Hook Example

- 6 Click **Save & Continue** to save your modifications and return to the Authentication Methods page.

Your authentication method appears in the list of authentication methods. It is recommended that you use the instructions in [Testing Authentication Methods](#) below to ensure that the authentication method you configured functions properly.

Testing Authentication Methods

It is recommended that you test the newly created authentication method to ensure that it is functioning properly.

To test an authentication method:

- 1 In the Authentication Methods section, click **Test** next to the authentication method you want to test.

The Test for Authentication Method window opens.

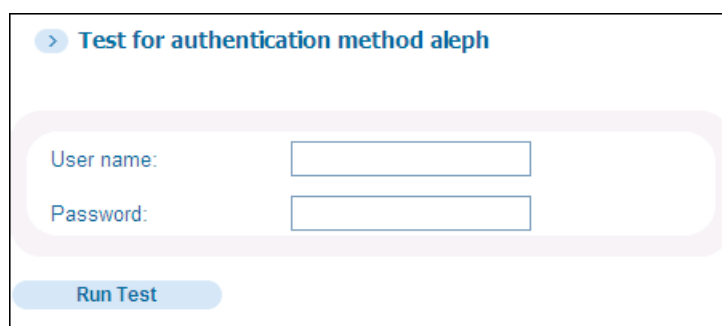


Figure 17: Test for Authentication Method Window

- 2 In the **User name** and **Password** fields, enter the user name and password with which you log in to the institution.
- 3 Click **Run Test** to test the authentication method.

A separate page displays with the results of the authentication method test. If the test is successful, click **Save & Continue** on the Authentication Methods page. If the test was not successful, you can try correcting the user authentication method information using the instructions in the previous section, or you can call your local Ex Libris representative for support.

Configuring Authentication Methods Manually

If you are configuring authentication methods manually, you must configure the AUTHENTICATE service in each institution's `tab_service.<institute>` file.

NOTE:

The AUTHENTICATE service should not be used in conjunction with LOAD_LOGIN, as LOAD_LOGIN will override the user authentication configuration.

The following table lists the authentication programs that can be used in configuring the AUTHENTICATE service:

Table 7. Authentication Programs

Program	Description
aleph_141.pl	<p>Uses the Aleph X-Server – sends host and port as well as other parameters to the X-Server. For example: <code>www.cityuniv,8991,BOR_AUTH,USM50,N</code> will be sent to the X-Server as:</p> <p><a href="http://www.cityuniv:8991/X?op=BOR_AUTHENTICATE&BOR_ID=<xxx>&VERIFICATION=<xxx>&BOR_LIBRARY=USM50">http://www.cityuniv:8991/X?op=BOR_AUTHENTICATE&BOR_ID=<xxx>&VERIFICATION=<xxx>&BOR_LIBRARY=USM50</p> <p>Depending on the N, the call will be HTTPD or HTTPDS.</p>
aleph_142.pl	
aleph_152.pl	
aleph.pl	
metalib_x_server.pl	<p>Uses the MetaLib X-Server – sends host and port as well as other parameters to the X-Server. For example:</p> <p><code>il-cleo01.corp.exlibrisgroup.com,8334,BOR_INFO,N,pds,pds</code></p>
digitool.pl	<p>Uses the DigiTool X-Server – sends host and port as well as other parameters to the X-Server. For example:</p> <p><code>ram7:8881,op=BOR_AUTH,USR00,PDS,PDS,N</code></p>
dps.pl	<p>Uses the Alma X-Server – sends host and port as well as other parameters to the X-Server. For example:</p> <p><code>il-dtldev07a.corp.exlibrisgroup.com,1801,BOR_AUTH,N,EXLDEV1_INST</code></p>
voyager.pl	<p>Uses the Voyager X-Server – sends host:port and other parameters to the X-Server:</p> <p><code>10.111.111.99:17304,auth,N</code></p>
ldap.pl	<p>Interfaces with an LDAP server for remote authentication. For each institute, this program requires the <code>ldap_prn.conf</code> configuration file.</p>
remote_cgi_hook.pl	<p>Remote authentication hook. For example:</p> <p><code>GET,130.182.125.3:2080,cgi-bin/rpa_cla.pl</code></p>
remote_cgi_hook_ssl_1.pl	<p>Remote authentication hook with SSL (Secure Sockets Layer) support for secure communication between the PDS hook and the local CGI program.</p> <p>For example: <code>POST,https://vulture.library.colostate.edu:443,cgi-bin/metauth.cgi,BOR_ID</code></p>

NOTE:

The Aleph interfaces depend on the version of Aleph that you are using. `Aleph.pl` works with Aleph 16 as well as Aleph 17 and later, but `Aleph_152.pl` does not work with Aleph 17.

As described in the above table, the PDS can authenticate users:

- against an external authentication source – For example, by using an LDAP server or a custom remote CGI hook, users can write their own custom authentication. For detailed information on configuring the LDAP and CGI hook/SSL hook external authentication services, see **LDAP Services – Manual Configuration** on page 81 and **Remote CGI Hook – Manual Configuration** on page 86. Note that these sections describe both user authentication and user attribute retrieval configuration.
- against a calling application’s own user database – A Perl interface is used to activate the calling application’s X-Server and validate the user name and password entered on the login page. This authentication method is described below.

NOTE:

For a list of values that the X-Servers can return, refer to the X-service documentation for your product.

You must configure parameters in the [AUTHENTICATE] section of the institution’s `tab_service.<institute>` file that specify the following:

- connection information for contacting the calling application’s X-Server, such as the server address and port to access when calling the X-Server
- the operation code, or X-Server service, with which you want to query the calling application. The following two options are available:
 - BOR_AUTH
 - BOR_INFO

NOTE:

It is recommended that you use BOR_AUTH and not BOR_INFO for Aleph.

- whether or not to use a secure connection (the `N` or `Y` parameter)

NOTE:

This option is not applicable for Voyager.

- (for Aleph only) the administrative library in Aleph that you want to query – for example, USM50.
- (for DigiTool only) the user schema that you want to query – for example, DAT01.

- the user name and password (the `service_user` and `service_password` parameters) used by the PDS to obtain permission to use the X-Server services. The default user name and password are `pds, pds`. It is recommended to change the default password, in which case the user name and password must be included in the `params` line. For information on encrypting this password, see [Encrypt Password Utility](#) on page 163.

NOTE:

This option is not applicable for Voyager.

IMPORTANT:

The PDS user name and password must be changed in the calling application in order to authorize the PDS to use the relevant X-Server services. Refer to the documentation for your product to determine how to change the PDS user name and password.

Examples of Authentication Configuration

The following are some examples of how to configure the PDS for authentication services.

NOTE:

When requesting a secure HTTPS connection (Y) in the interface, the port must be the secure port. For example:

```
PROGRAM-NAME      aleph.pl
160.45.152.196,443,BOR_AUTHENTICATE,FUB00,Y,WWW-X,WWW-X
```

Aleph

```
[AUTHENTICATE]
program = aleph.pl
params = il-aleph02,8994,BOR_AUTH,USM50,N,WWW-X,WWW-X
[END]
```

In addition, verify that the Aleph X-Server is working correctly (license and so forth) and that the WWW-X user name and WWW-X password is defined in your management interface.

DigiTool

```
[AUTHENTICATE]
program      = digitool.pl
params      = ram7:8881,op=BOR_AUTH,DAT01,PDS,PDS,N
[END]
```

MetaLib

```
[AUTHENTICATE]
program      = metalib_x_server.pl
params      = il-cleo01.corp.exlibrisgroup.com,8334,BOR-
AUTH,N,PDS,PDS
[END]
```

Alma

```
[AUTHENTICATE]
program      = dps.pl
params      = il-
urm08.corp.exlibrisgroup.com,1801,BOR_AUTH,N,EXLDEV1_INST
[END]
```

Voyager

```
[AUTHENTICATE]
program      = voyager.pl
params      = 10.111.111.99:17304,auth,N
[END]
```

LDAP

```
[AUTHENTICATE]
program = ldap.pl
params = ldap_prn.conf
[END]
```

NOTE:

You can configure the PDS so that it first attempts to authenticate users via an external authentication service and only if this fails, does it then attempt to authenticate users via an internal authentication service. To configure this workflow, the external authentication service must be listed first in the `tab_service.<institute>` file, before the internal authentication service using the calling application's X-Server. For example:

```
[AUTHENTICATE]
program = remote_cgi_hook.pl
params = GET,www.exlibris.com:8331,aleph-cgi/
remote_cgi_hook
program = metalib_x_server.pl
params = hostname,8331,BOR-AUTH,N,pds,pds
[END]
```

LDAP Services – Manual Configuration

The PDS has a standard program, `ldap.pl`, to interface with an LDAP server for remote authentication and the fetching of user attributes.

The `ldap.pl` program needs to interface with many different LDAP servers and many different configurations. It has a configuration file that defines how it interacts with the server. See [LDAP Configuration File](#) on page 81 for a full explanation of all the parameters in this file.

The flow of the `ldap.pl` script is as follows:

- The LDAP script (`ldap.pl`) reads the `ldap_prn.conf` file.
- The PDS connects to the given LDAP host name and establishes a secure or non-secure communication (depending on the configuration of the `secure_ldap` flag).
- If initial bind parameters (`init_bind_dn` and `init_bind_password`) are defined in the configuration file, an initial bind using the configuration file parameters is performed.
- The PDS searches the LDAP tree to find the user's record, according to the provided `search_base` and `search_filter`. The hard-coded `USERNAME` token is replaced with the name from the login page and the token `PASSWORD` is replaced with the verification from the login page.
- If the results are not unique (or zero size result), the search step is repeated for the next provided base/filter.
- If there is no `bor_info_only = Y` parameter, a bind is attempted for the search object, using the password from the login page.
- If `bindb4search = Y`, the `dn` parameter must be defined. PDS performs the LDAP bind step (using the `dn` together with the password provided by the user) prior to each search of the LDAP server (using the search bases and filters and returning `bor_info`).
- If the bind succeeds, `<auth>Y</auth>` is returned in the XML and the `AUTHENTICATE` or `BOR_INFO` step succeeds. Otherwise, the XML will contain `<auth>N</auth>`, the return code will be 11, and an error message will be displayed on the login page. A more specific error message will be printed to the `$LOGDIR/pds_server.log`.

NOTE:

PDS logs can be restarted using the `start_w` command. PDS logs that are older than 30 days are automatically removed.

LDAP Configuration File

The LDAP configuration file contains the following sections:

- General – For details, see [\[GENERAL\]](#) on page 82.

- XML setting – For details, see [XML_SETTING] on page 83.
- Attribute mapping – For details, see [ATTRIBUTES_MAPPING] on page 83.
- Defaults – For details, see [DEFAULTS] on page 83.

[GENERAL]

This section defines the following attributes of the LDAP server interaction:

- The **host name** and **port** of the remote LDAP server.
- **initial user name** and **password**. Many LDAP servers enable anonymous login, in which case no user name/password is required. If a user name/password is required for initial binding, enter the user name/password assigned to the calling application for accessing the LDAP server (in the `init_bind_dn` and `init_bind_password` parameters).
- **secure_ldap** Y/N flag, which defaults to **N**. If the flag is set to **Y**, the SSL protocol is used to communicate with the LDAP server.
- **init_bind_dn** – Enter the full **dn** (distinguished name) for the initial bind.
- **init_bind_password** – Enter the dn's password for the initial bind. For instructions on encrypting this password, see [Encrypt Password Utility](#) on page 163.
- **search_base** – Enter the full path search in the `ldap` directory tree to the user.
- **search_filter** – Enter the parameter to filter the results to return only one object.

NOTE:

The `search_base` and `search_filter` parameters can be repeated to search in more than one tree.

- **ldap_version** is not a mandatory line. If it is not present or if it is empty, the `ldap.pl` will attempt a bind with its default version, which is 2. If the LDAP server is a version 3 server, the `ldap_version` in the `conf_table` must be set to 3 for the bind to be successful.
- **bor_info_only** is an optional line. If it is present, the `ldap.pl` will skip the bind step and return `bor_info` without attempting to bind.
- **auth_only** is an optional line. If it is present, the `ldap.pl` will only validate the password and will not return `bor_info`.

NOTES:

- If `auth_only = Y`, `dn` is a mandatory line and contains the full distinguished name for the bind. The `search_base` and `search_filter` are not used in this scenario.

- `auth_only = Y` and `bor_info_only = Y` are mutually exclusive.
-
- `start_tls = Y`, switch to secure. Calling this method will convert the existing connection to use Transport Layer Security (TLS), which provides an encrypted connection. This is possible only if the connection uses LDAP version 3.
 - `utf_to_charset = ISO-8859-1`
 - `ldap_resp_convert` – This line is used for setting the encoding of the LDAP response before sending it back to the calling application. The only possible value is UTF8. No encoding is enabled for other values.
 - `convert_response_from_utf8_to_charset` - When the parameter holds a valid charset, for example 'ISO-8859-1', the LDAP response is decoded from utf-8 to the charset.
 - `bindb4search` Y/N flag, which defaults to N. See Bind and Search in the [PDS - Configure LDAP Details](#) on page 71

[XML_SETTING]

This section is required by the PDS and must not be modified or deleted.

The following is an example of this section:

```
[XML_SETTING]
xml_root_node = bor_authentication
[END]
```

[ATTRIBUTES_MAPPING]

This section is used to map the LDAP user record fields to the calling application's user record fields in the user table. For more information on this section, see [Attribute Mapping and Defaults](#) on page 85.

The following is an example of this section:

```
[ATTRIBUTES_MAPPING]
cn = name
mail = email_address
[END]
```

[DEFAULTS]

This section sets defaults for user attribute mapping. You can set a default value for any of the fields that can be mapped to the user record, particularly if the site wants to force a default value for all users. For more information on this section, see [Attribute Mapping and Defaults](#) on page 85.

For example, to set the **expiration date** for all users to **20201231**, use the following:

```
[DEFAULTS]
expiry-date,20201231
[END]
```

LDAP Configuration File Examples

The following is an example of an initial bind setup using an LDAP version 3 server:

```
[GENERAL]
host_name = cityuniv.library.edu
port      = 389
init_bind_dn = cn=Ldap ,o=Citytown University,c=US
init_bind_password = t670P11
ldap_version = 3
[END]
```

The following is an example of a `search_base` and `search_filter` for querying the LDAP server. The `search_base` defines the LDAP base to be searched and the `search_filter` defines the LDAP user. The `USERNAME` token is a placeholder and is replaced with the name value from the login page.

```
search_base = o= City University, st=New York ,c=US
search_filter = uid=USERNAME
```

In the above example, if a user named **johndoe** is trying to authenticate against the LDAP server, the PDS will perform LDAP searches on the base **City University**, and will filter the results where **uid= johndoe**.

Note that multiple `search_base` and `search_filter` pairs can be defined. For example, to search in both student and faculty trees, you can define the following:

```
search_base = ou=fac,o=sunynp
search_filter = cn=USERNAME
search_base = ou=stu,o=sunynp
search_filter = cn=USERNAME
```

The PDS supports references from LDAP servers. The system tries to bind against the defined LDAP server and follows a reference to another LDAP server, when provided.

In the following `auth_only` example, no attributes are returned—only an `<auth>Y</auth>`, based on the `dn` parameter:

```
[GENERAL]
host_name = cityuniv.library.edu
port      = 389
ldap_version = 3
auth_only=Y
dn= cn=USERNAME,o=Citytown University,c=US
[END]
```

In the following `bor_info_only` example, no bind is performed on the object found by the LDAP search.

```
[GENERAL]
host_name = cityuniv.library.edu
port      = 389
search_base = ou=fac,o=sunynp
search_filter = cn=USERNAME
bor_info_only = Y
[END]
```

When working in the secure mode (LDAP over SSL), enter a flag in the LDAP configuration file, as follows:

```
[GENERAL]
host_name = cityuniv.library.edu
port      = 636
search_base = o =City University, st=Ohio, c=US
search_filter = uid=USERNAME
secure_ldap=Y
[END]

[XML_SETTING]
xml_root_node = bor_authentication
[END]

[ATTRIBUTES_MAPPING]
cn = user_name
mailLocalAddress = email_address
PortalName = portal_name
portalName,EBSCO = z303-birth-date,19910101
{END}
```

Attribute Mapping and Defaults

NOTE:

Underscores and hyphens are interchangeable in the calling application update, but not in the mapping tables.

The tag names in the LDAP XML need to be mapped to names that the calling application recognizes. (For a full list of tags that are currently recognized by

each calling application, see [Appendix D: The bor_info.tags Attribute Mapping Table](#).)

For example, the LDAP code can return:

```
sn: Orange
cn: Becky Orange
mailLocalAddress: becky.orange@exlibris.co.il
PortalName: EBSCO
```

The [ATTRIBUTES_MAPPING] section normalizes the names and can also be used to map values and add defaults. For example, the LDAP `cn` is mapped to the field `user_name`. This is name normalization for the calling application's update program. An example of mapping values is the last line of the attributes mapping below, the `z303-birth-date,19910101`, which is appended to the XML when the input contains a `PortalName` whose value is `EBSCO`. The [DEFAULTS] section example adds an `expiry-date` containing the date `20051010` if there is no `expiry-date` field in the incoming XML.

```
[ATTRIBUTES_MAPPING]
cn = user_name
mailLocalAddress = email_address
PortalName = portal_name
portalName,EBSCO = z303-birth-date,19910101
[END]

[DEFAULTS]
expiry-date,20051010
[END]
```

Remote CGI Hook – Manual Configuration

The PDS provides a custom hook to enable an interface with user-defined external authentication servers for authentication, and user-defined sources for user attributes. The hook is enabled through `remote_cgi_hook.pl` or `remote_cgi_hook_ssl_1.pl`. Both programs are included in the PDS package.

An institution that wants to use this method should write a CGI program that interacts with the PDS hook, as specified below, and handles regular or secure transactions/communication between the PDS and the relevant authentication server. For information on the target attributes to be used in this CGI program, see [Appendix C: Ex Libris Calling Application Target Attributes](#).

CGI Hook Setup

To set up the PDS to work with a remote hook for authentication and user attribute retrieval, you must configure the `AUTHENTICATE` and `BOR_INFO` services in an institution's `tab_service.<institute>` file. The HTTP command syntax to be used is `GET` or `POST` and the HTTP address is to a user-defined CGI script.

For example:

```
[AUTHENTICATE]
program      = remote_cgi_hook.pl
params      = GET,10.1.235.39:8997,aleph-cgi/
remote_hasharon.pl
[END]

[BOR_INFO]
program      = remote_cgi_hook.pl
params      = GET,10.1.235.39:8997,aleph-cgi/
remote_hasharon.pl
[END]
```

CGI Hook Interface

The `remote_cgi_hook.pl` receives the address of the user CGI script from the `tab_service.<institute>` file (the parameters of the AUTHENTICATE or BOR_INFO service) and adds to it the following parameters from the login page:

- `id` – user ID
- `verification` – user password
- `institute` – user institution

The following is an example of an authentication request:

```
http://10.1.235.39:8997/aleph-cgi/
remote_hasharon.pl?BOR_ID=RINA&VERIFICATION=RINA&institute=REMOTE
CGI
```

Communication Between the CGI Program and the PDS

The CGI program communicates with the PDS hook in the following way:

- **Input** – accepts a user's ID, password (optional when the hook is used to retrieve attributes), and institution
- **XML output:**
 - For authentication – sends an authentication flag (Y/N) back to the PDS. Optionally, it may also provide user attributes.
 - For user attribute retrieval – sends the user's attributes back to the PDS.

The communication between the CGI program and the PDS is based on the HTTP protocol and XML.

NOTE:

It is possible for each institution to use its own CGI program.

You should use the `remote_cgi_hook_ssl_1.pl` program if your Apache server supports SSL for secure communication between the PDS and your CGI program.

The CGI program should return an XML document containing the mandatory fields included in the following table:

Table 8. CGI Mandatory Fields

Field	XML Tag	Note
Authenticated	<code><auth></code> <code></auth></code>	Y - authenticated N - not authenticated
User ID	<code><id></code> <code></id></code>	

NOTE:

This is a partial list. The full list of fields can be found in [Appendix C: Ex Libris Calling Application Target Attributes](#).

The following are examples of replies when the CGI program is used to authenticate a user:

```
<?xml version="1.0" ?>
<bor_authentication>
  <auth>Y</auth>
</bor_authentication>

Or:

<?xml version="1.0" ?>
<bor_authentication>
  <auth>N</auth>
</bor_authentication>
```


The following is an example of a reply when the CGI program is used to retrieve user attributes:

```
<?XML version="1.0" encoding="UTF-8"?>
<bor_info>
<id>9036860</id>
<institute>Harvard</institute>
<group>GRAD</group>
<name>Smith, John</name>
<open-date>00000000</open-date>
<name>Smith, John</name>
<title>Mr.</title>
<academic_status></academic_status>
<address_0>10 Tree street </address_0>
<address_1></address_1>
<address_2></address_2>
<address_3></address_3>
<zip>43123</zip>
<email_address>smith@john.com </email-address>
<telephone>+441253656880</telephone>
<bor_info>
```

NOTE:

In MetaLib, a staff user testing the results of the program can view the resulting user record via the MetaLib Management interface (/M). The first time a user logs in, a user record is created in the local calling application's user file, with the information provided via the CGI program. Thereafter, every time a user logs in, the user's details are updated.

Local Login Page Display

As part of the local login process, the PDS searches for and displays the login HTML page in the following manner:

- 1 If there is a request for the login page to be displayed in a specific language, the PDS searches for a login page file that matches the institute and the requested language (/institute-<institute name>/login.<language extension>) and displays this login page if found.
- 2 If there is no language request or if there is no language-specific login page file, the PDS searches for a login page file that matches the institution only (/institute-<institute name>/login) and displays this login page if found.
- 3 If the PDS cannot find the login page file that matches the institute, the PDS searches for a login page file that matches the calling application and the requested language (/calling_system-<calling application name>/login.<language extension>) and displays this login page if found.

- 4 If there is no language request or if there is no language-specific login page file, the PDS searches for a login page file that matches the calling application only (/calling_system-<calling application name>/login) and displays this login page if found.
- 5 If the PDS cannot find the login page file that matches the calling application, it searches for and displays the global login page (/global/login).

NOTE:

For details on login page customization, see **Customizing the Institution Login Page** on page 93 and **Language-Specific Login Pages** on page 93.

The following is an example of a global login page (such as <http://hostname:port/pds>):



Figure 18: Ex Libris Global PDS Login Page

The following is an example of a MetaLib PDS login page, displayed if `calling_system=metalib` was sent in the URL:

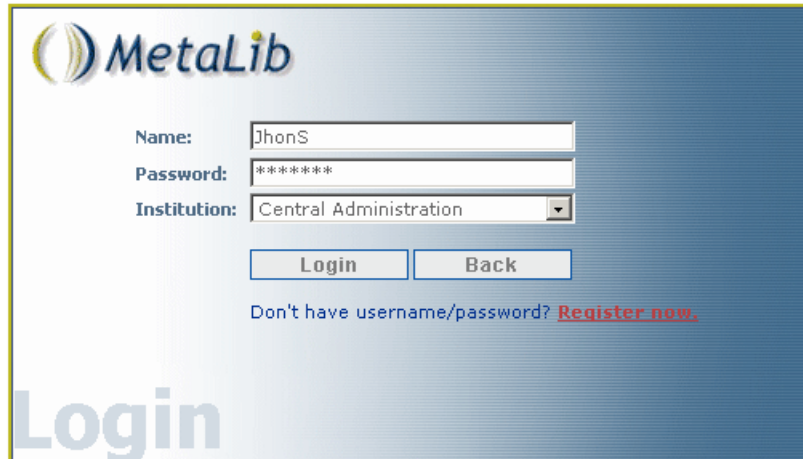


Figure 19: MetaLib PDS Login Page

The following is an example of a DigiTool PDS login page, displayed if `calling_system=digitool` was sent in the URL:

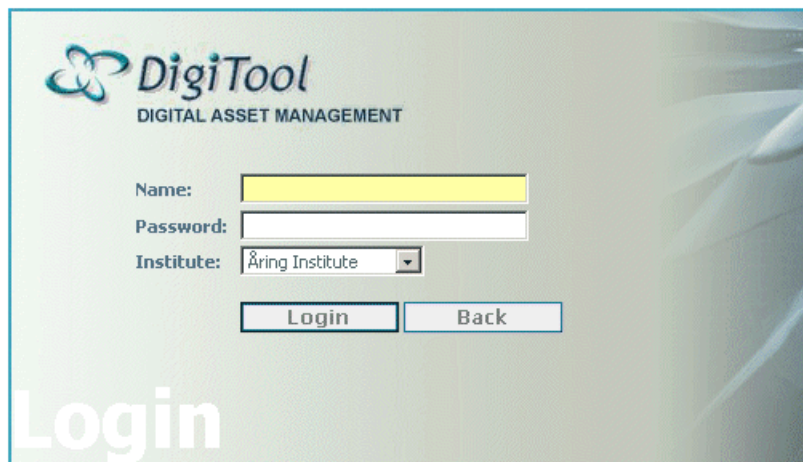


Figure 20: DigiTool PDS Login Page

In addition, a new error message, **Missing PIN**, should be added to the `./conf_table/heading_error.eng` file, as follows:

```
0006 L Missing PIN
```

NOTE:

If the authentication is performed against a non-Voyager system (for example, Shibboleth/LDAP) and user attribute retrieval is performed against Voyager, the authentication system should send a `voyager_ldap_pin` attribute, in addition to the `voyager_ldap_id` and `voyager_ldap_pass`, so that the PDS can later send this PIN attribute to Voyager for user attribute retrieval. The PDS should be configured to map this attribute. (For information on attribute mapping, see **Overview of Attribute Mapping** on page 102.)

Customizing the Institution Login Page

You can create a customized institution login page for each institution.

To create a customized institution login page (example for institution=ARTS):

- 1 Create a new directory for the institution using the following commands:

```
pdsroot
cd html_form
mkdir institute-arts
cp calling_system-<application name>/* institute-arts
```

- 2 Specify `login` as the file to be customized in the newly created directory:

```
./pds/html_form/institute-arts/login
```

After you have performed the above procedure and the ARTS institution is selected on the login page, when you select **View > Source** on the login page, you will see that the login HTML file was taken from the `institute-arts` directory. The file is `$TMPDIR/utf_files/pds/html_form/institute-arts/login`.

Language-Specific Login Pages

To create language-specific login pages, do one of the following:

- If you want the PDS to be able to open HTML pages that match the calling institution and the requested language, in each `institute-<inst_name>` directory, create new login HTMLs (or other PDS HTML pages) with language extensions—for example, `login.fre`, `login.heb`. When the PDS

receives a request (for example, load-login) from an application, it looks for a file that matches the `institute` and `lang` parameters and displays it.

- If you want the PDS to be able to open HTML pages that match the calling application and the requested language, in each `calling_system-<calling_system_name>` directory, create new login HTMLs (or other PDS HTML pages) with language extensions—for example, `login.fre`, `login.heb`. When the PDS receives a request (for example, load-login) from an application, it looks for a file that matches the `calling_system` and `lang` parameters and displays it.

NOTE:

As explained in the above section, the `institute` parameter has a higher priority than the `calling_system` parameter. This means that if there are both the `institute` and `calling_system` directories, the PDS will search for the necessary language-specific login HTML file in the `institute` directory first and, if the file is found, will ignore the `calling_system` directory (as shown in the example below).

For example, if the calling application sends the following parameters:

- `institute=xxx`
- `calling_system=aleph`
- `lang=fre`

the following flow occurs:

- 1 The PDS searches for `./html_form/institute-xxx/login.fre`.
- 2 If not found, the PDS searches for `./html_form/institute-xxx/login`.
- 3 If not found, the PDS searches for `./html_form/calling_system-aleph/login.fre`.
- 4 If not found, the PDS searches for `./html_form/calling_system-aleph/login`.
- 5 If not found, the PDS searches for `./html_form/global/login`.

When the user provides an incorrect user/password, the login page is displayed again with the appropriate language and displays a language-based error message. In order to implement the language-based error message, a new file `heading_error.<lang>` including the appropriate messages should be added to the `conf_table` directory.

Checking the Displayed Login Page File

After the PDS displays the login page, you can check which HTML file the PDS is displaying and where it was found.

To check the login page file name and location:

From the browser menu, select **View > Source**. The HTML source of the PDS login page is displayed.

At the top of the HTML source is a comment with the full path to the file:

```
<!-- The file is : $TMPDIR/utf_files/pds/html_form/  
calling_system-aleph/login -->
```

The HTML files are converted to UTF using the `pds_utf_prog` definition in `PDSDefinitions`:

```
our ($pds_utf_prog) = "/exlibris/calling application/  
xn_n/aleph/exe/pds_utf_file_name";
```

The output is stored in the `./tmp/utf_files/pds` directory.

For example, if the file is `html_form/calling_system-aleph/login` and the `$TMPDIR` is `/exlibris/aleph/m3_1/tmp`, the UTF file will be located in `/exlibris/aleph/m3_1/tmp/utf_files/pds/html_form/calling_system-aleph/login`.

NOTE:

If the UTF file exists, it is used. Thus, if you make changes in the HTML files or install new code that changes values in the HTML, the UTF files should be deleted using the following command: `> rm -r $TMPDIR/utf_files/pds/html_form/`

7

User Attribute Retrieval and Attribute Mapping

This section includes:

- [Overview of User Attribute Retrieval](#) on page 97
- [Configuring User Attribute Retrieval Methods Using the PDS Configuration Wizard](#) on page 98
- [Configuring User Attributes Manually](#) on page 101
- [Overview of Attribute Mapping](#) on page 102
- [Mapping User Attributes Using the PDS Configuration Wizard](#) on page 102
- [Mapping User Attributes Manually](#) on page 105
- [User Attribute Retrieval – Requests by Calling Applications](#) on page 107
- [User Attribute Retrieval with Voyager](#) on page 108

Overview of User Attribute Retrieval

After a successful authentication, the calling application requests user attributes. The PDS can be configured to obtain user attributes either from the calling application's user database or from external user directories. Attributes retrieved are handled in an XML format, mapped, and normalized. The attribute mapping process and configuration are described in [Overview of Attribute Mapping](#) on page 102.

The PDS handles the formatting of the attributes, but the calling application is responsible for interpreting these attributes and granting appropriate authorizations in the calling application.

To configure methods for obtaining user attributes:

- If you are working with the local disk PDS topology, you can configure methods for obtaining user attributes via the configuration wizard, using the instructions in **Configuring User Attribute Retrieval Methods Using the PDS Configuration Wizard** below, or manually, using the instructions in **Configuring User Attributes Manually** on page 101.
- If you are working with the shared Oracle database PDS topology, you must configure methods for obtaining user attributes via the configuration wizard, using the instructions in **Configuring User Attribute Retrieval Methods Using the PDS Configuration Wizard** below.

NOTES:

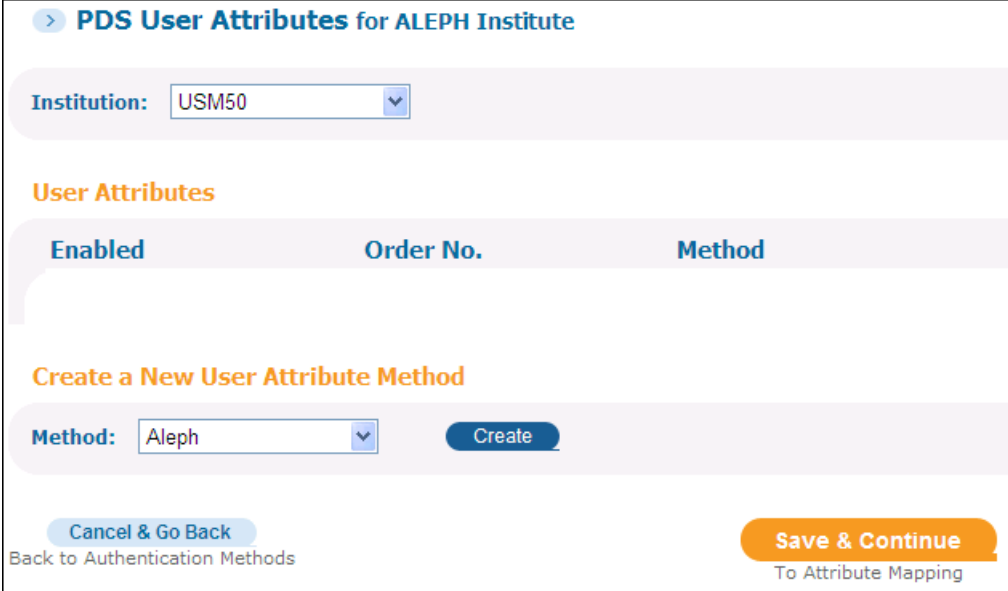
- Calling applications can use different PDS services to retrieve user attributes. For information on the PDS services that can be used by different calling applications, see **User Attribute Retrieval – Requests by Calling Applications** on page 107.
 - If you are working with Voyager, you can choose to have Voyager request user attributes using the LDAP authentication credentials. For details, see **User Attribute Retrieval with Voyager** on page 108.
-

Configuring User Attribute Retrieval Methods Using the PDS Configuration Wizard

If you are working with the PDS configuration wizard, you configure user attribute retrieval methods using the PDS User Attributes page.

To create a user attribute retrieval method using the configuration wizard:

- 1 Click **Save & Continue** on the PDS Authentication Methods page.
The PDS User Attributes page opens.



The screenshot shows the 'PDS User Attributes for ALEPH Institute' configuration page. At the top, there is a breadcrumb '> PDS User Attributes for ALEPH Institute'. Below this, the 'Institution' is set to 'USM50' in a dropdown menu. The main section is titled 'User Attributes' and contains a table with columns 'Enabled', 'Order No.', and 'Method'. Below the table, there is a section 'Create a New User Attribute Method' with a 'Method' dropdown set to 'Aleph' and a 'Create' button. At the bottom, there are two buttons: 'Cancel & Go Back' (with a link 'Back to Authentication Methods') and 'Save & Continue' (with a link 'To Attribute Mapping').

Figure 23: PDS User Attributes

- 2 From the **Method** drop-down list, select one of the following user attribute retrieval methods: **Aleph**, **MetaLib**, **Voyager**, **DigiTool**, **LDAP**, **CGI Hook**, or **CGI Hook SSL-1**.

NOTE:

z312 is not an available option.

- 3 Click **Create**. The new user attribute retrieval method displays under User Attributes.

- 4 Click **Edit** to customize the user attribute retrieval method to suit your system's needs. The PDS - Configure page displays the selected user attribute retrieval method.

Configuration of Aleph User Attributes

Host name/IP:

Port:

Operation code: BOR_AUTH

Admin library:

Use secure: Do not use secure
 Use secure

Service name:

Service password:

Encrypt service password: Yes
 No

To User Attributes

To User Attributes
Note: Will be saved in file only when User Attributes is saved

Figure 24: PDS Configure Page - User Attribute Retrieval Method (Example)

- 5 Edit the fields according to the method on which your user attribute retrieval method is based:
 - If the user attribute retrieval method is based on an internal user attribute retrieval method, such as Aleph, MetaLib, DigiTool, or Voyager, enter the information as described in [Table 4](#) on page [69](#) and in the figure above the table.
 - If the user attribute retrieval method is based on the LDAP external user attribute retrieval method, enter the information as described in [Table 5](#) on page [71](#) and in the figure below the table.
 - If the user attribute retrieval method is based on CGI hook or CGI hook SSL, enter the information as described in [Table 6](#) on page [75](#) and in the figure below the table.
- 6 Click **Save & Continue** to save your modifications and return to the User Attributes page.

Your user attribute retrieval method appears in the list of user attribute retrieval methods. It is recommended that you use the instructions in [Testing User Attribute Retrieval Methods](#) below to ensure that the user attribute retrieval method you configured functions properly.

Testing User Attribute Retrieval Methods

It is recommended that you test the newly created user attribute retrieval method to ensure that it is functioning properly.

To test a user attribute retrieval method:

- 1 In the User Attributes section, click **Test** next to the user attribute retrieval method you want to test.

The Test for User Attribute Method window opens.

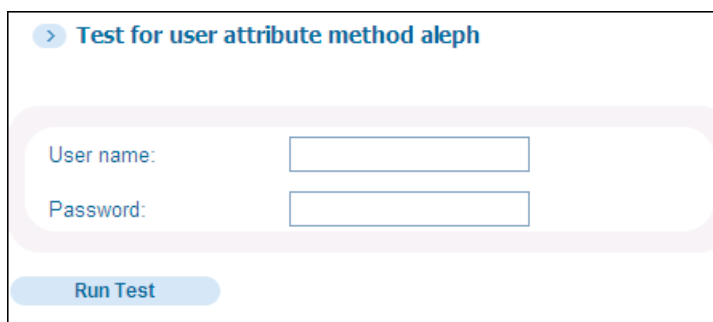


Figure 25: Test for User Attribute Method Window

- 2 In the **User name** and **Password** fields, enter the user name and password with which you log in to the institution.
- 3 Click **Run Test** to test the user attribute retrieval method.

A separate page displays with the results of the user attribute retrieval method test. If the test is successful, click **Save & Continue** on the User Attributes page. If the test was not successful, you can try correcting the user attribute retrieval method information using the instructions in the previous section, or you can call your local Ex Libris representative for support.

Configuring User Attributes Manually

If you are configuring authentication methods manually, you must configure the BOR_INFO service in each institution's `tab_service.<institute>` file. The programs and parameters to be used for this service are the same as those used for the AUTHENTICATE service. For a detailed explanation of the AUTHENTICATE service, see [Configuring Authentication Methods Manually](#) on page 76.

NOTE:

Appendix B: tab_service.<institute> Configurations contains an example of a BOR_INFO service configuration for each calling application.

Overview of Attribute Mapping

Information is delivered to the PDS in many formats and it is the responsibility of the PDS to normalize all the different types of information into a format that the calling application can handle. For example, normalization makes it possible for the PDS to receive local attributes with tag names containing the user table's prefix (such as z303 for Aleph, z312 for DigiTool and MetaLib, and ubid for Voyager), and to send global attributes without these prefixes to the calling applications.

The PDS comes with a set of mappings for all Ex Libris applications. (To view the default set of mappings, see the sections for each calling application in **Appendix D: The bor_info.tags Attribute Mapping Table**.) In many cases, this mapping is sufficient and no additional mapping is required. However, you may need to create new mappings for one of the following reasons:

- You need to override the default mapping of a source attribute/value to the calling application attribute/value.
- You want to add default attributes and values (for example, expiry-date = today).

To map user attributes:

- If you are working with the local disk PDS topology, you can map user attributes via the configuration wizard, using the instructions in **Mapping User Attributes Using the PDS Configuration Wizard** below, or manually, using the instructions in **Mapping User Attributes Manually** on page 105.
- If you are working with the shared Oracle database PDS topology, you must map user attributes via the configuration wizard, using the instructions in **Mapping User Attributes Using the PDS Configuration Wizard** below.

Mapping User Attributes Using the PDS Configuration Wizard

If you are working with the PDS configuration wizard, you map user attributes using the PDS - Attribute Mapping page.

To create a user attribute mapping using the configuration wizard:

- 1 Click **Save & Continue** on the PDS User Attributes page.

The PDS - Attribute Mapping page opens.

Enable	Calling Application	Source Attribute	Calling Application Attribute
Add New Mapping			

[Cancel & Go Back To User Attributes](#) [Save & Go Back To User Attributes to test your configuration](#) [Save & Continue To Institution Configuration](#)

Figure 26: PDS - Attribute Mapping Page

- 2 Click **Add New Mapping** to add a new mapping row.
The new attribute mapping appears in the Attribute Mapping section.
- 3 Select **Enable** to enable the mapping.
- 4 From the **Calling Application** drop-down list, select the relevant Ex Libris calling application—**MetaLib**, **Aleph**, **Primo**, **Voyager**, or **DigiTool**. This enables you to map several calling applications for the same institution. The PDS first searches for a specific calling application mapping and only if this is not found, does it look for a general institution mapping.

NOTE:

This option can be used only if you are working with the shared Oracle database PDS topology. If you are working with the local disk PDS topology, skip this step.

- 5 In the **Source Attribute** field, enter the source user attribute as it appears in the user database that is accessed during the user attribute retrieval process (for example, the Aleph user database or the user database accessed by the CGI program you are using).
- 6 From the **Calling Application Attribute** drop-down list, select the calling application attribute to which you want to map the source user attribute. This is the attribute that the PDS will return to the calling application.

NOTE:

To add a new default attribute, leave the **Source Attribute** field empty and select the required calling application attribute from the **Calling Application Attribute** drop-down list.

Enable	Calling Application	Source Attribute	Calling Application Attribute	
<input checked="" type="checkbox"/>	All	bor-status	z312-group	Edit
<input checked="" type="checkbox"/>	Metalib	z303-home-library	group_id	Edit
<input checked="" type="checkbox"/>	Aleph		expiry-date	Edit

Buttons: Add New Mapping, Cancel & Go Back (To User Attributes), Save & Go Back (To User Attributes to test your configuration), Save & Continue (To Institution Configuration)

Figure 27: New Attribute Mapping

- 7 If you want to map a specific value of a source user attribute to a specific value of a calling application attribute, in the Attribute Mapping section, click **Edit** next to the relevant user attribute mapping.

The PDS - Edit Attribute Mappings page opens.

- 8 Click **Add New Attribute Value**. A new entry appears in the Edit Attribute Values list.
 - c In the **Source Value** field, enter the value of the source attribute that you want to map to a specific value of the calling application attribute to which the source attribute is mapped.
 - d In the **Calling Application Value** field, enter the specific value of the calling application attribute to which you want to map the specified source attribute value.

For example, if you want the PDS to return a value of **STAFF** to the calling application if the value of the source attribute is **15**, enter **15** in the **Source Value** field and **STAFF** in the **Calling Application Value** field.

Enable	Source Value	Calling Application Value
<input checked="" type="checkbox"/>	01	UNDERGRAD
<input checked="" type="checkbox"/>	08	GRAD
<input checked="" type="checkbox"/>	15	STAFF

Buttons: Add New Attribute Value, Cancel & Go Back (To Attribute Mapping), Save & Continue (To Attribute Mapping)

Note: Will be saved in file only when Attribute Mapping is saved

Figure 28: PDS - Edit Attribute Mapping Page

NOTE:

To add a new default value, leave the **Source Value** field empty and type the required calling application value in the **Calling Application Value** field.

- e Click **Save & Continue** to save your settings and return to the PDS - Attribute Mapping page.
- 9 On the PDS - Attribute Mapping page, click **Save & Continue** to save your user attribute mapping settings.

Mapping User Attributes Manually

NOTE:

For information on LDAP user attribute mapping, see **Attribute Mapping and Defaults** on page 85.

You map user attributes manually by creating a custom mapping file (for each institution) named `INSTITUTE_calling_application.tags` or `INSTITUTE.tags` in the `./pds/conf_table` directory.

NOTES:

- If an `INSTITUTE.tags` file is found, the `INSTITUTE_calling_application` file is ignored.
 - It is recommended that you use the naming convention `<Institute code>.tags` (for example, `HUJI.tags`).
-

The `INSTITUTE_calling_application.tags` or `INSTITUTE.tags` file should contain only overrides and additions to the `pds/program/conf/bor_info.tags` file, which is a system file that translates local XML tags to global XML tags and contains the standard normalizations required between all the calling applications. For more information on the `bor_info.tags` file, see **Appendix D: The `bor_info.tags` Attribute Mapping Table**.

IMPORTANT:

The `bor_info.tags` file should not be modified, as changes made to it will be overridden by updates and fixes made to the PDS.

You can include the following in the `INSTITUTE_calling_application.tags` or `INSTITUTE.tags` file:

- Field-to-field mapping – Fields that are not already mapped in the `bor_info.tags` file or whose mapping you want to override should be listed in the `[ATTRIBUTES_VALUES_MAPPING]` section of the

`INSTITUTE_calling_application.tags` or `INSTITUTE.tags` file. This section consists of two columns divided by an equals (=) sign and lets you map values to be translated from the local XML to the global XML. For example:

```
[ATTRIBUTES_VALUES_MAPPING]
    bor-status = z312-group
[END]
```

- Value-to-value mapping – This is a conditional mapping, which means that based on an existing field and value, an additional field and value will be added to the global XML returned to the calling application. This type of mapping should also be included in the `[ATTRIBUTES_VALUES_MAPPING]` section of the `INSTITUTE_calling_application.tags` or `INSTITUTE.tags` file. It should consist of two columns (name, value pairs) divided by an equals (=) sign. The left-hand side should contain the local attribute name and value as defined in the local XML and the right-hand side should contain the global attribute name and value to be returned to the calling application in the global XML. For example:

```
[ATTRIBUTES_VALUES_MAPPING]
    z305-bor-status,01 = group,STAFF
[END]
```

NOTES:

- The string-matching process is not case-sensitive.
 - Value-to-value mapping works on the input before the `bor_info.tags` field-to-field mapping. Thus, the mapping in the above case must be `z305-bor-status,01=group,STAFF` and not `bor-status,01=group,STAFF`.
-
- New fields and values – You can add new fields and values to the global XML by adding these fields and values to the `[DEFAULTS]` section of the `INSTITUTE_calling_application.tags` or `INSTITUTE.tags` file. This section should consist of an attribute name and value, separated by a comma. For example:

```
[DEFAULTS]
    portal-name, SCIENCE
[END]
```

In the above example, if there is no field called `<portal-name>` in the local XML, a field called `portal-name` with the value of `SCIENCE` will be added to the global XML.

For example, if the local XML contains the following:

```
- <bor-info>
  <z305-bor-status>01</z305-bor-status>
</bor-info>
```

It will be converted to the following in the global XML:

```
- <bor-info>
  <bor-status>01</bor-status>
  <group>STAFF</group>
  <portal-name>SCIENCE</portal-name>
</bor-info>
```

NOTE:

The two additional entries are modified according to the attribute mapping rules defined. The `z305-bor-status` is changed to `bor-status` by the `bor_info.tags` file.

For date fields, there is a dynamic default mechanism. The input is: `today +ny +nm +nd`, where `n` is any number, `y` is years, `m` is months, and `d` is days. Any or all months, days, and years can be used in any order and spaces do not matter. For example:

```
[DEFAULTS]
  expiry_date, today+30d
[END]
```

IMPORTANT:

Only fields with values and fields that match a column on the left side of the `bor_info.tags` file will appear in the final global XML.

User Attribute Retrieval – Requests by Calling Applications

A calling application can request user attributes from the PDS using two different services: `BOR_ID` and `BOR_INFO`. In the first case, the user attributes are sent based on the configuration used—for example, the institution selected by the user when the user signs in to the PDS. In the second case, the user attributes can be based on user attribute mappings. This second option has an advantage if the user's institution is derived from the authentication system

from which user attributes are retrieved and if the different institutions share the same configuration (if, for example, the institutions share the same authentication system). In this case, multiple institutions can share the same configuration and the user does not have to select an institution when signing in to the PDS.

Currently, most, but not all, Ex Libris products support the BOR_INFO option, as delineated in the following table:

Table 9. Supported Services

Product	Service	PDS Institution Used For
Primo	BOR_INFO	Primo institution
Aleph	BOR_ID	Aleph ADM
MetaLib version 3.x	BOR_ID	MetaLib institution
MetaLib version 4.x	BOR_INFO	MetaLib institution
DigiTool	BOR_INFO (or like BOR_INFO. BOR_ID is not returned, but since there is no institution, there must always be a mapping)	Can be used for user group in attribute mapping
Voyager	BOR_INFO	Voyager database

NOTE:

If you require SSO with Aleph, you must have a separate configuration for each Aleph ADM. Otherwise, you can have a single configuration file for all institutions/databases and map each institution/database based on user attributes.

User Attribute Retrieval with Voyager

If you are working with Voyager, you can choose to have Voyager request user attributes using the LDAP authentication credentials rather than the credentials entered by the user during initial login.

If you are working with the local disk PDS topology, you can configure this option for Voyager either manually or using the PDS configuration wizard. If you are working with the shared Oracle database PDS topology and want to configure this option for Voyager, you must do so using the configuration wizard.

To configure the use of LDAP credentials for Voyager user attribute retrieval using the configuration wizard:

- 1 Access the PDS General Attributes page for the relevant institution. (See **Configuring Institutions Using the PDS Configuration Wizard** on page 26 for instructions on creating institutions.)
- 2 Under **Use LDAP attributes for Voyager BOR_INFO**, select **Yes**.

The screenshot shows the 'PDS General Attributes for' configuration page. At the top, there is a dropdown menu for 'Institution'. Below this, the 'PDS General Attributes' section contains several fields: 'Code' and 'Description' are text input fields; 'Character conversion' is a dropdown menu set to 'NONE'; 'Calling application & override code' includes checkboxes for 'Primo', 'MetaLib', 'Aleph', 'Rosetta', 'Voyager', and 'DigiTool', each with an associated text input field. The 'Remote login' section has radio buttons for 'Use remote Login' and 'Do not use remote Login' (selected), with a 'Method' dropdown set to 'Choose method'. The 'Logout from PDS' section has checkboxes for 'Redirect logout' and 'Remote logout', each with a text input field. The 'Credentials for application logout service' section has 'Service user name' and 'Service password' text input fields. The 'Remote SSO' section has radio buttons for 'Use remote SSO' and 'Do not use remote SSO' (selected), with a 'Method' dropdown set to 'Choose method'. The 'EZproxy' section has radio buttons for 'Use EZproxy' and 'Do not use EZproxy' (selected). The 'Use LDAP attributes for Voyager BOR_INFO' section has radio buttons for 'Yes' (selected) and 'No'. The 'Change institution display order' section has a dropdown menu set to 'select'. At the bottom, there are two buttons: 'Cancel & Go Back' (with a link 'To Institution Configuration') and 'Save & Continue' (with a link 'To Authentication Methods').

Figure 29: Use LDAP Attributes for Voyager BOR_INFO

3 Click **Save & Continue** to save your settings.

To manually configure the use of LDAP credentials for Voyager user attribute retrieval:

In the relevant `tab_service.<institute>` file, add the following lines:

```
[BOR_ID_FROM_LDAP]
params = Y
[END]
```

Note that you must also map the relevant LDAP attributes to `voyager_ldap_id` and `voyager_ldap_pass`. For information on doing this using the configuration wizard, see [Table 5](#) on page 71. The following is an example of a manual attribute mapping:

```
[ATTRIBUTES_MAPPING]
    cn = user_name
    mail = email_address
    employeenumber = voyager_ldap_pass
    sn = voyager_ldap_id
[END]
```

8

Logout Configuration

This section includes:

- [Overview of Logout Configuration](#) on page 111
- [Configuring Logout Using the PDS Configuration Wizard](#) on page 112
- [Configuring Logout Manually](#) on page 113

Overview of Logout Configuration

When an authenticated user chooses to log out of an application, a logout request is sent from the calling application to the PDS. The logout process removes the `pds_handle` that was created for the user and expires the user's session.

If you want to direct the user to a specific URL after logout, you can configure the **redirect logout** option.

If you have configured remote login using iChain, CAS, or Shibboleth and want to configure the user's logout from the remote authentication server (so that the next user to log in to the calling application is not logged in as the previous user), ensure that you configure the following:

- for iChain, the **redirect logout** option
- for CAS, the **remote logout** option
- for Shibboleth, one of the following:
 - the **remote logout** option (or `SHIB_LOGOUT` in the `tab_service.<institute>` file)
 - the mapping of the Shibboleth logout attribute to the PDS logout attribute

NOTE:

For details of the Shibboleth logout workflow, see [The Shibboleth Logout Workflow](#) on page 140.

If you are working with the local disk PDS topology, you can configure logout options via the configuration wizard, using the instructions in [Configuring Logout Using the PDS Configuration Wizard](#) on page 112, or manually, using the instructions in [Configuring Logout Manually](#) on page 113. If you are working with the shared Oracle database PDS topology, you must configure logout options via the configuration wizard, using the instructions in [Configuring Logout Using the PDS Configuration Wizard](#) on page 112.

NOTE:

For information on configuring remote Single Sign-Off (SSO) across Ex Libris products, see [Configuring SSO Using the PDS Configuration Wizard](#) on page 45 or [Configuring SSO Manually](#) on page 47.

Configuring Logout Using the PDS Configuration Wizard

If you are working with the PDS configuration wizard, you configure logout options using the PDS General Attributes page.

To configure logout options using the configuration wizard:

- 1 Access the PDS General Attributes page for the institution for which you want to configure logout options. (See [Configuring Institutions Using the PDS Configuration Wizard](#) on page 26 for instructions on creating institutions.)
- 2 Under **Logout from PDS**, select one or both of the following options:
 - **Redirect logout** – In the text box, enter the specific calling application HTML page or URL to which you want to redirect the user—for example, `http://www.cnn.com` or `Y/?func=find` (within the application). If you are working with an iChain remote authentication server and want to ensure that the next user who attempts to log in will not be logged in as the previous user, enter `http://<iChain server>:<port>/<iChain logout API>`.
 - **Remote logout** – If you are working with a CAS remote authentication server and want to configure the user's logout from this server so that the next user to log in to the calling application is not logged in as the previous user, enter the following in the text box:
 - For CAS: `https://<CAS server>:<port>/<CAS logout API>`

- For Shibboleth: `http://<Shibboleth server>:<port>/<Shibboleth logout API>`

NOTE:

For information on mapping the Shibboleth logout attribute to the PDS logout attribute, see **Configuring Shibboleth Using the Configuration Wizard** on page 134.

Figure 30: Redirect/Remote Logout Configuration

- 3 Click **Save & Continue**.

Configuring Logout Manually

To manually redirect a user to a specific calling application HTML page or another URL after the user has logged out, configure the REDIRECT_LOGOUT service in each relevant institution's `tab_service.<institute>` file.

For example:

```
[REDIRECT_LOGOUT]
params = Y/?func=find
[END]
```

or:

```
[REDIRECT_LOGOUT]
params = http://www.cnn.com
[END]
```

In the first example above, when a user logs out of the calling application, the user is redirected to the Find module, as specified in the `params` link.

NOTE:

When redirecting to an HTML page within the application, there is no need to enter the host name and port.

In the second example above, when a user logs out of the calling application, the user is redirected to <http://www.cnn.com>, as specified in the `params` field.

iChain Logout

If you are working with an iChain remote authentication server and want to redirect a user to a specific calling application HTML page or another URL after the user has logged out and ensure that the next user who attempts to log in will not be logged in as the previous user, configure the following in each relevant institution's `tab_service.<institute>` file:

```
[REDIRECT_LOGOUT]
params = http://<iChain_server>:<port>/<iChain logout API>
[END]
```

CAS Logout

If you are working with a CAS remote authentication server and want to configure the user's logout from this server so that the next user to log in to the calling application is not logged in as the previous user, configure the following in each relevant institution's `tab_service.<institute>` file:

```
[REMOTE_LOGOUT]
params = https://<cas_server>:<port>/<CAS logout API>
[END]
```

CAS 3.0 Logout

If you are using CAS 3.0, the following steps must be executed for the logout process.

- 1 Add `?service=` to the `REMOTE_LOGOUT` url.

```
[REMOTE_LOGOUT]
params = https://<cas_server>:<port>/<CAS logout
API>?service=
[END]
```

- 2 In the `httpd.conf` file, change `PerlSetVar CASSessionCookieName` from `APACHECASORACLE` to `APACHECAS`.
- 3 Create a customized `redirect-remote-logout` and add the following line after line 17:
 - `url = url.replace(/.url=/, "");`

Shibboleth Logout

If you are working with a Shibboleth remote authentication server and want to configure the user's logout from this server so that the next user to log in to the calling application is not logged in as the previous user, configure the `[REMOTE_LOGOUT]` or `[SHIB_LOGOUT]` section in each relevant institution's `tab_service.<institute>` file, as follows:

```
[REMOTE_LOGOUT]
params = http://<Shibboleth server>:<port>/<Shibboleth logout API>
[END]
```

```
[SHIB_LOGOUT]
params = http://<Shibboleth server>:<port>/<Shibboleth logout API>
[END]
```

NOTE:

For information on mapping the Shibboleth logout attribute to the PDS logout attribute, see [Configuring Shibboleth Manually](#) on page 137.

Part V

Remote Authentication Programs

This part contains the following sections:

- **Section 9: CAS and iChain Authentication** on page 119
- **Section 10: Shibboleth** on page 129

9

CAS and iChain Authentication

This section includes:

- **CAS Authentication** on page 119
- **iChain Authentication** on page 125

CAS Authentication

The PDS provides a “hook” for CAS (Central Authentication System). The PDS host’s Apache server needs to be configured to add restricted resources (one resource per institution) and to add a Perl module that communicates with the CAS server.

Because the interface is a `mod_perl` handler, which keeps sessions in the database, you must also perform additional configuration steps to activate this option.

To configure CAS authentication:

- 1 Add the following to the `htdocs/oracle_error.html` file:

```
<html><head>
<title>Cas Error</title>
</head><body>
<h1>Internal CAS Error</h1>
<p>Cannot process request: The service you are attempting to
access is denied
</p>
<p>More information about this error may be available
in the server error log.</p>
</body></html>
```

- 2 Ensure that the `$(application name)_dev/cas.sql` file includes the following lines. (If the file doesn't exist, create it.):

```
CREATE TABLE cas_sessions(  
  id      varchar(32) not null primary key,  
  last_accessed number not null,  
  "uid"   varchar(32) not null,  
  pgtiou  varchar(64) not null  
);  
  
CREATE TABLE cas_pgtiou_to_pgt (  
  pgtiou  varchar(64) not null primary key,  
  pgt     varchar(64) not null,  
  created number not null  
);
```

- 3 Create the CAS user and tables using the following commands.

- For Aleph, MetaLib, and DigiTool:

```
dlib vir00  
$(application name)_proc/create_ora_user_b cas  
  
sqlplus cas/cas @cas.sql
```

- For Primo:

```
dlib prm00  
$(application name)_proc/create_ora_user_b cas  
  
sqlplus cas/cas @cas.sql
```

NOTE:

You may need to install a patch. To determine whether you must install a patch, run the following:

```
perl -MDBD::Oracle -e 'print "OK\n"
```

This command should return an OK. If it does not, you will require a patch. Contact Ex Libris Support for details.

- 4 Create the restricted CAS directories (two in the case of multiple CAS authentication) using the following commands:

```
apch
mkdir cas1
mkdir cas2
cd cas1
ln -s ${application name}_dev/pds/program/pds_main pds_main
cd cas2
ln -s ${application name}_dev/pds/program/pds_main pds_main
```

- 5 In the Apache httpd.conf file:

- a Enter the following:

```
<Location /cas1>
    SetHandler perl-script
    PerlResponseHandler ModPerl::Registry
    Options +ExecCGI
    PerlOptions +ParseHeaders

    PerlSetVar isMetalib "true"
    PerlSetVar PDS_USER_NAME "pds_id"
    PerlSetVar PDS_PAGE_KEY "user"
    PerlSetVar PDS_INSTITUTE "castest"

    PerlSetVar CASHost "xxx.xxx.edu"
    PerlSetVar CASPort "443"
    PerlSetVar CASErrorURL "http://xx.xx.xx:8331/oracle_error.html"
    # For Oracle this is the SID
    PerlSetVar CASDatabaseName "dev"
    PerlSetVar CASSsl "[Y/N]"
    PerlSetVar CASProxyCheck "[Y/N]"
    PerlSetVar CASDatabaseHost "ic-dev.xxx.xx"
    PerlSetVar CASDatabasePort "nnnn"
    # DBI is case sensitive in UNIX environments so be careful.
    PerlSetVar CASDatabaseDriver "Oracle"
    PerlSetVar CASDatabaseUser "xxxx"
    PerlSetVar CASDatabasePass "yyyy"
    # Remember to have a different cookie for each instance
    PerlSetVar CASSessionCookieName "APACHECASORACLE"
    PerlSetVar CASSessionTimeout "1800"
    PerlSetVar CASLogLevel "0"
    PerlSetVar CASRemoveTicket "false"
    # to change the default login page from /cas/login to /sso/login
    # PerlSetVar CASLoginURI /sso/login
    AuthType Apache::AuthCASEXL
    AuthName "CAS"
    PerlAuthenHandler Apache::AuthCASEXL->authenticate
    require valid-user
</Location>
<Location /cas2>
    ...
```

- b** Update the CASHost and CASPort with the CAS host name and port number for the site:

```
PerlSetVar CASHost "xxx.xxx.edu"  
PerlSetVar CASPort "443"  
PerlSetVar CASSsl "[Y/N]"
```

- c** Update the CASDatabaseName with the Oracle SID and port number:

```
PerlSetVar CASDatabaseHost "ic-dev.xxx.xx"  
rt  
PerlSetVar CASDatabasePort "nnnn"
```

- d** Update the CASDatabaseUser and CASDatabasePass with the user name and password for CAS:

```
PerlSetVar CASDatabaseDriver "Oracle"  
PerlSetVar CASDatabaseUser "xxxx"  
PerlSetVar CASDatabasePass "yyyy"  
</Location>
```

- e** If you are using CAS for proxy authentication, set CASProxyCheck to `y`. If CASProxyCheck is set to `Y`, the PDS checks the syntax of the CAS user authentication information. If it starts with `ST-`, the PDS uses service validation to authenticate the user. Otherwise, it uses proxy validation to authenticate the user.
- 6** In the `httpd.conf` file, update the `ServerName` parameter with the PDS hostname in the following format:
 - `http(s)://<hostname>:<port>`
 - 7** If you are configuring CAS authentication for more than one institution:
 - a** Create a sub-directory in the `html_form` directory for each institution. For example:
 - `institute-cas1/`
 - `institute-cas2/`
 - b** Copy the following HTML files from the `html_form_global` directory to each one of the new directories:
 - `redirect-remote-cas`
 - `redirect-remote-cas-sso`
 - c** Configure each HTML page to match the corresponding CAS:
 - `<body onload = "location = '/goto/&server_httpd/cas1/pds`
 - `<body onload = "location = '/goto/&server_httpd/cas2/pds`
 - 8** If you are configuring CAS authentication using the PDS configuration wizard (required if you are working with the shared Oracle database

topology), access the PDS General Attributes page for the institution for which you want to configure remote login (see [Configuring Institutions Using the PDS Configuration Wizard](#) on page 26 for instructions on creating institutions), select **Use remote login** under **Remote login**, select **CAS** from the **Method** drop-down list, and click **Save & Continue**.

The screenshot shows the 'PDS General Attributes for USM50' configuration page. At the top, the 'Institution' is set to 'USM50'. Below this, the 'PDS General Attributes' section contains the following fields:

- Code:** USM50
- Description:** ALEPH Institute
- Character conversion:** NONE
- Calling application & override code:** A list of checkboxes for Primo, MetaLib, Aleph, Rosetta, Voyager, and DigiTool, each followed by an empty text input field.
- Remote login:** 'Use remote Login' is selected with a radio button. The 'Method' dropdown menu is set to 'CAS'. 'Do not use remote Login' is unselected.

Figure 31: Selecting the CAS Remote Login Method

If you are configuring CAS authentication manually, access the relevant institution's `tab_service.<institute>` file and enter `cas.pl` in the `[LOAD_LOGIN]` section's program line.

```
[LOAD_LOGIN]
program      = cas.pl
params      =
[END]
```

- 9 To use the CAS server to determine whether a user is already logged in (SSO):
 - If you are configuring CAS authentication using the PDS configuration wizard (which you must use if you are working with the shared Oracle database topology), access the PDS General Attributes page for the institution for which you want to configure remote SSO, select **Use remote SSO** under **Remote SSO**, select **CAS** from the **Method** drop-down list, and click **Save & Continue**.

- If you are configuring CAS authentication manually, access the relevant institution's `tab_service.<institute>` file and enter `cas.pl` in the [LOAD_SSO] section's `program` line and `sso` in the `params` line.

```
[LOAD_SSO]
program      = cas.pl
params      = sso
[END]
```

If the user is not logged in, the user is redirected back to the PDS as a guest.

- 10 Optionally, configure logout from the CAS server, as described in [Logout Configuration](#) on page 111.

NOTE:

The CAS interface provides authentication only, and only the ID is stored. User attribute retrieval must be set up using another method, such as LDAP.

After CAS authentication has been configured, the Apache server is activated and the user is redirected to a CAS login page. After the authentication completes the session, `pds_main?func=load-login` is accessed.

The PDS checks the environment for an `HTTP_PDS_ID` variable. If the variable exists, the authentication succeeds and a `PDS_HANDLE` is returned to the calling application.

EZproxy Authentication

In version 5.1a and later, EZproxy can act as a CAS service and can work with the PDS.

To configure EZproxy to act as a CAS service:

- If you are configuring EZproxy using the PDS configuration wizard (which you must use if you are working with the shared Oracle database topology), access the PDS General Attributes page for the institution for which you want to configure EZproxy, select **Use EZproxy** under **EZproxy**, and click **Save & Continue**.
- If you are configuring EZproxy manually, add the following line to the EZproxy `config.txt` file:

```
CASServiceURL WildcardServiceURL
```

where `wildcardServiceURL` is the PDS server name or any other application to which you want to give access to the CAS service.

For example:

```
CASServiceURL http://casapp.yourlib.org/*
```

iChain Authentication

There are two options for implementing iChain with the PDS:

- Form fill – iChain intercepts the PDS login page that is sent to the user, enters the user’s credentials, and sends the response back to the PDS. This process, whereby the iChain imitates the PDS, is seamless to the user. Upon receiving information, the PDS performs the login process and logs the user in to the calling application.
- A restricted resource, `/pds_ichain`, is set up in the `http.conf` file and is also defined as a `ScriptAlias` to `pds/program/pds_main`. The load-login process calls `/pds_ichain` instead of `/pds`. The Apache redirects the user to the iChain login page and then continues with the load-login PDS functionality.

To instruct the Apache to redirect the user, you must add the following rewrite rules to the Apache `httpd.conf` file:

```
RewriteEngine On
RewriteCond %{HTTP:Authorization} ^(.*)
RewriteRule ^/(.*)$ - [env=HTTP_AUTHORIZATION:%1]
```

NOTE:

iChain allows for SSO, login, and logout. There is no need for external scripts to perform these tasks.

Although iChain provides authentication only and user attribute retrieval must be handled using another method, such as LDAP, the user’s user name and password are extracted from iChain and stored so that they can be accessed later by the PDS.

If the user logs out of iChain, the calling application recognizes this and logs the user out. If the user logs out of the calling application, you can configure the **redirect logout** option to log the user out of iChain as well. If the user does not log out of iChain, the next user who attempts to log in will be logged in as the previous user.

To configure iChain authentication:

- If you are configuring iChain authentication using the PDS configuration wizard (required if you are working with the shared Oracle database topology), access the PDS General Attributes page for the institution for which you want to configure remote login (see [Configuring Institutions Using the PDS Configuration Wizard](#) on page 26 for instructions on creating institutions), select **Use remote login** under **Remote login**, select **iChain** from the **Method** drop-down list, and click **Save & Continue**.

> PDS General Attributes for USM50

Institution: USM50

PDS General Attributes

Code: USM50

Description: ALEPH Institute

Character conversion: NONE

Calling application & override code:

<input type="checkbox"/>	Primo	
<input type="checkbox"/>	MetaLib	
<input type="checkbox"/>	Aleph	
<input type="checkbox"/>	Rosetta	
<input type="checkbox"/>	Voyager	
<input type="checkbox"/>	DigiTool	

Remote login: Use remote Login Do not use remote Login Method: iChain

Figure 32: Selecting the iChain Remote Login Method

- If you are configuring iChain authentication manually, access the relevant institution's `tab_service.<institute>` file and enter `ichain.pl` in the `[LOAD_LOGIN]` section's `program` line.

```
[LOAD_LOGIN]
program      = ichain.pl
params      =
[END]
```

To use iChain to determine whether a user is already logged in (SSO):

- If you are configuring iChain authentication using the PDS configuration wizard (which you must use if you are working with the shared Oracle database topology), access the PDS General Attributes page for the

institution for which you want to configure remote SSO, select **Use remote SSO** under **Remote SSO**, select **iChain** from the **Method** drop-down list, and click **Save & Continue**.

- If you are configuring iChain authentication manually, access the relevant institution's `tab_service.<institute>` file and enter `ichain_sso.pl` in the `[LOAD_SSO]` section's `program` line and `sso` in the `params` line.

```
[LOAD_SSO]
program      =  ichain_sso.pl
params       =  sso
[END]
```

If the user is not logged in, the user is redirected back to the PDS as a guest.

NOTE:

You can configure logout from iChain. For information, see [Logout Configuration](#) on page 111.

10

Shibboleth

This section includes:

- **Overview of Shibboleth** on page 129
- **Prerequisites for Working with Shibboleth** on page 130
- **Additional Shibboleth Apache Configuration** on page 131
- **Configuring Shibboleth to Work with the PDS** on page 133
- **The Shibboleth Login Workflow** on page 139
- **The Shibboleth Logout Workflow** on page 140
- **The Shibboleth SSO Workflow** on page 141
- **Configuring Shibboleth as a WAYF** on page 141

Overview of Shibboleth

This chapter outlines the prerequisites and guidelines for configuring Shibboleth to work with the PDS in conjunction with Ex Libris products. The PDS supports Shibboleth 2.1 and later with Apache version 2.2.xx provided by Ex Libris.

NOTE:

Third-party products, such as Shibboleth, are not under the control of Ex Libris and Ex Libris is not responsible for any changes or updates to Shibboleth. Furthermore, Ex Libris is not responsible for providing any end-user support with respect to Shibboleth. For further information about Shibboleth, refer to the appropriate Web site (<http://shibboleth.internet2.edu/>).

Prerequisites for Working with Shibboleth

To implement Shibboleth with the PDS, you must first set up and configure the appropriate environment or have access to such an environment. An appropriate environment contains:

- A Shibboleth identity provider
- Shibboleth service provider software built and configured to work with the Shibboleth identity provider. The service provider software must be located on the same server as each PDS instance.

NOTE:

When you set this up, the PDS plays the role of a WAYF server. The PDS first presents patrons with a list of institutions from which to select for login. If a patron chooses a non-Shibboleth institution, the PDS presents a login page and handles the login. If a patron chooses a Shibboleth institution, the PDS redirects the patron to a Shibboleth service provider that links directly to one of the Shibboleth institution's identity providers. The service provider allows the identity provider to handle authentication, and then retrieves the appropriate attributes. The service provider then returns these attributes to the PDS in a format that the PDS can translate. In this scenario, there is a separate service provider for each identity provider that is handled by the PDS. For details, see **Configuring Shibboleth as a WAYF** on page 141.

- Configuration of Shibboleth on the product's Apache server
- A URL for performing logout from the identity provider (optional)

This section describes:

- **Tips for Installing the Shibboleth Service Provider** on page 130
- **Verifying That Prerequisites Are Met** on page 131

Tips for Installing the Shibboleth Service Provider

Ex Libris installs its own Apache server on the system. First, you need to make sure that this Apache server supports dynamic loading of shared modules (`mod_so` compiled in; check with `httpd -l`). You then need to make sure that Shibboleth is using the same shared libraries (`.so`) as the Apache server, which is compiled against its own OpenSSL libraries. If Shibboleth is not using the same shared libraries as the Apache server, segfaults of the Apache children will appear at startup.

It is highly recommended to obtain the latest Shibboleth SP package suitable for your operating system from the Shibboleth Web site. The following are some useful commands:

- `$ ldd /exlibris/metallib/m4_1/product/local/apache/bin/httpd`
- `$ /exlibris/product/httpd-2.0.xx/bin httpd -l`
- `$ ldd /exlibris/product/httpd-2.0.xx/modules/mod_ssl.so`
- `$ ldd /path/to/shibboleth-sp/libexec/mod_shib_20.so`

NOTE:

If the Apache server (and the supplied OpenSSL) does not use the same libraries as Shibboleth, contact Ex Libris Support for instructions on upgrading your Apache server.

Verifying That Prerequisites Are Met

To verify the prerequisite settings:

- 1 Log the user in to the Shibboleth identity provider. Using the address bar of your Web browser, type the following HTTP request:
`http://<product host>:<product port>/shib`
A login page of your identity provider is launched.
- 2 Enter a valid Shibboleth user name and password and confirm that you can log in to Shibboleth.

NOTE:

It is highly recommended to run the service provider on HTTPS.

Additional Shibboleth Apache Configuration

This section describes:

- [Adding a Symbolic Link in the /apache/htdocs Directory](#) on page 132
- [Editing the Apache Configuration File](#) on page 132
- [Modifying <application name>_start](#) on page 133

Adding a Symbolic Link in the /apache/htdocs Directory

You must create a separate institution subdirectory in the /shib directory, under `htdocs`, that has a symbolic link pointing to the `pds_main` program in the /pds/program/ directory. This directory is protected by Shibboleth, so the PDS uses the Shibboleth attributes after authentication.

To add a symbolic link:

Create the symbolic link using the following commands:

```
apch
mkdir shib
cd shib/
  mkdir $inst
  cd $inst
    ln -s <pdsroot>/program/pds_main pds_main
```

To confirm the creation of the symbolic link:

Run the following command:

```
ls -l
```

The result appears as follows:

```
lrwxrwxrwx  1 <product>  exlibris      43 Nov 22 08:18
pds_main -> <pdsroot>/program/pds_main*
```

Editing the Apache Configuration File

The Apache configuration file needs to be edited so that it regards `pds_main` as an executable program.

To edit the Apache configuration file:

- 1 Open the `httpd.conf` or `ssl.conf` file using the following commands.

```
apcc
cp httpd.conf httpd.conf.<date>
vi httpd.conf
```

NOTE:

If you are using SSL, use `ssl.conf` instead of `httpd.conf`, where applicable.

- 2 Edit the configuration file within a VirtualHost context.

NOTE:

For RequestMap to function properly, the VirtualHost definition must match the definition of the Host in the RequestMap section of the shibboleth2.xml file.

- 3 Edit the following commands so that they are appropriate for your setup. (At the very least, edit the SHIB_ROOT command.)

```
# you may replace /etc/shibboleth with the shibboleth
software dir
Include /etc/shibboleth/apache2.config

<Location /shib>
  AuthType shibboleth
  ShibRequireSession On
  #it might be smart to modify the next line:
  require valid-user
  Options ExecCGI FollowSymlinks
  ForceType application/x-httpd-cgi
</Location>
```

Modifying <application name>_start

In many cases, it is not necessary to add the Shibboleth lib directory to the library search path. However, you are advised to do so nevertheless.

To add the Shibboleth lib directory to the library search path:

Add the following lines to <application name>_start:

```
# set shib env variables
setenv SHIB_CONFIG SHIBROOT/shibboleth/etc/shibboleth/
shibboleth.xml
setenv SHIB_SCHEMAS SHIBROOT/shibboleth/etc/shibboleth
setenv LD_LIBRARY_PATH ${LD_LIBRARY_PATH}:SHIBROOT/
shibboleth/lib
```

Configuring Shibboleth to Work with the PDS

After the above prerequisites have been successfully fulfilled, you can configure the PDS to work with Shibboleth.

- If you are working with the local disk PDS topology, you can implement Shibboleth via the configuration wizard, using the instructions in [Configuring Shibboleth Using the Configuration Wizard](#) below, or manually, using the instructions in [Configuring Shibboleth Manually](#) on page 137.

- If you are working with the shared Oracle database PDS topology, you must implement Shibboleth via the configuration wizard, using the instructions in [Configuring Shibboleth Using the Configuration Wizard](#) below.

Configuring Shibboleth Using the Configuration Wizard

Using the configuration wizard, you can configure Shibboleth:

- to authenticate users and retrieve user attributes
- to determine whether a user is already logged in (SSO)

To configure Shibboleth using the configuration wizard:

- 1 Access the PDS General Attributes page for the institution for which you want to configure remote login/SSO (see [Configuring Institutions Using](#)

the **PDS Configuration Wizard** on page 26 for instructions on creating institutions).

> **PDS General Attributes for**

Institution:

PDS General Attributes

Code:

Description:

Character conversion:

Calling application & override code:

<input type="checkbox"/> Primo	<input type="text"/>
<input type="checkbox"/> MetaLib	<input type="text"/>
<input type="checkbox"/> Aleph	<input type="text"/>
<input type="checkbox"/> Rosetta	<input type="text"/>
<input type="checkbox"/> Voyager	<input type="text"/>
<input type="checkbox"/> DigiTool	<input type="text"/>

Remote login: Use remote Login Method:
 Do not use remote Login

Logout from PDS: Redirect logout
 Remote logout

Credentials for application logout service: Service user name:
Service password:

Remote SSO: Use remote SSO Method:
 Do not use remote SSO

EZproxy: Use EZproxy
 Do not use EZproxy

Use LDAP attributes for Voyager BOR_INFO: Yes
 No

Change institution display order:

Cancel & Go Back
To Institution Configuration

Save & Continue
To Authentication Methods

Figure 33: PDS General Attributes Page

- 2 Select one or both of the following options:
 - Under **Remote login**, select **Use remote login**, select **Shibboleth** from the **Method** drop-down list, and click **Edit Shibboleth**.
 - Under **Remote SSO**, select **Use remote SSO**, select **Shibboleth** from the **Method** drop-down list, and click **Edit Shibboleth**.

The PDS - Edit Remote Login/SSO page opens to allow you to map the Shibboleth attributes to the attributes that the calling application recognizes.

> PDS - Edit Remote Login for

Attributes for Remote Login Method "Shibboleth"

Source Attribute	Source Value	PDS Attribute	PDS Value
------------------	--------------	---------------	-----------

Create

Cancel & Go Back
To PDS General Attributes

Save & Continue
To PDS General Attributes
Note: Will be saved in file only when PDS General Attributes is saved

Figure 34: PDS - Edit Remote Login Page

- 3 Click **Create** to add a new attribute mapping.
A new attribute mapping row displays in the list of attributes.

> PDS - Edit Remote Login for

Attributes for Remote Login Method "Shibboleth"

Source Attribute	Source Value	PDS Attribute	PDS Value
<input checked="" type="checkbox"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>

Create

Cancel & Go Back
To PDS General Attributes

Save & Continue
To PDS General Attributes
Note: Will be saved in file only when PDS General Attributes is saved

Figure 35: Add Attribute Mapping Row

- 4 In the **Source Attribute** field, enter the name of the Shibboleth attribute you want to map (for example, **HTTP_SHIB_BORSTATUS** or **SHIB_LOGOUT**).
- 5 (Optional) In the **Source Value** field, enter a specific value of the Shibboleth attribute that you want to map to a calling application attribute or attribute value (for example, **MEDLINE**).

- 6 From the **PDS Attribute** drop-down list, select the calling application attribute to which you want to map the Shibboleth attribute you defined (for example, **academic_status** to map to **HTTP_SHIB_BORSTATUS** or **logout_url** to map to **SHIB_LOGOUT**).

NOTE:

To be able to save user information (for example, favorites) you must map the appropriate Shibboleth attribute to `z312_source_id`, the PDS ID attribute.

- 7 (Optional) In the **PDS Value** field, enter a specific value of the calling application attribute to which you want to map the Shibboleth attribute or Shibboleth attribute + value that you defined (for example, **portal_name**, **MEDLINE** to map to **group**, **STAFF**).
- 8 To enable the mapping you defined, ensure that the **Enabled** check box is selected.
- 9 Repeat steps 3 to 8 for each attribute that you want to map to the PDS.
- 10 Click **Save & Continue** to save your settings and return to the PDS General Attributes page.
- 11 Click **Save & Continue** on the PDS General Attributes page.

Configuring Shibboleth Manually

To configure Shibboleth manually, you must configure each relevant `tab_service.<institute>` file as well as the `shib.conf` file.

To configure Shibboleth manually:

- 1 In each relevant `tab_service.<institute>` file:
 - a Configure the following services (the code must be in uppercase letters):

```
[LOAD_LOGIN]
program      = shib.pl
params      = shib.conf
[END]
[BOR_INFO]
program     = z312.pl
[END]
[INSTITUTE_DISPLAY]
code       = <unique code of institute>
lang      = ENG
desc      = inst-description
[END]
```

b (Optional) Set up the logout service:

```
[SHIB_LOGOUT]
params      = http://<server>:<port>/<Shibboleth logout API>
[END]
```

NOTE:

The SHIB_LOGOUT configuration must include a link to the customer's identity provider logout URL, if an identity provider is being used. If not, the PDS logs out of the service provider and Ex Libris products only.

c Enter the following lines to optionally set up remote SSO:

```
[LOAD_SSO]
program     = shib_sso.pl
params     = shib.conf
[END]
```

d Enter the following lines to optionally set the character conversion:

```
[CONVERT_SHIB_ATTRIBUTE_TO_CHARSET]
params = ISO-8859-1
[END]
```

2 To complete the optional LOAD_SSO configuration, add the following lines to the `general_conf` (or `sso_conf`) file:

```
[DEFAULT_INSTITUTE]
cookie = _shibsession_(.*)
[END]
```

3 Create a `shib.conf` file in the `./pds/conf_table` directory. Base your file on the following guidelines and the example below of a `shib.conf` file.

- The file must be customized according to your institution's attributes.
- To be able to save user information (for example, favorites) you must map the appropriate Shibboleth attribute to `z312_source_id`, the PDS ID attribute.
- Fields can also be specified to receive default values.
- Dates can be configured according to context (see example).

The following is an example of a `shib.conf` file:

```
[SHIB_ATTRIBUTES]
REMOTE_USER = z312_source_id
HTTP_SHIB_PERSON_COMMONNAMELIMSNAME = z312_name
HTTP_SHIB_BORSTATUS = z312_academic_status
[END]

[ATTRIBUTES_MAPPING]
portal_name,MEDLINE = group,STAFF
[END]

[DEFAULTS]
expiry_date,today+1y
[END]
```

- 4 Add the path to the program directory in the `pds_main` using the following command:

```
use lib "$ENV{<product>_dev}/pds/program";
```

- 5 (Optional) The PDS can use a URL sent by the Shibboleth identity provider for logout. To instruct the PDS to do so, map the Shibboleth logout URL attribute to the PDS logout URL attribute in the `shib.conf` file. For example:

```
SHIB_LOGOUT = logout_url
```

NOTE:

The attribute sent by Shibboleth may have a name other than `SHIB_LOGOUT`, depending on the identity provider setup.

If you do not perform this mapping, the default logout is performed (that is, the PDS uses the `SHIB_LOGOUT` or `REMOTE_LOGOUT` definition, or alternatively, the `REDIRECT_LOGOUT` definition, in the `tab_service.<institute>` file).

The Shibboleth Login Workflow

This section describes the processes Shibboleth performs when a user logs in.

- 1 The user attempts to log in and chooses the institute to which the user belongs.
- 2 The Shibboleth program is started.
- 3 The PDS checks whether the user is already logged in to Shibboleth.

- If so, the PDS creates a new session and redirects the user back to the calling application.
 - If not, the PDS redirects the user to the Shibboleth identity provider login page. After the user logs in, the Shibboleth identity provider authenticates the user and sends the user's attributes to the PDS. The PDS creates a new session and writes the user attributes to the local disk or Oracle database for use during the user attribute retrieval process. Note that the attributes are mapped according to the configured Shibboleth attribute mapping.
- 4 The calling application sends a user attribute retrieval request to the PDS. The PDS retrieves the user attributes from the PDS local disk or Oracle database and sends an XML response to the calling application.

The Shibboleth Logout Workflow

This section describes the processes Shibboleth performs when a user logs out.

- 1 The user attempts to log out of the calling application.
- 2 The calling application redirects the logout request to the PDS.
- 3 The PDS checks whether the user is logged in to Shibboleth. If not, the PDS expires the user's session and redirects the user back to the calling application. If the user is logged in to Shibboleth:
 - If the Shibboleth logout attribute was mapped to the PDS logout attribute (either via the configuration wizard or in the `shib.conf` file) and the PDS received a logout URL (for example, `SHIB_LOGOUT = http: idp_ xxx`) from Shibboleth at login, the PDS redirects the user to this URL for logging out.
 - If the above conditions are not met, the PDS looks for the Shibboleth logout or remote logout parameter in the institution's configuration and redirects the user to this URL for logging out.

NOTE:

At this point, logout is performed only from the Shibboleth service provider session and not from the PDS session.

- 4 The identity provider expires its session and sends a logout request back to the PDS.
- 5 The PDS logs the user out of the PDS session and redirects the user back to the calling application or the redirect logout configured for the institution.

The Shibboleth SSO Workflow

When the remote SSO option is defined for an institution and a user logs in to Shibboleth from an external application and then accesses an Ex Libris application, the PDS checks Single Sign-On against the Shibboleth identity provider.

The SSO workflow is as follows:

- 1 The calling application sends an SSO request to the PDS.
- 2 The PDS checks whether the user is logged in to Shibboleth.
 - If so, the PDS creates a new session and redirects the user to the calling application.
 - If not, the PDS redirects the user to the Shibboleth identity provider with an SSO request. Shibboleth then authenticates the user and sends the user's attributes to the PDS. The PDS creates a new session and writes the user attributes to the local disk or Oracle database for use during the user attribute retrieval process. Note that the attributes are mapped according to the configured Shibboleth attribute mapping.
- 3 The calling application sends a user attribute retrieval request to the PDS. The PDS retrieves the user attributes from the PDS local disk or Oracle database and sends an XML response to the calling application.

Configuring Shibboleth as a WAYF

To configure Shibboleth as a WAYF, you must edit the `shibboleth.xml` file.

To configure Shibboleth as a WAYF:

- 1 After the `<Path name="shib" applicationId="default" requireSession="true">` line in the `shibboleth.xml` file, add a separate mapping line for each institute to be used. Use one of the following syntaxes:
 - `<Path name="Institute" applicationId="Institute" requireSession="true" exportAssertion="true"/>`
 - `<Path name="INSTITUTE" requireSession="true" requireSessionWith="INSTITUTE" exportAssertion="true"/></Path>`

- 2 Add a separate `Application` section for each institute. Each `Application` section must have a unique `handlerURLshireURL` and an identity provider-specific `wayfURL`.

```
<Application id="INSTITUTE">
  <Sessions lifetime="7200" timeout="3600"
  checkAddress="true"
  handlerURLshireURL="/shib/INSTITUTE/Shibboleth.sso"
  handlerSSLshire" shireSSL="false"
  wayfURL=https://IDP-SERVER/shibboleth-idp/SSO/HS/>
</Application>
```

NOTE:

For security reasons, you should run the PDS on HTTPS and set `handlerSSL` to `true`.

Part VI

Security Configuration

This part contains the following section:

- **Section 11: Configuring Security for the PDS** on page 145

11

Configuring Security for the PDS

This section includes:

- [Configuring the PDS to Use SSL](#) on page 145
- [Securing the Configuration Wizard](#) on page 146
- [Optional Security Enhancement for REMOTE_LOGIN and REMOTE_SSO](#) on page 146
- [X-Server Security Patch](#) on page 148
- [Securing the PDS_HANDLE Cookie](#) on page 149

Configuring the PDS to Use SSL

To configure the PDS to use SSL, you must replace all the HTTP lines in your program or PDS definitions with HTTPS.

For example:

- 1 Replace our `($server_pds)= "http://<IP or Domain Name>:<PORT>/pds";` with our `($server_pds)= "https://<IP or Domain Name>:<PORT>/pds";`
- 2 Verify that our `($server_httpsd)= "https://<IP or Domain Name>:<PORT>/pds";` exists and is active.
- 3 Change the `server_httpd` from our `($server_httpd) = "http://<IP or Domain Name>:<PORT>";` to our `($server_httpd)= "https://<IP or Domain Name>:<PORT>";`

NOTE:

For information on configuring the calling application to use SSL, see [Calling Application Configuration](#) on page 37. For additional information on configuring SSL for Primo, refer to the *Primo System Administration Guide*.

Securing the Configuration Wizard

If you are working with the configuration wizard, it is recommended that you secure the wizard.

To secure the configuration wizard:

- 1 Add the following lines to the `/pdsadmin` section of the configuration file:

```
<Location /pdsadmin>
  AddType application/x-httpd-cgi .cgi
  Options +ExecCGI
  AllowOverride None
  Order Deny,Allow
  Allow from all
  AuthType Basic
  AuthName "PDSADMIN"
  AuthUserFile passwd/.htaccess
  require valid-user
</Location>
```

- 2 Define the user name and password that should be used to access the wizard.

Enter the following commands from the command prompt:

```
cd $httpd_bin
./htpasswd -bm $httpd_root/passwd/.htaccess <user name>
<password>
```

The following message is returned:

```
Adding password for user <user name>
```

Optional Security Enhancement for REMOTE_LOGIN and REMOTE_SSO

NOTE:

This section describes the procedure for the security enhancement of a manual remote login and remote SSO configuration. For information on configuring the security enhancement via the PDS configuration wizard, see [Configuring Remote Login Using the PDS Configuration Wizard](#) on page 62 and [Configuring Remote SSO Using the PDS Configuration Wizard](#) on page 52.

If the `params = check`, a token is added as a security enhancement to protect against the following risks:

- **Playback** – The user saves the return URL from the login page and uses it to bypass login at a later date.
- **False identity** – The user alters the ID parameter in the return URL to impersonate another user.

The token is 40 characters in length and contains the following:

- 8 characters – hex version of the time at which the token was generated.
- 32 characters – MD5 hash of the time, the user ID, and a salt value.

The token is created and sent from the user-defined authentication CGI:

```
&PDS_HANDLE=43f1b446d434e1b963c02196b9610bb910a491b1
```

To activate the check in the `remote_login_gen_1.pl`, add the value `check` and the value of the time difference to allow (in seconds) to the parameters.

For example:

```
[REMOTE_LOGIN]
program      = remote_login_gen_1.pl
params      = check,1800
[END]
```

The same changes should be made in the `tab_service` for `REMOTE_SSO`:

```
[REMOTE_SSO]
program = remote_sso_gen_1.pl
params = check,600
[END]
```

The code added to `remote_login_gen_1.pl` does the following:

- Turns the hex time back into a decimal and compares it with the current time. If the difference between the two is greater than the second value in the parameters, it determines that the URL is being “played back.”
- Constructs an MD5 of the time (from the token), the user ID, and the salt and compares it with the MD5 passed in the token. If any single character of the token has been altered (for example, the user attempted to alter the timestamp portion), or if the user manually changed the PDS borrower ID value (that is, the current security hole), the login is not allowed.

The following is the code that encodes the token:

```
my $salt      = 'sadjkhaksjdhjkhkasfhkfhjkhsdfkhasdfkhsdf';
my $time      = time();
my $hex_time  = sprintf("%08x",$time);
my $check     = $hex_time.md5_hex($time, $id, $salt);
```

And in the HTML:

```
"PDS_HANDLE=$check&" +
```

The following is an example of REMOTE_LOGIN input with security enhancement:

```
http://il-metalsfx01.corp.exlibrisgroup.com:13006/  
pds?func=remote-login&  
pds_handle=4a66f205188c983b836fab2ac28849fbcca9224e&  
calling_system=metalib&  
institute=REMOTE&  
id=DEMO&  
url=http://il-metalsfx01.corp.exlibrisgroup.com:8331/V
```

The salt value contains a hard-coded value, but can be configured. In order to set the salt value, add a new line to the `PDSDefinitions` file:

```
our ($pds_salt) = "<SALT_VALUE>";
```

X-Server Security Patch

A security flaw exists that can expose attributes of authenticated users. The flaw causes attributes of an authenticated user to be available in XML using the following URL syntax:

http://10.1.235.47:13002/pds?func=get-attribute&attribute=BOR_INFO&pds_exl_id=dtl01

Manual changes are required in order block access to this restricted content.

To perform these manual changes:

- 1 Map the allowed IP addresses. Add the calling application server IP address to the `$apache_home/conf/hosts-allow` file, as in the following example:

```
10.1.235.47 FOUND
```

- 2 Using `mod_rewrite`, add a restriction in the Apache file so that only the calling application machine can access this link. In the `$apache_home/conf/httpd.conf` section, after the declaration `RewriteEngine On`, add the

following rewrite rules. These are based on the `hosts-allow` file configurations.

```
RewriteMap hosts-allow txt:/exlibris/pds/p1_2/apache/conf/
hosts-allow
RewriteCond %{QUERY_STRING} ^func=get-
attributeandattribute=BOR_INFO [NC]
RewriteCond ${hosts-allow:%{REMOTE_ADDR}|NOT-FOUND} NOT-FOUND
RewriteRule ^.*$ /PDSAccessNotAllowed.html
```

These rules determine that when the `$query_string` contains a `func=get-attribute` and the `$remote_addr` is not found in the `hosts-allow` lookup table, the browser is redirected to a `NotAllowed.html`.

- 3 Create the file `$apache_home/htdocs/PDSAccessNotAllowed.html`. The contents of this file should include the following code:

```
<html>
<head><title>PDS Error</title></head>
<body bgcolor="#ffffff" LEFTMARGIN="0" TOPMARGIN="0"
MARGINWIDTH="0" MARGINHEIGHT="0">
<TABLE CELLPADDING="0" CELLSPACING="0" BORDER="0" width="100%">
<TR bgcolor="#C0DDEA" >
<TD height="27" style="COLOR: #43596B; FONT-SIZE: 70%;FONT-
WEIGHT: BOLD; FONT-FAMILY: TAHOMA,ARIAL,VERDANA; PADDING-LEFT:
12PX;PADDING-RIGHT: 12PX;">Access Denied Error:</TD>
</TR>
</TABLE>
<div style="height:150px;"></div>
<table bgcolor="#C4E0ED" CELLPADDING="1" CELLSPACING="1"
align="center">
<tr>
<td bgcolor="#EEF4F7">
<table CELLPADDING="4" CELLSPACING="4">
<tr>
<td style="color: #FF0022; font-weight: bold; FONT-FAMILY:
VERDANA,TAHOMA,ARIAL;
FONT-SIZE: 70%;">Cannot process request: The service you are
attempting to access is denied</td>
</tr>
</table>
</td>
</tr>
</table></body></html>
```

Securing the PDS_HANDLE Cookie

- The `HttpOnly` flag is added by default to the `PDS_HANDLE` cookie to prevent the risk of the client side script accessing the protected cookie. In order to

remove the `HttpOnly` flag, add the following line to the `PDSDefinitions` file:

```
our ($pds_cookie_http_only) = "N";
```

- There is an option to set the `secure` flag in the `PDS_HANDLE` cookie. To set it, add the following line to the `PDSDefinitions` file:

```
our ($pds_cookie_secure) = "Y";
```

Part VII

Debugging

This part contains the following section:

- **Section 12: Debugging the PDS** on page 153

12

Debugging the PDS

This section includes:

- **PDS Direct Access** on page 153
- **Error Messages from the PDS** on page 156
- **PDS Debug Mode** on page 157

PDS Direct Access

The <http://<host>:<port>/pds> interface to the PDS does not entail a calling application. Through this interface, you can check the authentication and user attribute retrieval functionality without interactions from the calling application. This is mainly a debugging tool. The following sections explain the /pds interface.

The Remote and Local List Page

When the institute parameter is not sent to the PDS and remote login is configured for one of the institutions, the PDS displays a page that contains a list of both remote and local institutions, instead of the login page. For example:



Figure 36: PDS Remote and Local List Page

Main Menu – Logged Out Page

The main menu – logged out page resides in the `./pds/html_form/global` directory. It is displayed when the user accesses the PDS directly via an <http://<host>:<port>/pds> URL.

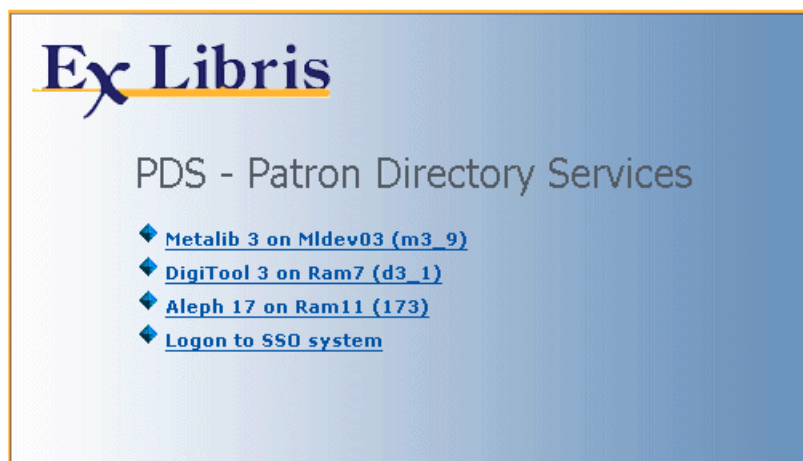


Figure 37: PDS Main Menu – Logged Out Page

The `main-menu-logged-out` file needs to be edited to reflect the choices presented to users logging in to the PDS. This includes a list of the applications sharing the PDS, the `href` links to the appropriate applications, and the display names. The following is an example of two applications defined in the `main-`

menu-logged-out file. Note that the lines in bold below must be edited to configure appropriate local settings and values:

```
PDS - Patron Directory Services $0100
  </TD>
</TR>
<TR STYLE="PADDING:2PX 0PX 6PX 0PX">
  <TD CLASS="LableBoldDark">
    
    <a class= LableBold
href="http://hostname:port/V"
title = "Metalib - 3"
Metalib 3 on Mldev03 (m3_9) </a>
  </TD>
</TR>
<TR STYLE="PADDING:2PX 0PX 6PX 0PX">
  <TD CLASS="LableBoldDark">
    
    <a class= LableBold
href="http://hostname:port/R"
title = "DigiTool 3.0"
DigiTool 3 on Ram7 (d3_1)</a>
  </TD>
```

If a site uses a single Ex Libris application, you can modify the login page so that it links back to the Ex Libris application instead of displaying the above menu. The following is an example of a file defined to link back to Aleph:

```
<!-- main-menu-logged-out -->
<html>
<head>
  <title> Main Menu - Logoff </title>
<include>meta-tags
<link rel="stylesheet" href="andserver_httpd/pds.css"
TYPE="text/css">
</head>
<body onload="top.location = 'http://www.ml-univ.com:8332/F'">

</body>
</html>
```

Automatic Logoff

You can also configure an automatic logoff option.

To activate the auto logoff option:

Add the following line:

```
<body onload="top.location = 'http://www.ml-univ.com:8332/V'">
```

in the `./pds/html_form/calling_system-product/main-menu-logged-out` file.

Main Menu – Logged On Page

The main menu – logged on page resides in the `./pds/html_form/global` directory. It is displayed when the user accesses the PDS directly via an <http://<host>:<port>/pds> URL, after a valid authentication.



Figure 38: PDS Main Menu – Logged On Page

Error Messages from the PDS

The `./pds/conf_table/` directory contains the `heading_error.eng` file, which defines the error messages used by the PDS. You can customize the PDS error messages by editing this file.

The following error messages are currently used for the login process:

- 0001 L Missing Password
- 0002 L Missing ID
- 0003 L Service not defined in tab_service
- 0004 L Invalid UserID and/or Password. Please re-enter.
- 0005 L Invalid institution

The file has three columns:

- The first column defines the error number used by the PDS programs and must not be modified.
- The second column is a language indicator.

- The third column defines the text to be displayed to the end user. This text may be customized.

PDS Debug Mode

In the `pds/program/PDSDefinitions` file there is a debug flag. The default debug mode is **off**:

```
our ($debug) = "N";
```

To change the debug mode to **on**, replace the `N` with `Y`.

The `$LOGDIR/pds_server.log` tracks which PDS programs were run with which parameters and can sometimes serve as a debugging aid.

NOTE:

The PDS logs can be restarted using the `start_w` command. PDS logs that are older than 30 days are automatically removed.

The PDS version runs in `mod_perl`, which means that all the PDS programs are loaded once. To change the debug mode, therefore, you must run the following from the `pdsroot/program` directory, in addition to changing the `PDSDefinitions` file:

```
touch ./PDSUtil.pm  
touch ./PDStouchfile
```

Alternatively, you can run the following from this directory:

```
./pds_debug_off  
./pds_debug_on
```

These additional changes cause the Apache server to reload the changed `PDSDefinitions` file.

NOTE:

There are `pds_debug_on/pds_debug_off` routines in the `program` directory under `pdsroot`, which you can use to turn debugging on and off.

Part VIII

Appendixes

This part contains the following sections:

- **Appendix A: Utilities for Manual Configuration** on page 161
- **Appendix B: tab_service.<institute> Configurations** on page 167
- **Appendix C: Ex Libris Calling Application Target Attributes** on page 175
- **Appendix D: The bor_info.tags Attribute Mapping Table** on page 179
- **Appendix E: Example of the PDSDefinitions File** on page 187

A

Utilities for Manual Configuration

This section includes:

- **PDS Setup Validity Testing Utility** on page 161
- **Encrypt Password Utility** on page 163

PDS Setup Validity Testing Utility

The PDS has a validity testing utility, designed to check the manual setup configuration of the PDS system in order to ensure that definitions are valid and that destinations are accessible.

The validity of the configuration files per institution can be checked by entering the user name, password, and a template file for the institution, or by checking all institutions, based on predefined user names and templates that are stored in a table.

To activate the validity testing:

- For MetaLib, Aleph, or DigiTool, enter the following:

```
cd <calling application proc directory> (For MetaLib/Aleph this is  
aleph_proc or ap. For DigiTool, this is dp.)  
  
pds_check
```
- For Primo, enter the following:

```
cd <pdsroot/program>  
  
pds_check.pl
```

The following menu appears:

```
PDS Check Utility - Main Menu
=====
0. Exit
1. Edit pds_tab_users
2. Check file PDSDefinitions
3. Check tab_service tables for institutes
4. View log files

Select [0]:
```

Explanation of the PDS Check Utility Options

Option 1: Edit pds_tab_users

This option allows you to change information in the `pds_tab_users` file.

The following submenu options exist:

- **Change default template** – applies the general template to all institutes that have no private template
- **Edit institute information** – prompts the user per institute for a user ID, verification, and template. The passwords are encrypted.

Option 2: Check file PDSDefinitions

This option checks the following:

- whether directories defined in `./pds/program/PDSDefinitions` exist
- whether the defined IP addresses can be accessed

Option 3: Check tab_service tables for institutes

If you select this option, you are prompted as follows:

1 Select institute or ALL

- If you select a single institute, you are prompted to enter the following:
 - user ID
 - verification
 - template extension
 - confirmation

- If you select all institutes (the **ALL** option), you are prompted to enter the following:
 - default template extension
 - confirmation
- 2 Verify that all services that appear in the template exist in the `tab_service.<institute>` file.
- 3 Verify that all parameter line services that appear in the template exist in the `tab_service.<institute>` file.
- 4 Verify the following:
 - the presence of the program file in the `service_proc` directory
 - that the parameters are valid
 - that the `function` parameter matches the service
 - that the port is accessible
 - that the code and language are supplied
 - that authentication is obtained with the supplied parameters

Option 4: View log files

Enables you to view a summary of the errors in the log files.

Encrypt Password Utility

The `./pds/program/pds_encrypt.pl` utility encrypts the LDAP `init_bind_password` and the service password to the X-Server session, as well as passwords in existing PDS configurations. This is an optional step, as the `x-server.pl` and `ldap.pl` programs and PDS configurations work in both encrypted and decrypted mode.

The `pds_encrypt.pl` program takes one parameter, `<institute>`, which is the suffix of the `tab_service.<institute>` file.

For example, to update the `tab_service.digitool` table, use the following commands:

```
pdsroot
cd program
perl pds_encrypt.pl digitool
```

The file originally appeared as follows:

```
[AUTHENTICATE]
program      =  digitool.pl
params      =  ram7:8881,op=BOR_AUTH,DAT01,PDS,PDS,N
[END]

[BOR_INFO]
program      =  digitool.pl
params      =  ram7:8881,op=BOR_INFO,DAT01,PDS,PDS,N
[END]
```

After `pds_encrypt.pl` is run, the file appears as follows:

```
[AUTHENTICATE]
program      =  digitool.pl
params      =
ram7:8881,op=BOR_AUTH,DAT01,PDS,!encrypt@G1dX3qXt0cS7nZMuMqOpqz
AIBF8BFsPcjMqhu4TGbm0=,N
[END]

[BOR_INFO]
program      =  digitool.pl
params      =
ram7:8881,op=BOR_INFO,DAT01,PDS,!encrypt@G1RT2KXt0cS7nZMuMqOpqz
AIBF8BFsPcjMqhu4rFZW8=,N
[END]
```

LDAP

If you are working with LDAP, use the following command:

```
perl pds_encrypt.pl <LDAP FILE>
```

For example, if the file originally appeared as follows:

```
[GENERAL]
host_name = il-dc01
port = 389
ldap_version = 3
secure_ldap = N
init_bind_dn = CN=test ldap,OU=Unknown
Mailboxes,OU=Users,OU=Israel,DC=Corp,DC=Exlibrisgroup,DC=com
init_bind_password = testldappass
start_tls = Y
search_base =
OU=Users,OU=Israel,DC=Corp,DC=ExlibrisGroup,DC=Com
search_filter = sAMAccountName=USERNAME
[END]
```

After `perl pds_encrypt.pl <LDAP FILE>` is run, the file appears as follows:

```
[GENERAL]
host_name = il-dc01
port = 389
ldap_version = 3
secure_ldap = N
init_bind_dn = CN=test ldap,OU=Unknown
Mailboxes,OU=Users,OU=Israel,DC=Corp,DC=Exlibrisgroup,DC=com
init_bind_password =
!encrypt@F1JR3aXJ80TG3pI3Lv56lBUTlcJDI7e10a4wtaFeS77109QSRMYUg==
start_tls = Y
search_base = OU=Users,OU=Israel,DC=Corp,DC=ExlibrisGroup,DC=Com
search_filter = sAMAccountName=USERNAME
[END]
```

Oracle

To encrypt the Oracle password in the `PDSDefinitions` file, use the following command:

```
perl pds_encrypt.pl PDSDefinitions
```


B

tab_service.<institute> Configurations

This section includes:

- **Aleph Standard Configuration** on page 168
- **MetaLib Standard Configuration** on page 169
- **DigiTool Standard Configuration** on page 169
- **Alma Standard Configuration** on page 170
- **Voyager Standard Configuration** on page 171
- **LDAP Configuration** on page 171
- **Aleph X-Server Configuration** on page 172
- **Remote CGI Configuration** on page 173

NOTE:

The `tab_service-example.txt` file in the `./pds/conf_table` directory contains an example table of all the sections in the `tab_service.<institute>` file.

Aleph Standard Configuration

The following example shows the `tab_service.science` entries for a new institution named **science**. This institution has decided to use the Aleph authentication service and the Aleph user database to obtain user attributes.

```
[AUTHENTICATE]
program = aleph.pl
params  = <server_name>,<port>, BOR_AUTH,USM50,N,ALEPH,ALEPH
[END]

[BOR_INFO]
program = aleph.pl
params  = <server_name>,<port>, BOR_INFO,USM50,N,ALEPH,ALEPH
[END]

[BOR_VERIFICATION]
program = get_pds_verification.pl
[END]

[INSTITUTE_DISPLAY]
code      = SCIENCE
lang      = ENG
desc      = Science Institute
[END]
```


MetaLib Standard Configuration

The following example shows the `tab_service.science` entries for a new institution named **science**. This institution has decided to use the MetaLib authentication service and the MetaLib user database to obtain user attributes.

```
[AUTHENTICATE]
program = metalib_x_server.pl
params  = <server_name>,<port>,BOR-AUTH,N,pds,pds
[END]

[BOR_INFO]
program = metalib_x_server.pl
params  = <server_name>,<port>,BOR-INFO,n,pds,pds
[END]

[BOR_VERIFICATION]
program = get_pds_verification.pl
[END]

[INSTITUTE_DISPLAY]
code      = SCIENCE
lang      = ENG
desc      = Science Institute
[END]
```

DigiTool Standard Configuration

The following example shows the `tab_service.science` entries for a new institution named **science**. This institution has decided to use the LDAP followed by DigiTool for both authentication and user attributes. If the user name/password is authenticated by the LDAP, the authentication succeeds. If

the LDAP authentication is unsuccessful, the PDS attempts to authenticate against the local DigiTool database.

```
[AUTHENTICATE]
program = ldap.pl
params = ldap_science.conf
program = digitool.pl
params = <server_name>:<port>, op=BOR_AUTH,DAT01,PDS,PDS,N
[END]

[BOR_INFO]
program = ldap.pl
params = ldap_science.conf
program = digitool.pl
params = <server_name>:<port>, op=BOR_INFO,DAT01,PDS,PDS,N
[END]

[INSTITUTE_DISPLAY]
code      = SCIENCE
lang      = ENG
desc      = Science Institute
[END]
```

Alma Standard Configuration

The following example shows the tab_service.science entries for a new institution named **science**. This institution has decided to use the Alma authentication service and the Alma user database to obtain user attributes.

```
[AUTHENTICATE]
program = dps.pl
params = il-urm08.corp.exlibrisgroup.com,1801,BOR-
AUTH,N,EXLDEV1_INST
[END]

[BOR_INFO]
program = dps.pl
params = il-urm08.corp.exlibrisgroup.com,1801,BOR-
INFO,N,EXLDEV1_INST
[END]

[INSTITUTE_DISPLAY]
code      = EXLDEV1_INST
lang      = ENG
desc      = Alma Main Campus
[END]
```

NOTE:

The last parameter in BOR_INFO is optional.

Voyager Standard Configuration

The following example shows the `tab_service.science` entries for a new institution named **science**. This institution has decided to use the Voyager authentication service and the Voyager user database to obtain user attributes.

```
[AUTHENTICATE]
program = voyager.pl
params  = <server_name>:<port>,auth,N
[END]

[BOR_INFO]
program = voyager.pl
params  = <server_name>:<port>,info,N
[END]

[INSTITUTE_DISPLAY]
code      = SCIENCE
lang      = ENG
desc      = Science Institute
[END]
```

LDAP Configuration

The following example shows the `tab_service.law` entries for a new institution named **law**. This institution has decided to use an LDAP server as its authentication method, as well as the LDAP user database to obtain user attributes.

```
[AUTHENTICATE]
program = ldap.pl
params  = ldap_law.conf
[END]

[BOR_INFO]
program = ldap.pl
params  = ldap_law.conf
END

[INSTITUTE_DISPLAY]
code = LAW
lang = ENG
desc = Law University
[END]
```

NOTE:

The PDS LDAP configuration file, `ldap_law.conf` in the `./pds/conf_table` directory, must be configured for this institution.

Aleph X-Server Configuration

The following example shows an institution named **cityuniv** using an X-Server configuration (for Aleph 17) to authenticate users and retrieve user attributes.

```
[AUTHENTICATE]
program = aleph.pl
params = ram19,8081,BOR_AUTHENTICATE,USM50,N,WWW-X,WWW-X
[END]

[BOR_INFO]
program = aleph.pl
params = ram19,8081,BOR_AUTHENTICATE,USM50,N,WWW-X,WWW-X
[END]

[INSTITUTE_DISPLAY]
code = CITYUNIV
LANG = ENG
Desc = City University
[END]
```

The following is an explanation of the Aleph X-Server service's third line, column three data string, ram19,8081,BOR_AUTHENTICATE,USM50,N,WWW-X,WWW-X:

Table 10. Explanation of Data String Parameters

Data String Parameter	Description
ram19	Server name
8081	Port
BOR_AUTHENTICATE	X-Server function/service
USM50	Active library
N	Use non-secure HTTPD (Y=Use secure HTTPS)
WWW-X	X-Server user name
WWW-X	X-Server password

Remote CGI Configuration

The following example shows a new institution named **arts**. This institution uses a remote CGI hook program for authentication and retrieval of user attributes.

```
[AUTHENTICATE]
program = remote_cgi_hook.pl
params  = GET,www.university.edu:8992,cgi-bin/remote_cgi
[END]

[BOR_INFO]
program = remote_cgi_hook.pl
params  = GET,www.university.edu:8992,cgi-bin/remote_cgi
[END]

[INSTITUTE_DISPLAY]
code = ARTS
lang = ENG
desc = Arts University
[END]
```

To customize the `tab_service.<institute>` file:

- 1 Copy the existing file and save it under a new file name.
- 2 Replace ARTS with your local institute.
- 3 Replace `GET,www.university.edu:8992,cgi-bin/remote_cgi` with the appropriate `<Method>`, `<Base-URL>`, `<remote-cgi path>`.

C

Ex Libris Calling Application Target Attributes

This section includes:

- [Overview](#) on page 175
- [MetaLib Target Attributes](#) on page 175
- [Primo Target Attributes](#) on page 176
- [DigiTool Target Attributes](#) on page 176

Overview

This appendix provides a list of target attributes that are used by the Ex Libris calling application. Since Aleph always takes user attributes from Aleph and not via the PDS, Aleph user attributes are not included. A site that writes a CGI hook requires this information to send the user attributes that use these tags.

MetaLib Target Attributes

- portal_name
- academic_status
- address_0
- address_1
- address_2
- address_3
- address_4
- address_5

- bor_id
- con_lng
- email_address
- expiry_date
- group
- id
- institute
- name
- resource_status
- sfx_base_url
- telephone_1
- telephone_2
- title
- zip

Primo Target Attributes

- email_address
- group
- id
- institute
- name

DigiTool Target Attributes

- address_0
- address_1
- address_2
- address_3
- address_4
- address_5
- bor_dept_m
- bor_group_m

- bor_tuples_m
- con_lng
- course_enrollment_m
- email_address
- expiry_date
- group
- name
- profile_id
- telephone_1
- telephone_2
- title
- zip

D

The bor_info.tags Attribute Mapping Table

This section includes:

- [Overview](#) on page 179
- [Aleph Attributes](#) on page 180
- [Voyager Attributes](#) on page 181
- [DigiTool Attributes](#) on page 182
- [MetaLib Attributes](#) on page 182

Overview

The `pds/program/conf/bor_info.tags` table is divided into sections. Each section lists the possible user attributes originating from a defined application, such as Aleph, MetaLib, DigiTool, or Voyager.

Even input that does not come from one of these sources is translated through this standard table.

The table consists of two columns divided by an equals (=) sign. The left column defines the values that the PDS receives from the source of the user attributes (Aleph, LDAP, CGI hook, and so forth) and the right column defines the normalized attribute names that the PDS sends to the calling application (Aleph, MetaLib, DigiTool, or Voyager).

This file handles all user attributes passed through the PDS and should not be changed.

Aleph Attributes

z303-alpha	=alpha
z303-birth-date	= birth-date
z303-budget	= budget
z303-con-lng	= con-lng
z303-delinq-1	= delinq-1
z303-delinq-1-cat-name	= delinq-1-cat-name
z303-delinq-1-update-date	= delinq-1-update-date
z303-delinq-2	= delinq-2
z303-delinq-2-cat-name	= delinq-2-cat-name
z303-delinq-2-update-date	= delinq-2-update-date
z303-delinq-3	= delinq-3
z303-delinq-3-cat-name	= delinq-3-cat-name
z303-delinq-3-update-date	= delinq-3-update-date
z303-delinq-n-1	= delinq-n-1
z303-delinq-n-2	= delinq-n-2
z303-delinq-n-3	= delinq-n-3
z303-field-1	= field-1
z303-field-2	= field-2
z303-field-3	= field-3
z303-home-library	= home-library
z303-id	= id
z303-ill-library	= ill-library
z303-name	= name
z303-name-key	= name-key
z303-open-date	= open-date
z303-profile-id	= profile-id
z303-proxy-for-id	= proxy-for-id
z303-title	= title
z303-update-date	= update-date

z304-address-0	= address-0
z304-address-1	= address-1
z304-address-2	= address-2
z304-address-3	= address-3
z304-email-address	= email-address
z304-telephone	= telephone
z304-zip	= zip
z305-bor-status	= bor-status
z305-expiry-date	= expiry-date

Voyager Attributes

ubid	= ubid
password	= password
passwordType	= passwordType
lastname	= lastname

DigiTool Attributes

z312-birth-date	= birth_date
z312-bor-id	= bor_id
z312-id	= id
z312-open-date	= open_date
z312-source-id	= id
z312-title	= title
z312-update-date	= update_date
z312_birth_date	= birth_date
z312_bor_id	= bor_id
z312_id	= id
z312_open_date	= open_date
z312_source_id	= id
z312_title	= title
z312_update_date	= update_date
z312m_bor_dept_m	= bor_dept_m
z312m_bor_group_m	= bor_group_m
z312m_bor_tuples_m	= bor_tuples_m
z312m_course_enrollment_m	= course_enrollment_m

MetaLib Attributes

The Internal User Name (Secondary Key of the Institute)

Field type > 20 characters

bor_id	= id
source_id	= id
id	= id
z312-source-id	= id
z312_source_id	= id

User Institution

institute	= institute
z312-institute	= institute
z312_institute	= institute

User Name to Be Displayed in the /V and /M Interface

Field type > 200 characters

user_name	= name
name	= name
z312-name	= name
z312_name	= name

User Title (Name Prefix)

Field type > 10 characters

user_title	= title
------------	---------

Groups Defined in the MetaLib Application

Field type > 30 characters

user_group	= group
group	= group
z312-group	= group

Portal Name

z312_portal_name	= portal_name
portal_name	= portal_name

Language to Be Used in the /V Application

The language to be used in the /V application is taken from the `./dat01/www_m_eng/user-language-include` file. If no language is set in this file, the default language is English (eng).

Field type > 3 characters

con_lng	= con-Ing
z312-con-Ing	= con-Ing

Resource Status Flag

Field type > 1 character

Valid values:

- A = active
- T = active + testing (IRDS authentication)

resource-status	= resource-status
z312-resource-status	= resource-status

User's Academic Status

The user's academic status is taken from the `./dat01/www_m_eng/academic-status-include` file.

Field type > 30 characters

Valid values:

- Undergraduate
- Graduate

user_academic_status	= academic-status
academic_status	= academic-status
z312-academic-status	= academic-status

Expiration Date

Field type > 8 characters in YYYYMMDD format

expiry_date	= expiry-date
z312-expiry-date	= expiry-date

User Address Details

Field type > 50 characters for each address

user_address_1	= address_0
z312_address_0	= address_0
z312-address-0	= address_0

When address-0 is the incoming attribute, the followed addresses are shifted accordingly:

user_address_2	= address_1
z312_address_1	= address_1
z312-address-1	= address_1
user_address_3	= address_2
z312_address_2	= address_2
z312-address-2	= address_2
user_address_4	= address_3
z312_address_3	= address_3
z312-address-3	= address_3
z312_address_4	= address_4
z312-address-4	= address_4
z312_address_5	= address_5
z312-address-5	= address_5

User Address Zip Code

Field type > 9 characters

user_zip	= zip
zip	= zip
z312-zip	= zip
z312_zip	= zip

User's E-Mail Address

Field type > 60 characters

user_email_address	= email_address
email_address	= email_address
z312_email_address	= email_address

User's Two Telephone Numbers

Field type -> 30 characters

user_telephone_1	= telephone-1
z312_telephone_1	= telephone-1
user_telephone_2	= telephone-2
z312-telephone-2	= telephone-2
z312_telephone_2	= telephone-2
course_enrollment_m	= course_enrollment_m
course-enrollment-m	= course_enrollment_m
bor_dept_m	= bor_dept_m
bor-dept-m	= bor_dept_m
bor_group_m	= bor_group_m
bor-group-m	= bor_group_m
bor_tuples_m	= bor_tuples_m
bor-tuples-m	= bor_tuples_m

E

Example of the PDSDefinitions File

The following is an example of the PDSDefinitions file. Note that the Oracle section is relevant only if you are working with a shared Oracle database topology.

```
our ($server_httpd)          = "https://il-
eldev03.corp.exlibrisgroup.com";
our ($server_httpsd)         = "https://il-
eldev03.corp.exlibrisgroup.com";
our ($server_pds)            = "https://il-
eldev03.corp.exlibrisgroup.com/pds";
our ($pds_directory)         = "/exlibris/metalib/m4_8/pds";
our ($pds_programs)          = "$pds_directory"."\/program";
our ($pds_html_form)         = "$pds_directory"."\/html_form";
our ($pds_conf_table)        = "$pds_directory"."\/conf_table";
our ($pds_proc)              = "$pds_directory"."\/service_proc";
our ($pds_icon)              = "$server_httpd/$pds_html_form"."\/
icon";
our ($logdir_directory)      = "/exlibris/metalib/m4_8/log";
our ($tmp_directory)         = "/exlibris/metalib/m4_8/tmp";
our ($debug)                 = "Y";
our ($exlibrisgateway)       = "N";
our ($default_con_lng)       = "ENG";
our ($pds_files)             = "$pds_directory"."\/pds_files";
our ($pds_cookie_exp)        = "0M";
our ($pds_utf_prog)          = "N";
if ($debug ne 'Y') {
    $debug = '';
} //
The following: only for Oracle

our ($db_driver)             = "Oracle";
our ($db_host)               = "il-mldev03.corp.exlibrisgroup.com";
our ($db_name)                = "example_name";
our ($db_port)                = "1521";
our ($db_rac)                 = "N"; // Y
our ($db_direct)             = "direct_con";
our ($db_user)                = "alma_oltp";
our ($db_pass)                = "!encrypt!HlVMUuX/
S06czvLK0FuUJe8t5Z2urT91EknsQlEnXcPfaM6c0K+Biw==";
our ($use_oracle)            = "N"; // Y
;
```

NOTES:

The "db_direct" parameter is used for RAC connection.

If this line is included, PDS uses it to build the Oracle connection string. Otherwise, it uses the standard variables - for example, \$db_host & \$db_name.

If you are using the pds_upgrade_kit to create the Oracle connection, enter the following:

Use Oracle RAC connection? (Y/[N]) Y

Enter direct Oracle connection [NONE] alma_oltp

Index

A

adding institutions manually, 32

Aleph

- authentication method, 69
- bor_info.tag attributes, 180
- configuring for working with the PDS, 39
- example of authentication configuration, 79
- example of PDS login page, 92
- standard tab_service.{institute} configuration, 168

Aleph X-Server

- standard tab_service.{institute} configuration, 172

Apache configuration

- and Shibboleth, 131

attribute mapping

- bor_info.tags table, 179, 187
- overview, 102

Attribute Mapping page, 102

attributes

- requesting, 14

AUTHENTICATE service, 76

authentication configuration

- examples, 79

authentication methods

- configuring manually, 76
- configuring via the configuration wizard, 68
- testing, 76

Authentication Methods page, 68

automatic logoff option, 155

B

BOR_INFO service, 101

bor_info.tags

- attribute mapping table, 179, 187

C

calling applications

- and institutions, 25
- and the PDS, 11
- configuration, 37
- requests for user attribute retrieval, 107
- target attributes, 175

CAS, 13

CAS authentication

- configuring, 119

CGI hook, 14

- configuring manually, 86

configuring via the wizard, 75

standard tab_service.{institute} configuration, 173

character conversion

- configuring manually, 33

check utility, 161

configuration wizard

- securing, 146

Configure page, 69

configuring

- authentication methods manually, 76
 - authentication methods via the configuration wizard, 68
 - calling applications, 37
 - institutions manually, 30
 - institutions via the configuration wizard, 26
 - LDAP services manually, 81
 - local login, 67
 - logout manually, 113
 - logout via the configuration wizard, 112
 - remote CGI hook manually, 86
 - remote login, 61
 - remote login manually, 63
 - remote login via the configuration wizard, 62
 - remote SSO manually, 53
 - remote SSO via the configuration wizard, 52
 - security for the PDS, 145
 - Shibboleth to work with the PDS, 133
 - SSO manually, 47
 - SSO via the configuration wizard, 45
 - user attribute mapping via the configuration wizard, 102
 - user attribute retrieval methods via the configuration wizard, 98
 - user attributes manually, 101
- customizing
- the institution login page, 93

D

debug mode, 157

debugging

- the PDS, 153

default institution override

- configuring, 34

DigiTool

- authentication method, 69
- bor_info.tag attributes, 182
- configuring for working with the PDS, 38

- example of authentication configuration, 79
- example of PDS login page, 91
- standard tab_service.{institute} configuration, 169
- target attributes, 176

direct access to the PDS, 153

E

encrypt passwords utility, 163

error messages, 156

Ex Libris products

- configuring for working with the PDS, 37
- target attributes, 175

Ex Libris products working with the PDS, 11

EZproxy authentication

- configuring, 124

G

General Attributes page, 28, 62

I

iChain, 13

iChain authentication

- configuring, 125

Institution Configuration page, 27

institution display name

- configuring manually, 32

institution display order

- configuring, 34

institution login page

- customizing, 93

institutions

- adding manually, 32
- concept, 14
- configuring manually, 30
- configuring via the configuration wizard, 26
- overview, 25

L

language-specific login pages, 93

LDAP

- attribute mapping, 85
- authentication method, 71
- configuration file, 81
- configuration file examples, 84
- example of authentication configuration, 80
- password encryption, 164
- server, 13
- services, configuring manually, 81
- standard tab_service.{institute} configuration, 171

LOAD_LOGIN

- configuring, 63

load-login request, 13

LOAD_SSO

- configuring, 54

local institution page, 154

local login

- configuring, 67
- page, 59
- page display, 89

logged on page, 156

logged out page, 154

logging in to a calling application, 13

login

- configuring local login, 67
- configuring remote login, 61

login page

- language-specific, 93
- local, 59
- overview of handling, 59
- remote, 60

login page for institution

- customizing, 93

login workflow

- Shibboleth, 139

logout, 14

- automatic option, 155
- configuring manually, 113
- configuring via the configuration wizard, 112

logout configuration

- overview, 111

logout workflow

- Shibboleth, 140

M

manual

- configuration of authentication methods, 76
- configuration of institutions, 30
- configuration of LDAP services, 81
- configuration of logout, 113
- configuration of remote CGI hook, 86
- configuration of remote login, 63
- configuration of remote SSO, 53
- configuration of Shibboleth, 137
- configuration of SSO, 47
- configuration of user attributes, 101
- mapping of user attributes, 105

manual configuration utilities, 161

mapping

- user attributes, 14
- user attributes manually, 105

MetaLib

- authentication method, 69
- bor_info.tag attributes, 182
- configuring for working with the PDS, 37
- example of authentication configuration, 80
- example of PDS login page, 91
- standard tab_service.{institute} configuration, 169

target attributes, 175

O

Oracle
password encryption, 165
override for default institution
configuring, 34
overview of the PDS, 12

P

password encryption utility, 163
PDS - Attribute Mapping page, 102
PDS - Configure page, 69
PDS - Institution Configuration page, 27
PDS 1.3, 12
PDS 2.0, 11
PDS 2.1, 11
PDS Authentication Methods page, 68
PDS check utility, 161
PDS General Attributes page, 28, 57, 62
PDS User Attributes page, 99
PDSDefinitions file
SaaS support, 19
PDS-SSO Configuration page, 45
Primo
configuring for working with the PDS, 37
target attributes, 176
products working with the PDS, 11

R

remote authentication systems, 13
remote CGI hook, 14
configuring manually, 86
configuring via the wizard, 75
standard tab_service.{institute} configuration, 173
remote institution page, 154
remote login
configuring, 61
configuring manually, 63
configuring via the configuration wizard, 62
remote login page, 60
remote PDS SSO check, 57
remote SSO, 51
configuring manually, 53
configuring via the configuration wizard, 52
REMOTE_LOGIN
configuring, 65
security enhancement, 146
REMOTE_SSO
configuring, 56
security enhancement, 146

S

SaaS
working together with local products, 57
SaaS environment, 19
security
and the configuration wizard, 146
enhancement for REMOTE_LOGIN and REMOTE_SSO, 146
patch for X-Server, 148, 149
security for the PDS
configuring, 145
setup validity testing utility, 161
Shibboleth, 13
Apache configuration, 131
configuring as a WAYF, 141
configuring manually, 137
configuring to work with the PDS, 133
configuring via the configuration wizard, 134
login workflow, 139
logout workflow, 140
overview, 129
prerequisites for working with, 130
SSO workflow, 141
Shibboleth service provider
tips for installing, 130
Single Sign-On (SSO), 13
configuration overview, 43
configuring manually, 47
configuring via the configuration wizard, 45
remote, 51
synchronizing subdomains, 50
SSL
configuring, 145
SSO Configuration page, 45
SSO workflow
Shibboleth, 141
subdomains
synchronizing SSO, 50

T

tab_service.{institute} file, 30
example configurations, 167
target attributes
Ex Libris calling applications, 175
Test for Authentication Method window, 76
Test for User Attribute Method window, 101
testing
authentication methods, 76
user attribute retrieval methods, 101
topologies, 11

U

user attribute mapping

- configuring manually, [105](#)
- configuring via the configuration wizard, [102](#)
- user attribute retrieval
 - overview, [97](#)
 - requests by calling applications, [107](#)
 - with Voyager, [108](#)
- user attribute retrieval methods
 - configuring via the configuration wizard, [98](#)
 - testing, [101](#)
- user attributes
 - configuring manually, [101](#)
 - mapping, [14](#)
 - requesting, [14](#)
- User Attributes page, [99](#)
- UTF8, [33](#)
- utilities
 - encrypt passwords, [163](#)
 - for manual configuration, [161](#)
 - setup validity testing, [161](#)

V

- Voyager
 - authentication method, [69](#)
 - bor_info.tag attributes, [181](#)
 - configuring for working with the PDS, [37](#)
 - example of authentication configuration, [80](#)
 - example of PDS login page, [92](#)
 - standard tab_service.{institute} configuration, [171](#)
 - user attribute retrieval, [108](#)

W

- WAYF
 - configuring Shibboleth as, [141](#)
- wizard
 - configuring authentication methods, [68](#)
 - configuring institutions, [26](#)
 - configuring logout, [112](#)
 - configuring remote login, [62](#)
 - configuring remote SSO, [52](#)
 - configuring Shibboleth, [134](#)
 - configuring SSO, [45](#)
 - configuring user attribute mapping, [102](#)
 - configuring user attribute retrieval methods, [98](#)
 - securing, [146](#)

X

- X-Server security patch, [148](#), [149](#)