UnboundID® Identity Broker

Installation Guide

Version 4.1

UnboundID Corp
13809 Research Blvd., Suite 500
Austin, Texas 78750
Tel: +1 512.600.7700
Email: support@unboundid.com

Copyright

This document includes possible practices and procedures relating to securing and monitoring an UnboundID software infrastructure. It is intended only for security professionals. While reasonable efforts have been made to include best practices and helpful suggestions, it is the sole responsibility of the security professional to evaluate all such practices and procedures, to decide the best approach for the applicable unique environment, to implement such decisions related to securing software infrastructure, and for the results obtained. This document does not constitute "Documentation" as such term is defined in any agreement between UnboundID and any of its customers.

This distribution may include materials developed by third parties. Third-party URLs are also referenced in this document. UnboundID is not responsible for the availability of third-party web sites mentioned in this document. UnboundID does not endorse and is not responsible or liable for any content, advertising, products, or other materials that are available on or through such sites or resources. UnboundID will not be responsible or liable for any actual or alleged damage or loss caused or alleged to be caused by or in connection with use of or reliance on any such content, goods, or services that are available on or through such sites or resources. “UnboundID” is a registered trademark of UnboundID Corporation. UNIX is a registered trademark in the United States and other countries, licenses exclusively through The Open Group. All other registered and unregistered trademarks in this document are the sole property of their respective owners.

Table of Contents

Preface ..... **i**
About This Guide i
Audience i
Related Documentation ii

Chapter 1 Introduction **1**
About the UnboundID Identity Broker 1
About the UnboundID Privacy Suite 2
About the Web Reference Interface 4
About UnboundID 5

Chapter 2 System Requirements **7**
Before You Begin 7
Configuring File Descriptor Limits 8
To Set the File Descriptor Limit 8
Setting the Maximum User Processes 9
Supported Platforms 9

Chapter 3 Quick Install **13**

Chapter 4 Installation **15**
Before You Begin the Install 15
Installing the JDK 16
Installing the Identity Data Stores and Identity Proxies 16
To Install the Identity Data Store 17
Installing the Identity Broker 20
To Unpack the Build Distribution 20
To Install the Identity Broker 21
To Install a Clone Identity Broker 28
To Install the Identity Broker with an Existing Truststore 29

Chapter 5 Configuration **31**
About the Command-Line Tools 31
Configuring Policies 32
To Configure a Policy Using the Command-Line Tools 32
Adding Additional Admin Accounts 37
To Add Additional Administrator Accounts 37
Running the sample-data-loader Tool 38
To Run the sample-data-loader Tool 38

Creating Trace Filters 39
To Create a Trace Filter 39

Chapter 6 Testing 41
Testing the Sample Policies 41
To Test the Sample Policies 41
Testing Consent 42
To Test Consent 42
Testing the OAuth2 Authorization Flows 44
Creating Resources, Purposes, Actions, Scopes and Applications 44
To Create Resources, Purposes, Actions, Scopes, and Apps 44
To Test the OAuth2 Resource Owner Grant Type 47
To Test the OAuth2 Client Credentials Grant Type 48
To Test the OAuth2 Auth Code and Implicit Grant Types 49

Chapter 7 Management 51
Running the Identity Broker 51
To Run the Identity Broker 51
To Run the Identity Broker in the Foreground 51
Stopping the Identity Broker 52
To Stop the Identity Broker 52
To Schedule a Server Shutdown 52
To Run an In-Core Restart 52
Uninstalling the Identity Broker 53
To Uninstall the Identity Broker 53

Chapter 8 Reference 55
About the Identity Broker Files and Folders 55
About the Identity Broker Tools 56
About the Identity Broker Roles and Scopes 58
Supported Storage Options 62

Index 65

Preface

The UnboundID Identity Broker Installation Guide presents the procedures to install and configure your Identity Infrastructure. We appreciate any feedback and requests for specific topics to cover in future revisions of this guide.

About This Guide

This guide presents procedures to install and configure your Identity Infrastructure, powered by the UnboundID product suite. The guide references the other products in the UnboundID product family including the UnboundID Privacy Suite, UnboundID Identity Broker, UnboundID Identity Data Store, UnboundID Identity Proxy, UnboundID Identity Data Sync Server plus the Identity Broker API.

This guide shows the basic installation steps to set up a simple Identity Broker deployment for evaluation purposes only. Actual production deployments may require additional steps and software depending on your system.

Audience

This guide is intended for Identity Architects who are evaluating and designing their Identity Infrastructure and Identity Administrators, responsible for installing and deploying the Identity Broker within their environments.

Familiarity with system-, user-, and network-level security principles is assumed. Knowledge of directory services principles is recommended. It is not necessary to be familiar with the features of the UnboundID system.

To use this guide effectively, readers should be familiar with the following topics:

- ReSTful web services and principles
- JSON or XML serialization formats
- XACML 3.0
- OAuth2 specification (RFC 6749)
- OAuth2 Bearer Token specification (RFC6750)

Related Documentation

The UnboundID Identity Broker 4.1 comes with the following document set, available in the `docs` folder of the server.

- *UnboundID Identity Broker Installation Guide*
- *UnboundID Identity Broker Client Application Developer Guide*
- *UnboundID Identity Broker Reference Guide (HTML)*
- *UnboundID Identity Broker REST API Reference*
- *UnboundID Identity Broker Administration Guide (available Summer 2013)*
- *UnboundID Identity Broker Deployment Guide (available Summer 2013)*

UnboundID Identity Data Store 4.1:

- *UnboundID Identity Data Store Administration Guide*
- *UnboundID Identity Data Store Reference Guide (HTML)*

UnboundID Identity Data Proxy 4.1:

- *UnboundID Identity Proxy Administration Guide*
- *UnboundID Identity Proxy Reference Guide (HTML)*

UnboundID Identity Data Sync 4.1:

- *UnboundID Identity Data Sync Server Administration Guide*
- *UnboundID Identity Data Sync Server Reference Guide (HTML)*

UnboundID Metrics Engine 4.1:

- *UnboundID Metrics Engine Administration Guide*
- *UnboundID Metrics Engine Reference Guide (HTML)*

Chapter 1 Introduction

The Identity Economy is predicted to be the new "oil" or resource of the 21st century (World Economic Forum), forecasting to grow into a $875 billion industry by 2020 (Boston Consulting Group). For companies with large user data (i.e., *Big Data*), the challenge is to monetize this valuable asset, while balancing data privacy regulations. The UnboundID Identity Broker is *the* solution to power your Big Data requirements in this new world market.

About the UnboundID Identity Broker

The UnboundID Identity Broker is the first of a new class of components for consumer and subscriber identity management architectures.

As a stand-alone server, it provides authorization decisions for client applications, provisioning systems, API gateways and analytical tools in any architecture involving personal, account, or sensitive identity data.

Unlike traditional role-based access control (RBAC) and XACML-based policy-evaluation tools, the Identity Broker is designed to make high-volume and high-speed authorization decisions based on ever-changing consumer profile and consent data. Functionally, the Identity Broker is both the Policy Decision Point and the OAuth2 provider for externalized authorization. Performance-wise, the Identity Broker can support the request volumes driven by the complex, real-time interactions necessary to support today's consumer-facing mobile, social, and cloud ecosystems.

Most organizations today are taking steps toward creating a common (or unified) customer profile – where the disparate and inconsistent pictures of a customer are reconciled into a single profile for use across multiple channels. An essential part of creating that common identity profile is to provide an analogous common consent

function—centralizing multiple overlapping opt-in and opt-out registries and centralizing the logic for determining which applications should access the data in the profile, and for which purposes.

In conjunction with the other components of an identity and access management (IAM) stack, the Identity Broker solves the problem of managing user consent and controlling access to a common user or subscriber profile by providing Policy Decision, Policy Information, and Policy Repository functions for data in existing repositories, without migration.

Since the policy and consent functions are centralized, the identity architecture ensures that corporate, regulatory, and per-user policies are applied consistently across all applications. In addition, the UnboundID Identity Data Platform (Identity Data Store, Identity Proxy, and Identity Data Sync) can be used to create a common identity and single view of the customer:

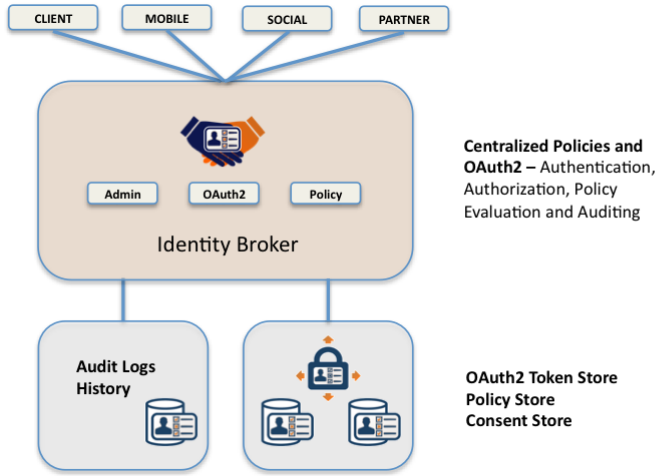


Figure 1-1. Identity Broker System

About the UnboundID Privacy Suite

The UnboundID Privacy Suite is a tightly integrated software solution enabling service providers to simultaneously access and protect customer data. Designed for tier-1 Telcos, the solution connects your mission critical data infrastructure to enhance and expand end-user experiences. The Privacy Suite consists of the following components:

- Telco-grade Identity Data Store
- High-performance Policy Engine
- OAuth2 Authorization Server

- Collection of stakeholder specific-applications that organize your data and make your policies actionable in real time
- Platform for integrating data consuming applications to the rules of the road

The Privacy Suite is deployed as three separate, run-time components: the web app layer, the middle-service layer, and the backend data layer. The web app layer consists of the administrative user interface and reference end-user interfaces that interact with the Identity Broker. At the heart of the service layer is the UnboundID Identity Broker, a robust and highly-scalable server that powers an OAuth2 authorization service, a Policy service, and an Administrative service for your Identity Infrastructure. The data layer consists of the UnboundID Identity Data Store server, which manages storage of end-user consent, OAuth2 tokens, configuration and other privacy domain data. The run-time components provide a robust and scalable, end-to-end privacy solution to power your Identity Infrastructure.

The Privacy Suite provides an easy way to deploy an Identity Infrastructure system out-of-the-box with the flexibility to integrate and customize the various components into your current deployment. The UnboundID Privacy Suite and UnboundID Identity Broker comes with REST APIs, such as the OAuth2 API used to customize token requests, validations, or revocations; the Admin API to customize policy creation and application registration; the Policy Definition Import API to configure how you want to export your policies into XACML; and other APIs for easy integration with your production environment.

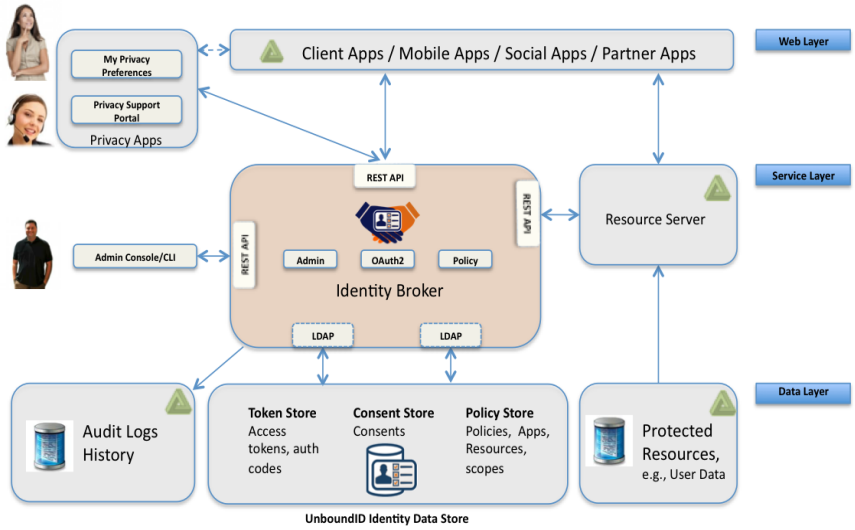


Figure 1-2. UnboundID Privacy Suite

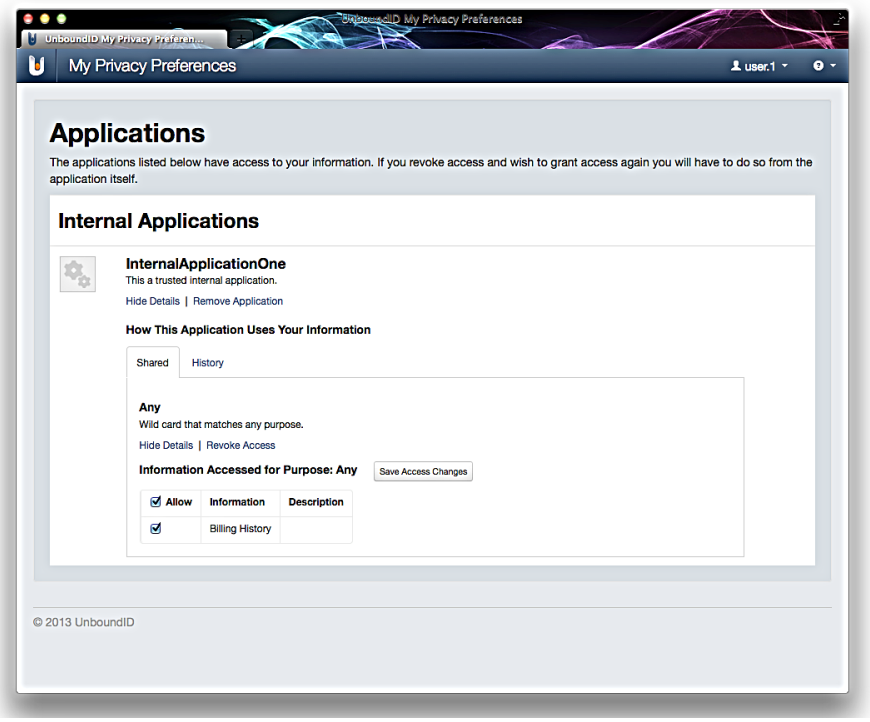
The Privacy Suite comes with the Broker Console, a graphical administrative interface to manage your Privacy Suite. You can also build your own interface using the

APIs provided with the Identity Broker. Command-line tools are also available for administrators who prefer scripting.

The OAuth2, Consent, and Policy data are stored in a backend set of UnboundID Identity Data Store servers. If you want to use your existing data stores, you can easily integrate this system with your current infrastructure. For example, if you want to store your OAuth2 tokens in another database, you can configure it via an extension.

About the Web Reference Interface

The Privacy Suite comes with a web reference implementation, My Privacy Preferences, that operates in two modes: a customer portal and a support portal. If a customer grants consent for third-party access to his/her resources (typically on the customer's web site), the customer can view or revoke consent on the My Privacy Preferences app.



If a support staff member logs into the My Privacy Preferences app, it redirects her to a Privacy Portal page to assist any user with a consent management problem.

This reference implementation can be used as-is or re-branded by your company.

Note that the Identity Broker communicates with its backend datastores over LDAP and uses a load-balancing algorithm to communicate with one or many data stores. The Privacy Preferences web app communicates with the user data store over an HTTPS SCIM endpoint. Also, the Identity Broker REST API retrieves or revokes consent over the SCIM endpoint. If an external backend data store goes down, the Identity Broker can still continue running. However, this may not be the case for the Privacy Preferences web app or Identity Broker REST API, which are reliant on a single external datastore.

About UnboundID

UnboundID Corp is a leading Identity Infrastructure Domain solutions provider with proven experience in large-scale identity data solutions with certified information privacy professional team members. The following are just a few reasons why UnboundID does it best:

- **Secure End-to-End Customer Data Privacy Solution**. UnboundID has developed a comprehensive integrated Identity Data platform with authorization and access controls to enforce privacy policies, control user consent, and manage resource flows. The system protects data in all phases of its life cycle (at rest, in motion, in use).
- **Purpose-Built Identity Data Platform**. UnboundID has designed a solution to consolidate, secure, and expose customer consent-given identity data. The system provides unmatched security measures to protect sensitive identity data and maintain its visibility. The broad range of platform services include, policy management, cloud provisioning, federated authentication, data aggregation, and directory services.
- **Unmatched Performance across Scale and Breadth**. UnboundID has developed a solution that accommodates the three pillars of performance-at-scale: users, response time, and throughput. The system supports real-time data at large-scale consumer facing service providers today.
- **Support for External APIs**. UnboundID has developed solutions to interface with various external APIs to access a broad range of services to its products. These APIs include XACML 3.0, SCIM, LDAP, OAuth2 and very soon in a future release, OpenID Connect and SAML.
- **Leading Manageability, TCO, and ROI with Identity Management Domain Expertise**. UnboundID has developed a lightweight architecture that keeps hardware and people costs down with the ability to easily add new services. The Identity Data Platform has components that can exist as a complete solution or can be

individually integrated within an existing production system with multiple RESTful API endpoints and extension points to interface with the server.

Chapter 2 System Requirements

The UnboundID Identity Broker requires few technical prerequisites and can be deployed in multiple configurations. The Identity Broker has multiple admin and end-user RESTful API endpoints that allow integration with existing user interfaces and administrative tools, supporting swappable identity management schemes that accept in-house single-sign-on technologies. The Identity Broker can be deployed atop virtualized and/or commodity hardware, monitored using the platform's built-in tools or through external monitoring tools of your choice accessible through the API.

Before You Begin

The UnboundID Identity Broker requires certain software packages for the proper operation of the server and other packages may be required depending on your deployment. Read the UnboundID Identity Broker Deployment Guide to determine the configuration alternatives that may exist for your system.

Minimally, the following items are required before you install the Identity Broker:

- Java 6 or Java 7
- Minimum of 2 GB RAM
- UnboundID Identity Data Store 4.1

There may be other required software for your system, please review the Supported Platform chart.

Configuring File Descriptor Limits

The UnboundID Identity Broker allows for an unlimited number of connections by default, but is restricted by the file descriptor limit on the operating system. Many Linux distributions have a default file descriptor limit of 1024 per process, which may be too low for the Identity Broker if it needs to handle a large number of concurrent connections.

To fix this condition, we recommend setting the maximum file descriptor limit per process to 65,535 on Linux systems.

To Set the File Descriptor Limit

1. Display the current hard limit of your system. The hard limit is the maximum server limit that can be set without tuning the kernel parameters in the proc file-system.

   ```
   ulimit -aH
   ```

2. Edit the **`/etc/sysctl.conf`** file. If there is a line that sets the value of the **`fs.file-max`** property, make sure its value is set to at least 65535. If there is no line that sets a value for this property, add the following to the end of the file:

   ```
   fs.file-max = 65535
   ```

3. Edit the **`/etc/security/limits.conf`** file. If the file has lines that sets the soft and hard limits for the number of file descriptors, make sure the values are set to 65535. If the lines are not present, add the following lines to the end of the file (before "#End of file"). Also note that you should insert a tab, rather than spaces, between the columns.

   ```
   * soft nofile 65535
   * hard nofile 65535
   ```

4. Reboot your system, and then use the **`ulimit`** command to verify that the file descriptor limit is set to 65535.

   ```
   ulimit -n
   ```

Setting the Maximum User Processes

On some Linux distributions (Redhat Enterprise Linux Server/CentOS 6.0 or later), the default maximum number of user processes is set to 1024, which is considerably lower than the same parameter on older distributions (e.g., RHEL/CentOS 5.x). The default value of 1024 leads to some JVM memory errors when running multiple servers on a machine due to each Linux thread being counted as a user process. (Note that this is not an issue on Solaris and AIX platforms as individual threads are not counted as user processes.)

At startup, the Identity Broker and its tools automatically attempt to raise the maximum user processes limit to 16,383 if the value reported by **`ulimit`** is less than that. If, for any reason, the server is unable to automatically set the maximum processes limit to 16,383, an error message will be displayed. It is recommended that the limit be set explicitly in **`/etc/security/`** limit.conf. For example:

```
* soft nproc 100000
* hard nproc 100000
```

Administrators can also manually configure the value by setting the **`NUM_USER_PROCESSES`** environment variable to 16383 or by setting the same variable in a file named **`config/num-user-processes`**.

Supported Platforms

The following chart shows the current supported platforms and software versions, which may or may not be required for the UnboundID Identity Broker depending on your deployment. UnboundID does not require specific hardware; our products run excellently on commodity systems.

- **Reference**. Indicates that UnboundID QA has tested and confirmed that the system works as documented.
- **Yes**. Indicates that the supported platform is included in UnboundID support agreements
- **Eval Use Only**. Indicates that the platform can be used to evaluate UnboundID software but should not be used for production deployments.
- **Experimental**. Indicates that UnboundID is undergoing tests on the platform and may or may not be supported in the future.

Table 1. Supported Platforms & Software

Operating Systems	Supported?	Comments
RedHat Linux 5.5	Reference	
RedHat Linux 6.2	Reference	
Solaris 10 x86 update 9	Reference	
Solaris 11	Yes	
AIX 6.1	Yes	Included in UnboundID Support Agreements
Centos 5.5	Yes	Included in UnboundID Support Agreements
Centos 5.6 through 5.0	Reference	
Centos 6.0, 6.1	Yes	Included in UnboundID Support Agreements
Centos 6.2	Reference	
Centos 6.3	Yes	
Windows	Eval Use Only	
MacOS	Eval Use Only	

Table 2. Java JDKs

JDKs	Supported?	Comments
Oracle/IBM JDK6.x 64-bit	Reference	
Oracle JDK 7.x 64-bit	Reference	
IBM JDK 7.x 64-bit	Yes	
Azul Zing JVM	Experimental	

Table 3. Virtual Hosts/Platforms

Virtual Hosts/Platforms	Supported?	Comments
VMWare vSphere & ESX 4 update 2	Yes	
VMWare vSphere & ESX 5	Yes	
IBM AIX Virtualization (LPAR, PR/SM)	Yes	
Amazon EC2	Experimental	

Table 4. App Servers/Servlet Containers

App Servers/Servlet Containers	Supported?	Comments
Apache Tomcat 6.x	Yes	
JBoss 7.x	Yes	

Table 5. Identity Data Platform

Identity Data Platform	Supported?	Comments
UnboundID Identity Data Store, 4.0	Yes	Required for the Policy backend
UnboundID Identity Data Proxy, 4.0	Yes	
UnboundID Identity Data Sync, 4.0	Yes	
UnboundID Metrics Engine, 4.0	Yes	Optionally, used for monitoring the Identity Broker

Table 6. Browser Software

Auxiliary Software	Supported?	Regional Preference
Internet Explorer 9 & 10	Yes	U.S.
Chrome 23+	Yes	Europe
Firefox 16+	Yes	U.S.
Safari 6+	Yes	U.S./Europe mobile

Chapter 3 Quick Install

For those who have some familiarity with UnboundID products, the following Quick Install procedures show how to set up the UnboundID Identity Broker for evaluation purposes. See "Installing the Identity Broker" on page 20 for more verbose instructions.

1. **Before You Begin**. Ensure that your system has the required software installed, such as Java 6 or 7.
2. **Install the UnboundID Identity Data Store**. Download the UnboundID Identity Data Store zip file, which is used for the Identity Broker's datastore. It must be running with no errors prior to installing the Identity Broker. Unpack it in a directory of your choice. Then, run the **`setup`** command. Follow the prompts and enter the information needed to complete the install.

   ```
   ./setup --cli
   ```

3. **Configure the Identity Data Store**. During the setup process, make sure to configure the following items on the Identity Data Store:
   - **HTTP with SSL**. The Identity Broker requires access to the user store over SCIM/HTTPS.
   - **Sample Data**. This is optional but you can quickly load sample user data.
   - **LDAPS**. Configure LDAP over SSL and generate a self-signed certificate when prompted.
4. **Install the UnboundID Identity Broker**. Download the UnboundID Identity Broker zip file. Unpack it in a directory of your choice. Then, run the **`setup`** command and accept the default prompts.

   ```
   ./setup
   ```

5. **Configure the Identity Broker**. During the setup process, read through each step as presented in the setup wizard and accept the default prompts. Make sure

to configure the following items on the Identity Broker:

- **HTTP with SSL**. The Identity Broker requires access to its backend data-stores over SCIM/HTTPS.
- **LDAPS**. Configure LDAP over SSL and generate a self-signed certificate when prompted.
- **Install All Web Apps**. Install the Broker Console and the Privacy Preferences app.
- **Install Sample Policies**. Install the sample policies so that you can view and run policy evaluations.
- **Install Sample Data**. Install the sample data, so that you can test the Identity Broker's features.

6. **Test the Identity Broker**. See See "Testing" on page 41 to open the Broker Console GUI and test the sample policies.

You have successfully installed a Privacy Suite, featuring the UnboundID Identity Broker. You can now test the various components. See "Testing" on page 41

Chapter 4 Installation

The UnboundID Identity Broker provides easy-to-use installation tools to quickly configure your server. Make sure you have set up Java and any other software required for your deployment.

This chapter presents detailed installation steps to get your Privacy Suite, powered by the UnboundID Identity Broker, up-and-running.

Before You Begin the Install

Before you begin, consider the following deployment-related issues:

- **Determine your Backend Datastore Topology**. You must first determine what your backend data store deployment will be prior to installing the Identity Broker. The deployment determines where the Identity Broker stores its policies, consent, and OAuth2 tokens for each user.

  There are two deployment models for the Identity Broker: *the Unified Policy and Consent Store*, and *the Separate Policy and Consent Store*. The Unified Policy and Consent Store stores the policy definitions in the same directory as user information. The Identity Broker attaches consent and OAuth2 tokens directly to the user entry as operational attributes.This model allows the Identity Broker direct access to the user repository.

  The Separate Policy and Consent Store ensures that the Identity Broker has no direct access to the user repository but maintains its own shadow set of user entries in the consent store, where user ID, consent, and an OAuth2 token information is stored. The shadow user entries can be created by a sync mechanism to the consent store or when the Identity Broker first writes consent or token information for the user.

For more information, see the *UnboundID Identity Broker Deployment Guide*.

- **Code Your Application and Resource Server**. The Identity Broker provides ReST endpoints for your web, mobile, social and partner applications as well as your resource server for access to the OAuth2 and Policy services and the admin tools. See the *UnboundID Identity Broker Client Developer Guide* for more information.

Installing the JDK

For optimized performance, the UnboundID Identity Broker requires Java for 64-bit architectures. You can view the minimum required Java version on your Customer Support Center portal or contact your authorized support provider for the latest software versions supported.

Even if your system already has Java installed, you may want to create a separate Java installation for use by the UnboundID Identity Broker to ensure that updates to the system- wide Java installation do not inadvertently impact the Identity Broker. This setup requires that the JDK, rather than the JRE, for the 64-bit version, be downloaded.

On Solaris systems, if you want to use the 64-bit version of Java, you need to install both the 32-bit and 64-bit versions. The 64-bit version of Java on Solaris is not a full stand-alone installation, but instead relies on a number of files provided by the 32-bit installation. Therefore, the 32-bit version should be installed first, and then the 64-bit version installed in the same location with the necessary additional files.

On other platforms (for example, Linux and Microsoft Windows), the 64-bit version of Java contains a complete installation. If you only want to run the 64-bit version of Java, then it is not necessary to install the 32-bit JDK. If you want to have both versions installed, then they should be installed in separate directories, because the files cannot co-exist in the same directory as they can on Solaris systems.

Installing the Identity Data Stores and Identity Proxies

The UnboundID Identity Broker requires that the UnboundID Identity Data Store server be in place as the backend repository for the Policy data, including the defined policies, resources, actions, applications, purposes and other items needed to make

policy decisions. The user consent decisions must also be stored in its own repository or together with user data.

The Policy store can be housed together with the user consent data on a single Identity Data Store server as a Policy-and-Consent store. This deployment is known as the *Unified Policy and Consent Store* and is recommended for installing the Identity Broker for evaluation purposes.

If your company's user data is housed on a different server or database, the Identity Broker will require access to the user information through an UnboundID Server SDK extension or through attribute copies of the user entries (usernames, DN, or IDs of users). This deployment is known as the *Separate Policy and Consent Store*.

In actual deployments, you will likely deploy multiple proxy servers, such as the UnboundID Identity Proxy Server, which fronts a backend set of UnboundID Identity Data Store servers. These solutions and others require careful pre-planning. For more information, see the *UnboundID Identity Broker Deployment Guide*.

To Install the Identity Data Store

The following procedure shows how to install a single UnboundID Identity Data Store server, deployed as a Unified Policy and Consent Store. For more information on the Identity Data Store, see the *UnboundID Identity Data Store Administration Guide*.

1. Download the UnboundID Identity Data Store zip distribution and unpack it in a folder of your choice. The zip file is labelled, **`UnboundID-DS-<version>.zip`**, where <version> is the latest 4.x build.

   ```
   $ unzip UnboundID-DS-4.1.0.0.zip
   ```

2. Change to the root directory of the UnboundID-DS folder.

   ```
   $ cd UnboundID-DS
   ```

3. Run the **`setup`** command from the command line. The **`setup`** command interactively prompts you for input to simply the install process.

   ```
   $ ./setup --cli
   ```

4. Read the license and if you agree to its terms, enter "yes".
5. Next, enter the initial root user DN for the Identity Data Store, or accept the default, (**`cn=Directory Manager`**). Then, enter a password for the root user DN, and re-enter it to confirm it.

6. Select how you would like to enable access through HTTP. You will need to set up HTTPS for SCIM access to the My Privacy Preferences web app. In this example, enter the option for "HTTP with SSL."

```
Would you like to enable access through HTTP?

 1)  Do not configure HTTP access at this time
 2)  HTTP
 3)  HTTP with SSL
 4)  Both HTTP and HTTP with SSL

Enter choice [1]:3
```

7. Enter the port to accept connections from HTTPS clients. This port is used to access user information over HTTPS using SCIM. For this example, press Enter to accept the default (8443). Note that the SCIM URL will be **`https://<hostname>:8443/`**. When installing the Identity Broker, you will be asked for this SCIM URL.

8. For the LDAP port, enter the port to accept connections from LDAP clients. For this example, press Enter to accept the default (1389).

9. For enabling LDAPS, enter "yes", and then enter the port to accept connections from LDAPS clients. For this example, press Enter to accept the default LDAPS port (1636).

10. For StartTLS, press Enter to accept the default (no).

11. For certificate options, select the certificate option for the server. For this example, press Enter to accept the default (generate self-signed certificate). For actual deployments, you will likely use an existing certificate.

```
Certificate server options:

    1)  Generate self-signed certificate (recommended for testing
        purposes only
    2)  Use an existing certificate located on a Java Key Store (JKS)
    3)  Use an existing certificate located on a PKCS12 key store
    4)  Use an existing certificate on a PKCS11 token

Enter choice [1]:
```

12. If you want to specify particular IP addresses to only accept connections, enter "yes" and then enter the IP addresses. For this example, press Enter to accept the default (no).

13. Next, specify the base DN for the Identity Data Store repository. Press Enter to accept the default (**`dc=example,dc=com`**).

14. Next, select the option to populate the database. For this example, select load auto-generated samples and then enter the number of entries (2000). Note: If you

select "Leave the database empty" to add the base entry at a later time, you must manually create an LDIF file with the base entry, then run **`ldapmodify`** to add the entry to the Identity Data Store. Next, run the **`prepare-external-store`** tool on the Identity Broker, so that the global access control instructions (ACIs) are properly added to the backend Identity Data Store.

```
$ cat base-entry.ldif
dn: dc=example,dc=com
objectClass: top
objectClass: domain
objectClass: extensibleObject
dc: example

$ bin/ldapmodify -p 1389 -D "cn=Directory Manager" -w password -a -l base-entry.ldif

# Change to the UnboundID Identity Broker server root
$ cd ../UnboundID-Broker

# Run the prepare-external-store to set up the communication and ACIs
$ bin/prepare-external-store --hostname server.example.com --port 1636 --useSSL \
--brokerBindPassword password --brokerTrustStorePath config/truststore \
--brokerTrustStorePasswordFile config/truststore.pin --isConsentStore
```

15. For JVM Memory optimization, press Enter for "no"(i.e., we do not want to tune the JVM memory).
16. For auto-priming of the database contents, select the option to prime the database. For this example, press Enter for "no". In actual deployments, you will want to prime the database contents.
17. To start the server after the configuration, press Enter for "yes".
18. Next, review the Setup Summary, and accept the configuration, redo it, or cancel.

```
Setup Summary
=============
Root User DN:         cn=Directory Manager
LDAP Listener Port:   1389
HTTP Listener Port:   disabled
Secure Access:        Enable SSL on LDAP Port 1636
                      Enable SSL on HTTP Port 8443
                      Create a new Self-Signed Certificate
Directory Data:       Create New Base DN dc=example,dc=com
                      Base DN Data: Import Automatically-Generated Data (2000 Entries)

Start Server when the configuration is completed

What would you like to do?

    1)  Set up the server with the parameters above
    2)  Provide the setup parameters again
    3)  Cancel the setup

Enter choice [1]:
```

19. Run the **status** tool to see if the server is running and view its available ports.

```
$ bin/status
```

20. Run **curl** to query the server's Schemas endpoint to confirm that the standard "User" resource type is supported. Specify the HTTPS port configured in step 7. The **-k** option runs the command as insecure connection due to the self-signed certificate.

```
curl -k -u "cn=directory manager:password" \
'https://localhost:8443/Schemas?filter=name+eq+"User"'
```

Installing the Identity Broker

The Identity Broker provides a **setup** command-line tool to install and configure the server on your system. The tool installs the Identity Broker, web apps, and sets up the connections and endpoint in three phases: install, configuration, and communication setup.

During the initial install phase, the **setup** tool sets up the client connections, ports, and security, configures memory preload and tests the connections. The configuration phase invokes the **create-initial-broker-config** tool to set up a basic initial configuration, which is recommend for all new users. You can run the **dsconfig** tool to fine-tune your configuration at a later time.

During the final install phase, the **create-initial-broker-config** tool invokes the **prepare-external-store** tool within its session to set up the Broker Admin accounts on the backend servers and then tests the communication between the Identity Broker and the backend Identity Data Stores. The tool configures the Identity Broker, sets up the certificates, and starts the server for operation.

You can then access the Broker Console GUI or the **broker-admin** tool to set up your policies.

To Unpack the Build Distribution

1. Download the latest zip distribution of the UnboundID Identity Broker software.
2. Unzip the compresses zip archive file in a directory of your choice.

```
$ unzip UnboundID-Broker-<version>.zip
```

You can now set up the Identity Broker.

To Install the Identity Broker

1. Run the **setup** command in the **<server-root>** directory. The initial setup is similar to that of the Identity Data Store, where **setup** installs the code, sets up the ports for the client connections, and tests the connections.

```
$ ./setup
```

2. Review the licensing agreement, and accept the terms of this license by typing "yes".
3. The **setup** tool allows you to clone a configuration by adding to an existing Identity Broker topology. For this example, we are assuming that you are installing the Identity Broker for the first time, press Enter to accept the default (no).
4. Enter the fully qualified host name or IP address of the local machine that hosts the Identity Broker. For this example, press Enter to accept the default.
5. Enter the initial root user DN and password for the Identity Broker. For this example, press Enter to accept the default (**cn=Directory Manager**).
6. Enter the port to accept HTTPS client connections. This HTTPS port is used for the Identity Broker to provide the various endpoints to process a policy or OAuth2 token. For this example, press Enter to accept the default.
7. Enter the port to accept LDAP client connections. For this example, press Enter to accept the default.
8. For enabling LDAPS, type "yes" to enable secure connections, and then enter the port to accept connections from LDAPS clients. For this example, press Enter to accept the default.
9. Type "yes" if you want to enable StartTLS connections over regular LDAP connection. For this example, press Enter to accept the default (no).
10. If you selected secure connections, select the server certificate options: 1) generate a self-signed certificate, 2) use an existing JKS certificate, 3) use a PKCS12 certificate, or 4) use a PKCS11 token.
11. Next, you can designated specific addresses on which the Identity Broker listens for client connections. Type "yes" and then enter the specific IP addresses for the client connections. For this example, press Enter to accept the default (no).
12. Next, if you want to tune the JVM memory for maximized performance on the machine that houses the Identity Broker, enter "yes" and then the amount of memory to allocate the memory to the server and tools. For this example, press Enter to accept the default (no), but for an actual production deployment, you may want to set this property for optimum performance.

13. Next, you can start the server when the configuration is applied to the host machine by pressing Enter to accept the default (yes).

14. Review the configuration options that you entered or selected, you can always cancel what you entered and redo the installation, and then press Enter to accept the default (1 for "set up the server with the parameters).

```
Setup Summary
=============
Root User DN:        cn=Directory Manager
LDAP Listener Port:  2389
Secure Access:       Enable SSL on LDAP Port 2636
                     Enable SSL on HTTP Port 9443
                     Create a new Self-Signed Certificate

Start Server when the configuration is completed

What would you like to do?

    1)  Set up the server with the parameters above
    2)  Provide the setup parameters again
    3)  Cancel the setup

Enter choice [1]:
```

15. The next phase of the install prompts you to configure the server using the **`create-initial-broker-config`** tool. The **`setup`** tool will run it if you decide to continue. The **`create-initial-broker-config`** tool interactively prompts you for input and is useful for first-time configurations. For additional changes, you can run the **`dsconfig`** tool later to fine-tune your configuration.

```
This server is now ready for configuration.  What would you like to do?

    1)  Start 'create-initial-broker-config' to create a basic initial configuration
        (recommended for new users)
    2)  Start 'dsconfig' to create a configuration from scratch
    3)  Quit

Enter choice [1]:
```

16. The **`create-initial-broker-config tool`** describes the two basic deployment models for the Identity Broker. The *Unified Policy and Consent Store* combines the consent data together with the Policy definitions in the same Identity Data Store. The Identity Broker attaches consent information and tokens to the user entries as operational attributes. The *Separate Policy and Consent Store* maintains its own shadow set of user entries, made up of specific attributes (e.g., user ID, consent, and token data) in the consent store. The shadow user entries can be created using the UnboundID Data Sync server, or can be created when the Identity Broker first writes consent or token data for the user.

17. First, enter the location for Identity Broker. The **`location`** property is used to designate the server's physical location. By default, the Identity Broker prefers to

fail over to other servers in the same location or datacenter. Locations, typically, refer to the city where the data center resides.

```
Enter a location name for this Identity Broker: austin
```

18. Next, define the failover locations for other Identity Broker servers if more than one will be configured. For this example, type "no" (default is "yes") since this is the first Identity Broker instance. You can define the failover locations later when the clone Identity Broker (failover) servers are installed.

```
Do you want to define a list of preferred failover locations for the location 'aus-
tin'? (yes / no) [yes]: no
```

19. Next, install the standalone web apps that can be deployed directly on the Identity Broker. At a minimum, you will need to deploy the Identity Broker Console. Note that the Privacy Preferences web app requires a SCIM interface to the user store, which was configured in a previous section when the Identity Data Store was set up. For this example, press Enter to accept the default (All of the Applications).

```
    1)  Identity Broker Console
    2)  Privacy Preferences / Customer Support Portal
    3)  All of the applications
    4)  None of the applications

    b)  back
    q)  quit

Which applications do you want to deploy on the local Identity Broker? [3]:
```

20. The Identity Broker will set up an additional HTTPS connection handler for the Broker Console and other web applications. Enter an HTTPS port to be used for the Broker Console.

21. Enter the context path for the Identity Broker Admin Console. For this example, press Enter to accept the default (/broker-console).

```
Enter the HTTPS port to use for deploying web applications on the Identity Broker
[10443]:
Enter the context path to use for the Identity Broker Console [/broker-console]:

The Identity Broker Console will be available at https://example.com:10443/broker-con-
sole
```

22. Next, enter the SCIM URL that you set up on the Identity Data Store. For example, **`https://<hostname>:8443/`**. This allows the Identity Broker to access the user data over SCIM.

```
Enter the SCIM service URL for the user store: https://<hostname>:8443/
```

23. Enter a login name for interacting with the SCIM service. For example, "admin". Then, enter password credentials ("password") for the account. The login name and credentials are used so that the Privacy Preferences application can access the user data over SCIM. Note that the account is not used by any users to log into the Privacy Preferences client app. The account must be present in the user store and only requires READ access. The Privacy Preferences app will not attempt to write or update any user entries over SCIM.

```
Enter the login name for interacting with the SCIM service on the user store: admin
Enter the password for 'admin':
Confirm the password for 'admin':

The Privacy Preferences app will be available at https://example.com:10443/privacy-
preferences
```

24. Next, set up the Policy Store Access credentials so that the Identity Broker can communicate the Policy Store. Press Enter to accept the default **(cn=Broker, cn=Root DNs,cn=config**), and then enter the password for the credentials. The Policy Store holds the policies, application registration and configuration information to make access decisions. The Policy Store must be stored on an UnboundID Identity Data Store or a set of Identity Data Store servers fronted by an UnboundID Identity Proxy.

```
>>>> Step 3 of 11: Policy Store Access Credentials

This step allows you to specify the credentials that the Identity Broker should use
when communicating with the policy store. For simplicity, this tool assumes that all
policy store instances will use the same credentials.

    1)  Use cn=Broker User,cn=Root DNs,cn=config
    2)  Use a different account

    b)  back
    q)  quit

Choose an option [1]:

Enter the password for 'cn=Broker User,cn=Root DNs,cn=config':
Confirm the password for 'cn=Broker User,cn=Root DNs,cn=config':
```

25. Specify the type of security that the Identity Broker uses when communicating with the Policy Store instances. Press Enter to accept the default (SSL).

26. Enter the **SSL host:port** of the first Identity Data Store server in the Policy Store. For this example, enter **localhost:1636**.

27. Next, specify the base DN where the policy data will be stored on the backend Identity Data Store server. Press Enter to accept the default **(ou=Identity Broker,dc=example,dc=com**).

```
Specify the base DN where the policy data should be stored [ou=Identity Broker,dc=e-
xample,dc=com]:
```

28. Enter "yes" to configure a Unified Policy and Consent Store, where both stores are housed on one Identity Data Store backend server. If you want to configure a Separate Policy and Consent Store, enter "no". For this example, press Enter to accept the default (yes).

29. Specify the base DN where the user entries are stored on the Consent Store. For this example, press Enter to accept the default. The **`setup`** tool configures a default username regular expression mapper based on the **`uid`** attribute to associate a consent to a user entry.

```
Specify the base DN where the user entries are stored in the consent store:

    1)  Use ou=people,dc=example,dc=com
    2)  Use a different base DN

    b)  back
    q)  quit

Choose an option [1]:
```

30. Next, enter the **`uid`** name of the policy store administrator entry. This policy store administrator entry will be created under **`ou=people,dc=example,dc=com`** for managing the Consent Store to allow the Identity Broker Console and **`broker-admin`** tool to access the store. For example, enter "admin" and then, enter the password for this account.

```
An account entry will be created under ou=people,dc=example,dc=com for managing the
policy store by users of tools such as the Identity Broker Console and broker-admin
tool.  Enter the name (uid) of the entry to be created [admin]:

Enter the password for 'admin':
Confirm the password for 'admin':
```

31. The last phase of the install process will prepare and test the connections between the Identity Broker and backend stores. This phase checks that the broker user account is configured on the database backend and checks the communication with the servers. If you want to skip this step, you can run the **`prepare-external-store`** tool at a later time. For this example, press Enter to prepare all servers (3).

```
Would you like to prepare policy store localhost:1636 for access by the Identity
Broker?

    1)  Yes
    2)  No
    3)  Yes, and all subsequent servers
    4)  No, and all subsequent servers
```

```
    b)  back
    q)  quit

Enter choice [3]:
```

32. Next, review the certificate. If you trust it, enter "y". The certificate will be added to the **config/truststore** file.

33. Next, create the root user **"cn=Broker User,cn=Root DNs,cn=config** account on the backend Identity Data Store server. The Identity Broker uses this account to access the Identity Data Store. Press Enter to accept the default (yes).

```
Would you like to create or modify root user 'cn=Broker User,cn=Root DNs,cn=config'
so that it is available for this Identity Broker? (yes / no) [yes]:
```

34. Enter the DN and password of the root user to manage the **"cn=Broker User, cn=Root DNs,cn=config** account. The Identity Broker sets up the DN and tests that it can access the account. The **setup** tool then imports the Broker Schema and Policy Structure, verifies the backends as well as admin accounts on the backend data store.

```
Enter the DN of an account on localhost:1636 with which to create or manage the
'cn=Broker User,cn=Root DNs,cn=config' account and configuration [cn=Directory Man-
ager]:

Enter the password for 'cn=Directory Manager':

Created 'cn=Broker User,cn=Root DNs,cn=config'

Testing 'cn=Broker User,cn=Root DNs,cn=config' access ..... Done


Testing 'cn=Broker User,cn=Root DNs,cn=config' privileges ..... Done
Importing Broker Schema ..... Done
Importing Policy Structure ..... Done
Verifying backend 'ou=Identity Broker,dc=example,dc=com' ..... Done
Enabling Short Unique ID Virtual Attribute ..... Done
Verifying backend 'ou=people,dc=example,dc=com' ..... Done
Creating Policy Store Admin ..... Done
```

35. Next, the **setup** tool provides an option to install a few general-purpose policies that are ready for use or can be used as a starting point in configuring your own policies. Press Enter to accept the default (yes).

```
Do you want to install the UnboundID-Broker-provided policies?

    1)  Yes
    2)  No

    b)  back
    q)  quit
```

```
Enter choice [1]:
```

36. If you want to load sample data to see how the Identity Broker works, enter "1" for "yes". If not, you can load the data at a later time using the **sample-data-loader** tool.

37. Review the configuration summary, and then press Enter to accept the default (w) to write the configuration to a **dsconfig** batch file. The configuration is written to **<server-root>/broker-cfg.txt** . Certificate files are written to **external-server-certs.zip**. You can copy the certificates onto a failover Identity Broker instance when setting it up. The servers trust store file is at **config/truststore** and its password is stored at **config/keystore.pin**.

```
>>>> Step 10 of 10: Configuration Summary

  Identity Broker Admin Services:        https://example.lab:9443/auth/api/v1
  Identity Broker Policy Services:       https://example.lab:9443/pdp/v1
  Identity Broker OAuth2 Services:       https://example.lab:9443/oauth
  Identity Broker Admin Console:         https://example.lab:10443/broker-console
  Identity Broker Privacy Preferences:   https://example.lab:10443/privacypreferences

  Identity Broker Location: austin

  Policy Store
    Base DN: ou=Identity Broker,dc=example,dc=com
    Broker User DN: cn=Broker User,cn=Root DNs,cn=config
    Connection Security: SSL
    Servers: localhost:1636

  Consent Store
    Unified with Policy Store
    Base DN: ou=people,dc=example,dc=com
    UserName Mapper: "UID" Regex Mapper

  b)  back
  q)  quit
  w)  write configuration file

Enter choice [w]:
```

38. Apply the configuration to the local Identity Broker. Press Enter to accept the default (yes).

```
This tool can apply the configuration changes to the local Identity Broker. This
requires any configured Server SDK extensions to be in place. Do you want to do this?
(yes / no) [yes]:
```

39. This completes the initial configuration for the Identity Broker. Run the **bin/status tool** to see that the Identity Broker server is up and running.

You have successfully installed a Privacy Suite, featuring the UnboundID Identity Broker. You can now test the various components. See "Testing" on page 41

To Install a Clone Identity Broker

You can configure a clone Identity Broker instance to set up a failover server for your system. The first Identity Broker is called the peer server; the new server that you will be adding to the topology will be called the cloned server.

1. Unpack the zip distribution. Make sure to unpack the software in a folder different from the peer Identity Broker.
2. Run the **`setup`** command in the **`<server-root>`** directory of the cloned server.
3. Accept the licensing agreement by typing: yes.
4. Enter yes to add this server to an existing Identity Broker topology.
5. Enter the host name of the peer Identity Broker server from which the configuration will be copied.
6. Enter the port of the peer Identity Broker.
7. Enter the security communication to the peer Identity Broker.
8. Enter the manager account and password for the peer Identity Broker. Default is cn=Directory Manager.
9. Enter the fully-qualified host name or IP address of the local host.
10. Enter the HTTPS client connection port for the Identity Broker.
11. Enter the LDAP client connection port for the Identity Broker.
12. Enter yes if you want to enable LDAPS.
13. Enter yes if you want to enable StartTLS.
14. Enter the server certificate options: 1) self-signed certificate for testing-purposes only, 2) JKS certificate, 3) PKCS12 certificate, or a 4) PKCS11 token.
15. Enter yes if you want to specify particular addresses on which the server will listen to client connections.
16. Enter yes if you want to tune the JVM memory for performance. If yes, you will be prompted to enter the amount of memory to allocate to the JVM.
17. Enter yes if you want to start the server after the server has been configured.
18. Review the information that you set up for the configuration, and press enter to set up the server with the parameters shown. Or if you want to provide the setup parameters again, enter "2", or enter "3" to cancel the setup procedure and try again at a later time.

19. If you customized the **spring-security-config.xml** file, manually copy the **config/spring-security-config.xml** file from the peer server to the new cloned server. The file is used to configure Spring Security to authenticate using specific authentication providers and against alternate user repositories. If the server has already started, you must restart the server after you have manually copied the file.

To Install the Identity Broker with an Existing Truststore

If you want to use an existing keystore and truststore for your Identity Broker deployment, you can run the **setup** tool, then run the **create-initial-broker-config** separately. The following procedures runs **setup** from the command-line in non-interactive mode. You can also run it interactively, but do not run the **create-initial-broker-config** tool during the same session.

1. On the Identity Broker, run **setup** non-interactively from the command line. In this example, we assume the keystore and truststore passwords are the same.

```
./setup --cli --no-prompt --acceptLicense \
--ldapPort 2389 --ldapsPort 2636 --httpsPort 8443 --rootUserPassword password \
--useJavaTrustStore ~/tmp/keystores/truststore.jks \
--useJavaKeystore ~/tmp/keystores/broker1keystore.jks \
--trustStorePasswordFile ~/tmp/keystores/password.txt \
--keystorePasswordFile ~/tmp/keystores/password.txt \
--certNickname server-cert
```

2. Run the **create-initial-broker-config** tool non-interactively from the command line. The keystore and truststore passwords are the same. You must provide both the **--brokerTrustStorePath** and the **--trustStorePath** with their respective passwords. The **create-initial-broker-config** tool invokes the **prepare-external-store** tool to set up the communication between the Identity Broker and the datastores. If you run the **prepare-external-store** tool at a later time, you must include the **--brokerTrustStorePath** argument.

```
./bin/create-initial-broker-config \
--brokerTrustStorePath ~/tmp/keystores/truststore.jks \
--brokerTrustStorePasswordFile ~/tmp/keystores/password.txt \
--trustStorePath ~/tmp/keystores/truststore.jks \
--trustStorePasswordFile ~/tmp/keystores/password.txt \
--keyStorePath ~/tmp/keystores/broker1keystore.jks \
--keystorePasswordFile ~/tmp/keystores/password.txt
```

Chapter 5 Configuration

During the setup process, the Identity Broker's **`setup`** tool invokes the **`create-initial-broker-config`** script, configuring the communication between the Identity Broker and its repositories, testing the connections, and setting up the admin accounts necessary for the Identity Broker to access each data store.

Once you have completed the initial configuration, you can fine-tune the server further using the **`dsconfig`** tool to set the properties needed for your deployment. For a summary of each component and its properties, see the *UnboundID Identity Broker Configuration Reference* (HTML).

This chapter presents additional configuration steps to get your system up and running.

About the Command-Line Tools

The **`setup`** command installs Broker Admin Console, a web application that administrators can use to manage the Identity Broker and a number of command-line tools that access the Identity Broker's endpoints. The command-line tools provide the same functionality as the Broker Console GUI and can be used for easy scripting. Each command-line tools provides a Help options with examples, which you can view by including the **`--help`** or **`--help-subcommands`** arguments. For example, you can get a list of all commands using the **`--help`** argument, a list of all sub-commands using the **`--help-subcommand`** argument, and a detailed help for a single subcommand using the **`--help`** argument with the subcommand name.

```
$ bin/broker-admin --help
$ bin/broker-admin --help-subcommands
$ bin/broker-admin update-policy-template --help
```

The following tools invoke the Identity Broker REST API to manage the various Identity Broker components:

- **broker-admin**. Runs administrative operations.
- **consent-admin**. Runs consent management operations.
- **evaluate-policy**. Requests a policy decision from the Identity Broker.
- **oauth2-request**. Tests token functions of the Identity Broker and manages OAuth2 tokens on behalf of a registered application.

Configuring Policies

A policy is a set of rules that decide if a particular client application can gain access to a user's resource data. During the setup process, you are prompted to import sample policies that you can view and try out. If you did not import the sample policies, you can configure them manually using the **`broker-admin`** command-line tool.

To Configure a Policy Using the Command-Line Tools

The following example procedures are command-line equivalents of the sample data that was installed during the setup process. These steps show how to configure a policy, scopes, applications, and resources needed to run the Policy Service. You can create these elements as described below or using the Broker Console GUI.

1. **Create the actions**. *Actions* determines the operation that the application intends to perform on the resource, such as "read" or "write".

```
broker-admin create-action \
--set name:Read \
--set description:"Allows the requester to read a resource." \
--trustAll --authID admin --authPassword admin
```

2. **Create the tags**. *Tags* are descriptive properties that you assign to your applications or resources for easier organization and management. Policies use tags, for example, by allowing you to define a single Policy for related applications or resources. For example, you can create a HIPAA (Health Insurance Portability and Accountability Act) tag, which can be used for any resource or application that accesses patient information. Tags can affect policy decisions and OAuth2 flows.

```
broker-admin create-tag \
--set name:External \
--set description:"Tag indicating external application or \
resource that is available to external apps." \
--trustAll --authID admin --authPassword admin
```

```
broker-admin  create-tag \
--set name:Internal \
--set description:"Tag indicating internal application or resource \
that is available only to internal apps." \
--trustAll --authID admin --authPassword admin
```

3. **Create the trust levels**. *Trust levels* are values you assign to applications and resources to indicate how much access a particular application can have or how sensitive a particular resource is. Applications that wants access to a particular resource must have the same trust level or higher as the resource. For example, an internally developed application might have a higher trust level than a 3rd party application. Likewise a user's social security number might have a higher trust level than something like a user's email address. Trust levels can range from a numerical value of 1 to 1000, providing fine-grain levels if needed.

```
broker-admin create-trust-level \
--set name:Low \
--set value:1 \
--set description:"This is an untrusted application." \
--trustAll --authID admin --authPassword admin

broker-admin create-trust-level \
--set name:Medium \
--set value:2 \
--set description:"This is a partially-trusted application." \
--trustAll --authID admin --authPassword admin

broker-admin create-trust-level \
--set name:High \
--set value:3 \
--set description:"This is well-trusted application." \
--trustAll --authID admin --authPassword admin

broker-admin create-trust-level \
--set name:"Very High" \
--set value:4 \
--set description:"This is unlimited-trust application." \
--trustAll --authID admin --authPassword admin
```

4. **Create the resources**. A *resource* is a collection, attribute or thing that represents an organizational data asset; e.g., "Person", "Phone Number", "Social Security Number" and "Email".

```
broker-admin create-resource \
--set "name:Billing History" \
--set urn:"urn:example:resource:billing-history" \
--set tagIds:name=Internal \
--set trustLevelId:name=Medium \
--set description:"The user's billing history." \
--trustAll --authID admin --authPassword admin

broker-admin create-resource \
--set "name:Customer Preferences" \
```

```
--set urn:"urn:example:resource:customer-preferences" \
--set trustLevelId:name=Medium \
--set description:"The customer preferences" \
--trustAll --authID admin --authPassword admin

broker-admin create-resource \
--set "name:Customer Profile" \
--set urn:"urn:example:resource:customer-profile" \
--set trustLevelId:name=Medium \
--set description:"The User resource. Includes all user attributes." \
--trustAll --authID admin --authPassword admin
```

5. **Create the resource groups**. A *resource group* is a collection of resources so that a policy can be applied to the group.

```
broker-admin create-resource \
--set name:"Consumer Data" \
--set urn:"urn:example:group:consumer-data" \
--set memberResourceIds:name="Customer Profile" \
--set memberResourceIds:name="Customer Preferences" \
--set description:"Resource group containing consumer data" \
--trustAll --authID admin --authPassword admin
```

6. **Create resource aliases**. *Aliases* are labels that you can assign a resouce for easier recall and usage.

```
broker-admin create-resource-alias \
--set urn:"urn:example:resource:customer-preference  \
--set resourceId:name="Cust Preference" \
--trustAll --authID admin --authPassword admin

broker-admin create-resource-alias \
--set urn:"urn:example:resource:billing-history" \
--set resourceId:name="Bills" \
--trustAll --authID admin --authPassword admin

broker-admin create-resource-alias \
--set name:"urn:example:resource:customer-profile"" \
--set resourceId:name="Profile" \
--trustAll --authID admin --authPassword admin
```

7. **Create the purposes**. A *purpose* specifies the application's intended use of the resource. For example, "Marketing", "Billing Verification", "Credit Check". Policies can be written to allow or deny requests based on their purpose.

```
broker-admin create-purpose \
--set name:Marketing \
--set description:"Application-specific marketing promotions \
highlighting new releases, pertinent features, best practices, etc." \
--trustAll --authID admin --authPassword admin

broker-admin create-purpose \
--set name:Any \
--set description:"Wild card that matches any purpose." \
--trustAll --authID admin --authPassword admin
```

8. **Create the scopes**. A *scope* indicates the resource, action, and purpose of an OAuth2 request. Policy requests specify these three elements directly instead of using a scope.

```
broker-admin create-scope \
--set name:"urn:unboundid:scope:ConsumerData" \
--set actionId:name=Read \
--set purposeId:name="Marketing" \
--set resourceIds:name="Consumer Data" \
--set description:"Scope that provides read access to apps." \
--trustAll --authID admin --authPassword admin
```

9. **Create the applications**. An *application* is the software program or system that requests authorization to access the resource. Applications can be web apps, mobile apps, social apps, or partner apps for B2B operations.

```
broker-admin create-application \
--set name:"ExternalApplicationTwo" \
--set emailAddress:"externalApplication@example.com" \
--set redirectUrls:"https://www.example.com/externalApplicatoin/redirect" \
--set url:"http://www.example.com/externalApplication" \
--set iconUri:"/assets/img/appicons/extapp.png" \
--set tagIds:name=External \
--set trustLevelId:name=Low \
--set grantTypes:authorization_code \
--set grantTypes:password \
--set accessTokenValiditySeconds:15 \
--set refreshTokenValiditySeconds:86400 \
--set restrictScopes:true \
--set scopeIds:name="ConsumerData" \
--set description:"External application" \
--trustAll --authID admin --authPassword admin

broker-admin create-application \
--set name:"InternalApplicationOne" \
--set emailAddress:"internalApplication@example.com" \
--set redirectUrls:"https://www.example.com/internalApplicatoin/redirect" \
--set url:"http://www.example.com/internalApplication" \
--set iconUri:"/assets/img/appicons/intapps.jpg" \
--set tagIds:name=Internal \
--set trustLevelId:name=Low \
--set grantTypes:authorization_code \
--set grantTypes:refresh_token \
--set grantTypes:password \
--set restrictScopes:true \
--set scopeIds:name="ConsumerData" \
--set description:"Trusted internal application" \
--trustAll --authID admin --authPassword admin
```

Note that when you create an application and specify a grant type, the Identity Broker generates a client ID and a client secret, which is unique to the application.

```
Property                    Value(s)
--------------------------------------------------------------------------------
id                          GKE1
```

```
deletable                    true
editable                     true
name                         ExternalApplicationTwo
description                  This is an untrusted external application.
emailAddress                 externalApplication@example.com
url                          http:/www.example.com/externalApplication
iconUri                      -
tags                         -
tagIds                       -
clientId                     969c1dec-e446-43fd-8fce-101258deac1c
clientSecret                 hAaix0l9Wm
hidden                       false
redirectUrls                 http:/www.example.com/externalApplication/redirect
grantTypes                   authorization_code
grantTypes                   password
accessTokenValiditySeconds   -
refreshTokenValiditySeconds  -
consentValiditySeconds       -
scopeIds                     GKE1
scopes                       GKE1:ConsumerData
trustLevelId                 -
trustLevel                   -
bypassAuthzApproval          false
restrictScopes               true
```

10. **Create application groups**. An *application group* is a collection of applications for easier designation and management. For example, "Internal Applications" and "Partner Applications".

```
broker-admin create-application-group \
--set name:"External Application" \
--set applicationGroupMemberIds:name="ExternalApplicationTwo" \
--set description:"This group containing external application." \
--trustAll --authID admin --authPassword admin

broker-admin create-application-group \
--set name:"Internal Applications" \
--set applicationGroupMemberIds:name="InternalApplicationOne" \
--set description:"This group containing internal application." \
--trustAll --authID admin --authPassword admin
```

11. **Create the policy sandboxes**. A *policy sandbox* can test a policy or policy set beore enabling them for actual deployment to ensure that the definitions have the intended effect.

```
broker-admin create-policy-sandbox \
--set name:"OAuth Policy Sandbox" \
--set includeEnabledPolicies:true \
--set excludedPolicyIds:name="Basic Consent Policy" \
--set description:"Policy sandbox used during OAuth authorization requests" \
--trustAll --authID admin --authPassword admin
```

Adding Additional Admin Accounts

The UnboundID Identity Broker 4.1 does not currently provide any tools to add additional administrator accounts. If your admin acounts are stored in an LDAP directory server, you can create additional admin accounts by creating an LDIF file and then manually adding the entries to the server. You must also include the required roles in the entry. See "About the Identity Broker Roles and Scopes" on page 58.

The only consideration about creating additional admin users is that the configuration be placed where the Username Mapper configuration can find it. If it is placed out-of-the-box in the same location as peers of the original admin user, for example, **`ou=people,dc=example,dc=com`**. However, if you change the Username Mapper configuration or change where the users are stored, you may need to create the new admin entries elsewhere.

An example is presented below.

To Add Additional Administrator Accounts

1. Create an LDIF entry using a text editor include the role or authority to the entry. Save the file as `new-admin.ldif`. The new admin account is a peer to the original admin (**`uid=admin`**) located in the same branch. This allows the UserName Mapper to locate the new admin account with minimum additional configuration.

```
dn: uid=admin-2,ou=People,dc=example,dc=com
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetorgperson
description: This is the administrative user #2 for the UnboundID Identity Broker
uid: admin
cn: Broker
sn: Admin
userPassword: password
id-broker-admin-privilege: broker_admin
```

2. On the user store backend, add the entry to your database. Make sure to include your bind parameters depending on your setup (not shown below).

```
$ bin/ldapmodify -a -f new-admin.ldif
```

Running the sample-data-loader Tool

During the setup process, the **create-initial-broker-config** tool prompts you if you would like to have sample data installed on the Identity Broker for demonstration purposes. If you selected the default ("no"), you can run the **sample-data-loader** tool to install the sample data at a later time.

The **sample-data-loader** tool provides an **install** subcommand to install the sample data and a **remove** subcommand to delete the sample data after you have run your demonstrations.

Note that the **sample-data-loader** tool requires the two owner parameters to be present during its invocation using existing users in the data store. For example, the **create-initial-broker-config** session installs two internal users, **sampleuser1** and **sampleuser2**, which are used to install the sample data. If you are adding the sample data after running the **create-initial-broker-config** tool, you can add these users manually prior to running **sample-data-loader**. The example procedure shows how to do so below.

To Run the sample-data-loader Tool

1. On the backend user store, add two internal entries, **sampleuser1** and **sampleuser2**, to be used with the **sample-data-loader** tool. Or, you can use two existing user accounts with the **sample-data-loader**. The following example shows the LDIF file that you can create using a text editor, and then using the **ldapmodify** tool.

```
dn: uid=sampleuser1,ou=People,dc=example,dc=com
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetorgperson
description: This is a test user to exercise sample data within the UnboundID Iden-
tity Broker
uid: sampleuser1
cn: Sample
sn: User1
userPassword: password

dn: uid=sampleuser2,ou=People,dc=example,dc=com
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetorgperson
description: This is a test user to exercise sample data within the UnboundID Iden-
tity Broker
uid: sampleuser2
cn: Sample
```

```
sn: User2
userPassword: password

bin/ldapmodify -p 1389 -D "uid=admin,dc=example,dc=com" -w passord -a -f sample-
data.ldif
```

2. On the Identity Broker, run the **sample-data-loader** tool to install the sample data onto your Identity Broker system.

```
sample-data-loader install \
--trustAll --authID admin --authPassword password \
--owner1 sampleuser1 --owner2 sampleuser2 --no-prompt
```

3. After you have finished running the demonstration, run the **sample-data-loader** tool to remove the sample data from your Identity Broker system.

```
sample-data-loader remove \
--trustAll --authID admin --authPassword password \
--owner1 sampleuser1 --owner2 sampleuser2 --no-prompt
```

Creating Trace Filters

Administrators or support staff can troubleshoot production policies that may not generate the correct decisions. Trace filters use the XACML **target** element to define conditions on the subject, resource, and action that determine whether the policy or rule applies to a request.

Targets define simple conditions on the subject, resource, action, and environment that partly determine whether the policy, policyset, or rule applies to a request.

To Create a Trace Filter

1. Open a text editor and create a trace filter file. In this example, set the file as "test-trace-filter". A trace filter is a XACML **target** element. The following example traces all requests from the application, ExternalApplicationTwo.

```
<?xml version="1.0" encoding="UTF-8"?>
<Target xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17">
  <AnyOf>
    <AllOf>
      <Match
        MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
          <AttributeValue Data-
Type="http://www.w3.org-
/2001/XMLSchema#string">ExternalApplicationTwo</AttributeValue>
            <AttributeDesignator
```

```
             MustBePresent="false"
               Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-sub-
ject"
               AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
         DataType="http://www.w3.org/2001/XMLSchema#string"/>
      </Match>
    </AllOf>
  </AnyOf>
</Target>
```

2. Configure the trace filter using the **`broker-admin`** tool.

```
bin/broker-admin create-trace-filter -u admin -w pass \
--set name:test-trace-filter  --set enabled:true \
--set description:"Test trace filter" \
--setFromFile targetXml:test-trace-filter.xml
```

The command returns the following message:

```
Created trace-filter '8AP:Test Filter #1'

Property     Value(s)
-----------------------------------------------------------------------------------------
id           8AP
deletable    true
editable     true
name         Test Filter #1
enabled      true
targetXml    <?xml version="1.0" encoding="UTF-8"?>
<Target xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17">
 <AnyOf>
   <AllOf>
     <Match
       MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
         <AttributeValue Data-
Type="http://www.w3.org/2001/XMLSchema#string">ExternalAppTwo</AttributeValue>
           <AttributeDesignator
               MustBePresent="false"
               Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"
               AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
               DataType="http://www.w3.org/2001/XMLSchema#string"/>
      </Match>
    </AllOf>
  </AnyOf>
</Target>

description  Test Filter Example
```

3. Verify the trace filter.

```
bin/broker-admin list-properties -u admin -w pass
```

Chapter 6 Testing

Once you have configured the UnboundID Identity Broker, you can test the various components installed during the setup process.

Testing the Sample Policies

During the setup process, you are presented with the option to install some sample data to allow you play around with the Identity Broker's Policy Service. Four sample Policy Tests are available for demonstration:

- External App Request for Billing History
- External App Request for Customer Profile
- Internal App Request for Billing History
- Internal App Request for Consumer Data

To Test the Sample Policies

1. Login into the Broker Console.
2. On the Administration Home page, click Policy Tests.
3. On the Policy Tests page, look over the sample policies. Each sample policy describes the expected result, so that you can see the output. The Policy Tests assume that you are the Policy Administrator who has created the policies and are now testing them. As a result, the output displays a verbose output.
4. On the Policy Tests page, click the down arrow next to the Edit button for the External App request for Billing History, and then select **Run Test**. In this example, the ExternalAppTwo is requesting to see the Billing History resource

for the **sampleuser1**, who has given consent to let the app view his billing history.

5. On the Policy Test dialog, click Run Test. The result is displayed showing a decision of "Deny" on the fourth line.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns2:Response xmlns:ns2="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17">
    <ns2:Result xsi:type="ResultType" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
        <ns2:Decision>Deny</ns2:Decision>
            <ns2:Status>
                <ns2:StatusCode Value="urn:oasis:names:tc:xacml:1.0:status:ok"/>
            </ns2:Status>
...(more output not shown)...
```

6. Run the other Policy Tests to view the decision.

Testing Consent

The UnboundID Identity Broker features *informed consent*, offering your customers the transparency over their application consent history and the context for that consent in exchange for access to their resources. Your customers can give their consent to share their data with third-party applications and actively manage or revoke their consent at any time. They have complete control over which data third-party application can use and for how long. This informed consent improves customer service and increases brand-loyalty through targeted marketing, while strictly adhering to regulatory standards on privacy.

This section shows how to test consent on the Identity Broker after you have installed it.

To Test Consent

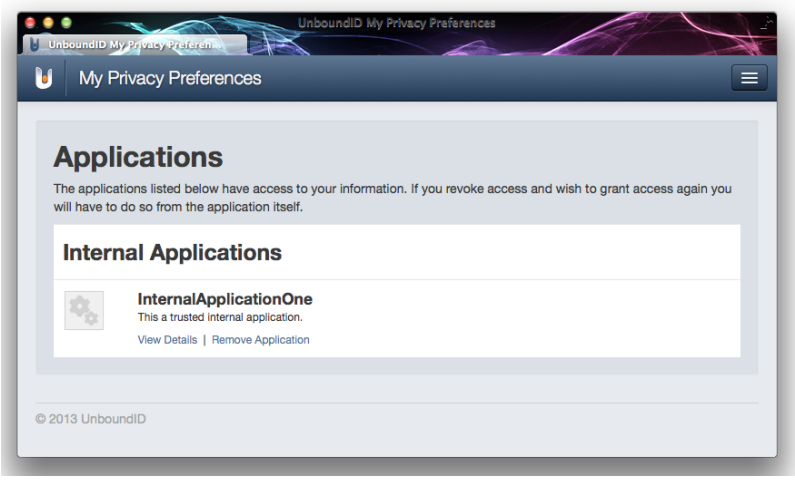
The following procedure assumes that **user.1** is an existing user account in the user store. It also assumes the sample data has been loaded in an earlier process.

1. Use the **consent-admin** tool to create a consent. The following command creates a consent for **user.1** to allow the application InternalApplicationOne to have "Read" access to the "Billing History" resource for "Any" purpose.

```
./consent-admin add-consent \
--authId admin --authPassword admin --trustAll \
```

```
--purpose Any --application InternalApplicationOne \
--owner user.1 --resource "Billing History" --action Read
```

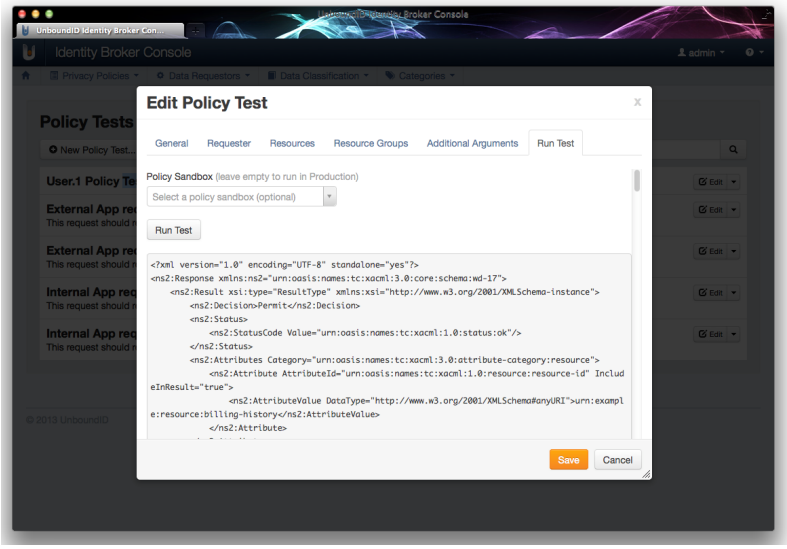
2. Log into the Privacy Preferences GUI as **`user.1`** to view the consent.



3. Log into the Broker Console GUI. Click the Privacy Policies tab, click Policy Tests, and then click New Policy Test. You will create a new policy test that uses the consent. On the Create Policy Test dialog, use the following parameters:

   - **General Tab**: Name=User.1 Policy Test
   - **Requestor Tab**: Subject=InternalApplicationOne, Action=Read, Purpose=Any, Owner=user.1
   - **Resources Tab**: Billing History
   - **Run Test Tab**: Click Save.

4. On the Run Test tab, click the Run Test button. The XACML response to the authorization request appears and includes a "Permit" decision on the fourth line.



5. To see a "Deny" decision, edit the policy test. On the Requestor tab, change the Requestor to ExternalApplicationTwo and save the policy test. On the Run Test tab, click Run Test. You should see a "Deny" decision on the fourth line.

Testing the OAuth2 Authorization Flows

The Identity Broker provides a REST-based **`broker-admin`** command-line tool that you can use to manage the Identity Broker, policies, and consents.

Creating Resources, Purposes, Actions, Scopes and Applications

Before an OAuth2 authorization flow can be tested, you must add a number of objects to the Identity Broker's policy store:

- **Resources**. Specifies data or attributes belonging to resource owners.
- **Purposes**. Specifies reasons for accessing a resource.
- **Actions**. Specifies ways in which resources will be used.
- **Scopes**. Specifies the composite objects representing the combination of a purpose, an action, and a resource.
- **Applications**. Specifies the entities that will request access to a resource owner's resources. Also, referred as "clients".

To Create Resources, Purposes, Actions, Scopes, and Apps

The following procedures show how to create resources, purposes, actions, and applications. The connection arguments are omitted from these command line examples for clarity.

1. First, create a resource.

```
$ bin/broker-admin create-resource --set "name:TestResource" \
--set "description:Resource for testing the Identity Broker." \
--set "urn:com.example.testresource"

Created resource 'Z3F8:TestResource'

Property           Value(s)
-----------------------------------------------------------
id                 Z3F8
deletable          true
```

```
editable           true
name               TestResource
urn                urn:com.example.testresource
description        Resource for testing the Identitiy Broker.
memberResources    -
memberResourceIds  -
tags               -
tagIds             -
trustLevelId       -
trustLevel         -
```

2. Create a purpose.

```
$ bin/broker-admin create-purpose --set "name:TestPurpose" \
--set "description:For testing the Identity Broker"

Created purpose 'NE74:TestPurpose'

Property     Value(s)
-------------------------------------------
id           NE74
deletable    true
editable     true
name         TestPurpose
description  For testing the Identity Broker
default      false
```

3. Create an action.

```
$ bin/broker-admin create-action --set "name:TestAction" \
--set "description:Allows the requester to test a resource."

Created action '3ZL5:TestAction'

Property     Value(s)
-----------------------------------------------------
id           3ZL5
deletable    true
editable     true
name         TestAction
description  Allows the requester to test a resource.
```

4. Create a scope using the above resource, purpose, and action. By convention, scopes are named using URNs or URIs.

```
$ bin/broker-admin create-scope \
--set "name:urn:com.example.scope.test_resource" \
--set "purposeId:name=TestPurpose" \
--set "actionId:name=TestAction" \
--set "resourceIds:name=TestResource" \
--set "description:Represents the ability to use the \
      TestAction on the TestResource for the purpose TestPurpose."

Created scope '3JXB:urn:com.example.scope.test_resource'

Property     Value(s)
-----------------------------------------------------------------------------------------
```

```
id           3JXB
deletable    true
editable     true
name         urn:com.example.scope.test_resource
action       3ZL5:TestAction
actionId     3ZL5
resources    Z3F8:TestResource
resourceIds  Z3F8
purposeId    NE74
purpose      NE74:TestPurpose
description  Represents the ability to use the TestAction on the TestResource for the
purpose TestPurpose.
```

5. Create an application. This application will be allowed to use all OAuth2 grant types. Its default scope will be the scope defined in the previous step.

```
$ broker1/bin/broker-admin create-application \
--set "name:TestApplication" --set "grantTypes:authorization_code" \
--set "grantTypes:implicit" --set "grantTypes:client_credentials" \
--set "grantTypes:password" --set "grantTypes:refresh_token" \
--set "scopeIds:name=urn:com.example.scope.test_resource"

Created application 'FOT9:TestApplication'

Property                     Value(s)
-------------------------------------------------------------------------
id                           FOT9
deletable                    true
editable                     true
name                         TestApplication
description                  -
emailAddress                 -
url                          -
iconUri                      -
tags                         -
tagIds                       -
clientId                     a07bfe59-124a-4747-9037-e3e099b68203
clientSecret                 1RMGfNz38e
hidden                       false
redirectUrls                 -
grantTypes                   authorization_code
grantTypes                   implicit
grantTypes                   client_credentials
grantTypes                   password
grantTypes                   refresh_token
accessTokenValiditySeconds   -
refreshTokenValiditySeconds  -
consentValiditySeconds       -
scopeIds                     3JXB
scopes                       3JXB:urn:com.example.scope.test_resource
trustLevelId                 -
trustLevel                   -
bypassAuthzApproval          false
restrictScopes               true
```

To Test the OAuth2 Resource Owner Grant Type

The oauth2-request tool can be used to request and receive an OAuth access token using the resource owner grant type. In this example, we will use the credentials for the **`TestApplication`** created previously as the requesting application. For the resource owner, we will use a user in the user store with the name "user.1".

Assume that the user store is an UnboundID Data Store, and the Identity Broker's Consent Store configuration is set up to use the default "UID Mapper" username mapper. Note that "user.1" corresponds to the LDAP uid value of a user entry. For example, its DN may be "uid=user.1,ou=people,dc=example,dc=com".

1. Get the token from the resource owner.

```
$ bin/oauth2-request token-from-resource-owner-password \
--clientID a07bfe59-124a-4747-9037-e3e099b68203 \
--clientSecret 1RMGfNz38e \
--ownerId user.1 \
--ownerPassword password \
--displayToken \
--requestRefreshToken

Parameter      Value(s)
-----------------------------------------------------------------------------------
Access Token   MD2AAQGBBmg2OEVvUYIwkSy_c8CR_JO5PGom1hJtZBwo-AUgKe9WVq9qSGdbul-
GpcWvKgIHcGms7KOsGRE5
Token Type     bearer
Scope          urn:com.example.scope.test_resource
Expires In     11 hours, 59 minutes, 58 seconds
Expiration     20130419091849Z
Refresh Token  MD2AAQGBBmg2O-
EVvUYIwth72ANm0IgySRjG9a79GLxi2q7Iti497QMxzHpB7QNKGpcWvKgIHcGms7KOsGRE5
```

2. Confirm that token validation works.

```
$ bin/oauth2-request validate-token \
--clientID a07bfe59-124a-4747-9037-e3e099b68203 \
--clientSecret 1RMGfNz38e \
--token MD2AAQGBBmg2OEVvUYIwkSy_c8CR_JO5PGom1hJtZBwo-AUgKe9WVq9qSGdbul-
GpcWvKgIHcGms7KOsGRE5

Parameter   Value(s)
-----------------------------------------------
Username    user.1
Authority   ROLE_READ_OWN_CONSENT
Authority   ROLE_READ_OWN_ACCESSHISTORY
Authority   ROLE_READ_OWN_CONSENTHISTORY
Authority   ROLE_USER
Authority   ROLE_CREATE_OWN_CONSENT
Authority   ROLE_DELETE_OWN_CONSENT
Client ID   a07bfe59-124a-4747-9037-e3e099b68203
Scope       urn:com.example.scope.test_resource
Issued At   20130418211850Z
Expires In  11 hours, 45 minutes, 4 seconds
```

3. Confirm that the refresh token can be used to retrieve a new access token.

```
$ bin/oauth2-request token-refresh \
--clientID a07bfe59-124a-4747-9037-e3e099b68203 \
--clientSecret 1RMGfNz38e \
--sourceRefreshToken MD2AAQGBBmg2O-
EVvUYIwth72ANm0IgySRjG9a79GLxi2q7Iti497QMxzHpB7QNKGpcWvKgIHcGms7KOsGRE5 \
--displayToken

Parameter      Value(s)
-----------------------------------------------------------------------------------------
Access Token   MD2AAQGBBmg2OEVvUYIw7s_esgBtbID8NfLODw_vLfZsfrncStE_
AYB1hu51NySGpcWvKgIHcGms7KOsGRE5
Token Type     bearer
Scope          urn:com.example.scope.test_resource
Expires In     11 hours, 59 minutes, 58 seconds
Expiration     20130419093527Z
Refresh Token  MD2AAQGBBmg2O-
EVvUYIwth72ANm0IgySRjG9a79GLxi2q7Iti497QMxzHpB7QNKGpcWvKgIHcGms7KOsGRE5
```

To Test the OAuth2 Client Credentials Grant Type

The oauth2-request tool can also be used to retrieve an access token the OAuth 2.0 client credentials grant type. This grant type is used when the requesting application would like to access its own resources, so a separate set of owner credentials is not needed.

Note that refresh tokens cannot be requested using the client credentials grant type.

1. Obtain the token from client credentials.

```
$ broker1/bin/oauth2-request token-from-client-credentials \
--clientID a07bfe59-124a-4747-9037-e3e099b68203 \
--clientSecret 1RMGfNz38e --displayToken

Parameter     Value(s)
-------------------------------------------------------------------------------
Access Token  MC2AAQGBBmg2OEVvUYIg-XlYUSaBC4BxLoeOEwbl-CcC66tRPYvtWHo9-TxhyxM
Token Type    bearer
Scope         urn:com.example.scope.test_resource
Expires In    11 hours, 59 minutes, 58 seconds
Expiration    20130419093824Z
```

2. Validate the token.

```
$ broker1/bin/oauth2-request validate-token \
--clientID a07bfe59-124a-4747-9037-e3e099b68203 \
--clientSecret 1RMGfNz38e \
--token MC2AAQGBBmg2OEVvUYIg-XlYUSaBC4BxLoeOEwbl-CcC66tRPYvtWHo9-TxhyxM

Parameter   Value(s)
-----------------------------------------------
Client ID   a07bfe59-124a-4747-9037-e3e099b68203
Scope       urn:com.example.scope.test_resource
```

```
Issued At   20130418213824Z
Expires In  11 hours, 59 minutes, 8 seconds
```

To Test the OAuth2 Auth Code and Implicit Grant Types

The authorization code and implicit grant types are intended to be used interactively with a web browser; for this reason, the **`oauth2-request`** command line tool does not support those grant types.

Contact your authorized support provider or UnboundID on how you can test these grant types on your system.

Chapter 7 Management

The Identity Broker provides server management tools needed to run basic functions, such as stop, start, uninstall, and others. The tools are located in the server root directory or in the **bin** (or **bat**) directory of the server.

Running the Identity Broker

To start the Identity Broker, run the **bin/start-broker** tool on UNIX/Linux systems (the same command is in the **bat** folder on Windows systems).

To Run the Identity Broker

On the command line, run the following command.

```
$ bin/start-broker
```

To Run the Identity Broker in the Foreground

1. Enter the **bin/start-broker** with the **--nodetach** option to launch the Identity Broker as a foreground process.

   ```
   $ bin/start-broker --nodetach
   ```

2. You can stop the Identity Broker by pressing CNTRL-C in the terminal window where the server is running or by running the **bin/stop-broker** command from another window.

Stopping the Identity Broker

The Identity Broker provides a shutdown script, **bin/stop-broker**, to stop the server.

To Stop the Identity Broker

Use the **bin/stop-broker** tool to shut down the server.

```
$ bin/stop-broker
```

To Schedule a Server Shutdown

The Identity Broker provides the capability to schedule a shutdown and send a notification to the **server.out** log file. The following example sets up a shutdown task that is schedule to be processed on April 3rd, 2013 at 11:00pm CDT. The server uses the UTC time format if the provided timestamp includes a trailing "Z", for example, 201304032300Z. The example also uses a **--stopReason** option that writes the reason for the shutdown to the logs.

```
$ bin/stop-broker --task --hostname server1.example.com \
--bindDN uid=admin,dc=example,dc=com --bindPassword password \
--stopReason "Scheduled offline maintenance" --start 201304032300Z
```

To Run an In-Core Restart

You can re-start the Identity Broker using the **bin/stop-broker** command with the **--restart** or **-R** option. Running the command is equivalent to shutting down the server, exiting the JVM session, and then starting up again. Shutting down and restarting the JVM requires a re-priming of the JVM cache. To avoid destroying and re-creating the JVM, use an in-core restart, which can be issued over LDAP. The in-core restart will keep the same Java process and avoid any changes to the JVM options.

```
$bin/stop-broker --task --restart --hostname 127.0.0.1 \
--bindDN uid=admin,dc=example,dc=com --bindPassword password
```

Uninstalling the Identity Broker

The Identity Broker provides an **`uninstall`** tool provides an interactive method to remove the components from the system.

To Uninstall the Identity Broker

1. From the server root directory, run the **`uninstall`** command.

```
$ ./uninstall
```

2. Select the option if you want to remove all components or select the components to be removed. If you select remove all, then the server removes all of the components.

```
Do you want to remove all components or select the components to remove?

    1)  Remove all components
    2)  Select the components to be removed

    q)  quit

Enter choice [1]: 2
```

3. If you remove selected components, respond "yes" when prompted.

```
Remove Server Libraries and Administrative Tools? (yes / no) [yes]: yes
Remove Log Files? (yes / no) [yes]: no
Remove Configuration and Schema Files? (yes / no) [yes]: yes
Remove Backup Files Contained in bak Directory? (yes / no) [yes]: no
Remove LDIF Export Files Contained in ldif Directory? (yes / no) [yes]: no

The files will be permanently deleted, are you sure you want to continue? (yes / no)
[yes]:
```

4. Manually delete any remaining files or directories.

Chapter 8 Reference

The following chapter provides general reference about various files and components of the UnboundID Identity Broker.

About the Identity Broker Files and Folders

Once you have unzipped the Identity Broker distribution file, you will see the following folders and command-line utilities, shown in the table below.

Table 7. Layout of the Identity Broker Folders

Directories/Files/Tools	Description
LICENSE.txt	Licensing agreement for the Identity Broker.
README	README file that describes the steps to set up and start the Identity Broker.
bak	Stores the physical backup files used with the backup command-line tool.
bat	Stores Windows-based command-line tools for the Identity Data Store.
broker-cfg.txt	Stores the configuration history for the Identity Broker. Appears after you have configured the Identity Broker.
classes	Stores any external classes for server extensions.
collector	Stores collector files.
config	Stores the configuration files for the backends (admin, config) as well as the directories for messages, schema, tools, and updates.
docs	Provides the release notes, Configuration Reference file and a basic Getting Started Guide (HTML).
import-tmp	Stores temporary imported items.
ldif	Stores any LDIF files that you may have created or imported.
legal-notices	Stores any legal notices for dependent software used with the Identity

Table 7. Layout of the Identity Broker Folders(continued)

Directories/Files/Tools	Description
	Broker.
lib	Stores any scripts, jar, and library files needed for the server and its extensions.
locks	Stores any lock files in the backends.
logs	Stores log files for the Identity Broker.
metrics	Stores files for the UnboundID Metrics Engine.
resource	Stores the MIB files for SNMP.
revert-update	The revert-update tool for UNIX/Linux systems.
revert-update.bat	The revert-update tool for Windows systems.
setup	The setup tool for UNIX/Linux systems.
setup.bat	The setup tool for Windows systems.
tmp	Temp directory.
unboundid_logo.png	UnboundID logo
uninstall	The uninstall tool for UNIX/Linux systems.
uninstall.bat	The uninstall tool for Windows systems.
update	The update tool for UNIX/Linux systems.
update.bat	The update tool for Windows systems.
webapps	Stores the war files for reference implementations (privacy preferences and the admin console)

About the Identity Broker Tools

After you have installed and configured the Identity Broker, you can access the various tools in the **`bin`** directory of the server. The following is a brief description of each tools. Each tool provides its own help, which can be invoked online using the **`--help`** argument.

Table 8. Summary of Identity Broker Tools

Tool	Description
backup	Run full or incremental backups on one or more Identity Brokers. This utility also supports the use of a properties file to pass predefined command-line arguments. See Managing the tools.properties File
base64	Encode raw data using the base64 algorithm or decode base64-encoded data back to its raw representation.
broker-admin	Invokes administrative operations over the Identity Broker

Table 8. Summary of Identity Broker Tools(continued)

Tool	Description
	REST API. The tool provides commands that allow you to invoke admin operations to manage the Identity Broker.
collect-support-data	Collect and package system information useful in troubleshooting problems. The information is packaged as a ZIP archive that can be sent to a technical support representative.
consent-admin	Manage a resource owner consent. This tool provides commands that allow you to invoke consent management operations over the Identity Broker REST API. Consent is authorized by a resource owner to allow access to resources by an application.
create-initial-broker-config	Create an initial Identity Broker configuration.
create-rc-script	Create a Run Control (RC) script that can be used to start, stop, and restart the Identity Broker on Unix-based systems.
dsconfig	View and edit the Identity Broker configuration.
dsframework	Manage administrative server groups or the global administrative user accounts that are used to configure servers within server groups.
dsjavaproperties	Configure the JVM arguments used to run the Identity Broker and its associated tools. Before launching the command, edit the properties file located in config/java.properties to specify the desired JVM arguments and the JAVA_HOME environment variable.
evaluate-policy	Request a policy decision from the Identity Broker.
ldapmodify	Perform LDAP modify, add, delete, and modify DN operations in the Identity Broker.
ldappasswordmodify	Perform LDAP password modify operations in the Identity Broker.
ldapsearch	Perform LDAP search operations in the Identity Broker.
ldif-diff	Compare the contents of two LDIF files, the output being an LDIF file needed to bring the source file in sync with the target.
ldifmodify	Apply a set of modify, add, and delete operations against data in an LDIF file.
list-backends	List the backends and base DNs configured in the Identity Broker.
manage-extension	Install or update extension bundles. An extension bundle is a package of extension(s) that utilize the Server SDK to extend the functionality of the Identity Broker. Any added extensions require a server re-start.
oauth2-request	Performs OAuth2 requests on the Identity Broker. This tool can be used to test OAuth2 functions of the Identity Broker, and to manage OAuth2 tokens on behalf of registered applications. See the **`--help-subcommands`** option for a list of supported sub-commands.
prepare-external-store	Prepares the external data stores for the Identity Broker. You do not need to run this tool if you have run the **`create-initial-broker-config`** tool. This tool creates the broker user account, sets the correct password, and configures the account with required privileges. It will also install the necessary schema required by the Identity Broker. Optionally, it can also install the

Table 8. Summary of Identity Broker Tools(continued)

Tool	Description
	base policy store DIT structure using an LDIF file. If necessary you are prompted for manager credentials in order that the tool can perform any required modifications to the external server.
remove-defunct-server	Removes a permanently unavailable Identity Broker after it has been removed from its topology by the **`uninstall`** tool.
restore	Restore a backup of the Identity Broker.
review-license	Review and/or indicate your acceptance of the product license.
sample-data-loader	Install or remove sample data for Identity Broker testing and demonstration.
server-state	View information about the current state of the Identity Broker processes.
start-broker	Start the Identity Broker.
status	Display basic server information.
stop-broker	Stop or restart the Identity Broker.
sum-file-sizes	Calculate the sum of the sizes for a set of files.
summarize-config	Generate a configuration summary of either a remote or local Identity Broker instance. By default, only basic components and properties will be included. To include advanced component, use the **`--advanced`** argument.

About the Identity Broker Roles and Scopes

Authentication is accomplished on the Identity Broker through Spring Security, which is enabled on the HTTPS Connection Handler. Only one Spring Security HTTP Servlet Extension may be used with both the HTTP and HTTPS connection handlers per server instance, so we recommend using it with the HTTPS connection handler.

The Identity Broker defines Admin Roles and scopes to perform operations using the Identity Broker REST APIs. The Identity Broker supports a single "Default" scope that all admin apps and clients use. It indicates that the user has successfully authenticated. The operations allowed by the authorities/roles that the user has determines the operations that the admin can carry out.

The roles must be populated by the Spring AuthenticationProvider at login time and are stored with the access token. The Identity Broker pulls the granted authorities from a configurable attribute on the user entry and group entries of which the user is a member. The default attribute is "id-broker-admin-privilege".

```
dn: uid=user.341,ou=Users,ou=Identity Broker,dc=example,dc=com
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
givenName: Anette
sn: Associates
cn: Anette Associates
initials: AKA
employeeNumber: 341
uid: user.341
mail: user.341@example.com
userPassword: password
telephoneNumber: +1 999 002 6412
homePhone: +1 003 008 0090
pager: +1 609 134 0692
mobile: +1 470 169 0659
street: 52513 Washington Street
l: Harrisonburg
st: MD
postalCode: 98130
postalAddress: Anette Associates$52513 Washington Street$Harrisonburg, MD
98130
description: This is the description for Anette Associates.
id-broker-admin-privilege: create_action
id-broker-admin-privilege: read_action
id-broker-admin-privilege: update_action
id-broker-admin-privilege: read_scope
id-broker-admin-privilege: read_application
id-broker-admin-privilege: read_tag
id-broker-admin-privilege: update_tag
```

The Spring Secuirty automatically adds the "ROLE_" prefix and uses this internally to represent a "SimpleGranted Authority". The values of the **`id-broker-admin-privilege`** attribute can be "create_action" or "admin" or "read_purpose". There are CRUD roles for every resource type, plus some special roles at the end for the history, policy decision point, and authorization log history endpoints.

By default all users will have these five authorities (implicitly):

- ROLE_CREATE_OWN_CONSENT
- ROLE_READ_OWN_CONSENT
- ROLE_DELETE_OWN_CONSENT
- ROLE_READ_OWN_CONSENTHISTORY
- ROLE_READ_OWN_ACCESSHISTORY

Authorities

- ROLE_CREATE_ACTION
- ROLE_READ_ACTION
- ROLE_UPDATE_ACTION
- ROLE_DELETE_ACTION

- ROLE_CREATE_APPLICATION
- ROLE_READ_APPLICATION
- ROLE_UPDATE_APPLICATION
- ROLE_DELETE_APPLICATION

- ROLE_CREATE_APPLICATIONGROUP
- ROLE_UPDATE_APPLICATIONGROUP
- ROLE_DELETE_APPLICATIONGROUP

- ROLE_CREATE_POLICY
- ROLE_READ_POLICY
- ROLE_UPDATE_POLICY
- ROLE_DELETE_POLICY

- ROLE_CREATE_POLICYSANDBOX
- ROLE_READ_POLICYSANDBOX
- ROLE_UPDATE_POLICYSANDBOX
- ROLE_DELETE_POLICYSANDBOX

- ROLE_CREATE_POLICYSET
- ROLE_READ_POLICYSET
- ROLE_UPDATE_POLICYSET
- ROLE_DELETE_POLICYSET

- ROLE_CREATE_POLICYTEMPLATE
- ROLE_READ_POLICYTEMPLATE
- ROLE_UPDATE_POLICYTEMPLATE
- ROLE_DELETE_POLICYTEMPLATE

- ROLE_CREATE_PURPOSE
- ROLE_READ_PURPOSE
- ROLE_UPDATE_PURPOSE
- ROLE_DELETE_PURPOSE

- ROLE_CREATE_REQUEST
- ROLE_READ_REQUEST
- ROLE_UPDATE_REQUEST
- ROLE_DELETE_REQUEST

- ROLE_CREATE_RESOURCE
- ROLE_READ_RESOURCE
- ROLE_UPDATE_RESOURCE
- ROLE_DELETE_RESOURCE

- ROLE_CREATE_RESOURCEALIAS
- ROLE_READ_RESOURCEALIAS
- ROLE_UPDATE_RESOURCEALIAS
- ROLE_DELETE_RESOURCEALIAS

- ROLE_CREATE_RESOURCEGROUP
- ROLE_READ_RESOURCEGROUP
- ROLE_UPDATE_RESOURCEGROUP
- ROLE_DELETE_RESOURCEGROUP

- ROLE_CREATE_SCOPE
- ROLE_READ_SCOPE
- ROLE_UPDATE_SCOPE
- ROLE_DELETE_SCOPE

- ROLE_CREATE_TAG
- ROLE_READ_TAG
- ROLE_UPDATE_TAG
- ROLE_DELETE_TAG

- ROLE_CREATE_TRACEFILTER
- ROLE_READ_TRACEFILTER
- ROLE_UPDATE_TRACEFILTER
- ROLE_DELETE_TRACEFILTER

- ROLE_CREATE_TRUSTLEVEL
- ROLE_READ_TRUSTLEVEL
- ROLE_UPDATE_TRUSTLEVEL
- ROLE_DELETE_TRUSTLEVEL

- ROLE_CREATE_OWN_CONSENT
- ROLE_CREATE_OTHER_CONSENT
- ROLE_READ_OWN_CONSENT
- ROLE_READ_OTHER_CONSENT
- ROLE_DELETE_OWN_CONSENT
- ROLE_DELETE_OTHER_CONSENT

- ROLE_READ_OWN_CONSENTHISTORY
- ROLE_READ_OTHER_CONSENTHISTORY

- ROLE_READ_OWN_ACCESSHISTORY
- ROLE_READ_OTHER_ACCESSHISTORY

- ROLE_INVOKE_PDP

- ROLE_READ_AUTHLOGS
- ROLE_BROKER_ADMIN (this is the super-user authority; a user with this can do anything)

For scopes, all admin tools (including the Broker Admin Console and Privacy Preferences app) will need to request the following scope when performing authentication.

- **urn:unboundid:broker:scope:default** -- The catch-all scope indicating the user has logged in successfully. It does not denote any particular authorities are held, but is necessary to allow an admin client to get a token.

Supported Storage Options

The UnboundID Identity Broker can be deployed in a variety of topologies depending on your existing infrastructure. The "New" column indicates that any new installation of the UnboundID Identity Data Store can be used with the data store. The "Existing" column indicates deployments that already use the UnboundID Identity Data Store for current token stores, for example, can be used as a token store repository with the Identity Broker. The "3rd Party Directory or Database" column indicates that a non-UnboundID directory or database can be used.

The Application Registry, Policy Store, and Consent Store must be on UnboundID Identity Data Store servers for version 1.0 of the Identity Broker due to the absence of extensions for 3rd-party directories or databases.

Table 9. Summary of Storage Options

Store	UnboundID Identity Data Store		3rd Party Directory or Database
	New	Existing	
Consumer Accounts	Yes	Yes	Yes, ID sync can be used to initialize sparse user entries.
Token Store	Yes	Yes	Yes, via an extension
Consent Store	Yes	Yes	No (Future: available via an extension)
Policy Store	Yes	Yes	No
Application Registry	Yes	Yes	No
Access History	Yes	Yes	Yes with indexing through Apache Lucene, which is

Table 9. Summary of Storage Options(continued)

Store	UnboundID Identity Data Store		3rd Party Directory or Database
	New	Existing	
			provided in the distribution zip file.
Configuration Data	Yes	Yes	Yes
Administrator Accounts	Yes	Yes	Yes, with SSO

Index

B

backup
  described 56
base64
  described 56
broker-admin
  described 56

C

collect-support-data
  described 57
consent-admin
  described 57
create-initial-broker-config
  described 57
create-rc-script
  described 57

D

dsconfig
  described 57
dsframework
  described 57
dsjavaproperties
  described 57

E

evaluate-policy
  described 57

I

Identity Broker
  described 1
  folders 55
  installing 21
  installing with existing truststore 29
  supported platforms 9
  tools 56

L

ldapmodify
  described 57
ldappasswordmodify
  described 57
ldapsearch
  described 57
ldif-diff
  described 57
ldifmodify
  described 57
list-backends
  described 57

M

manage-extension
  described 57

O

OAuth2
  testing the authorization flow 44
oauth2-request
  described 57

P

prepare-external-store
  described 57
Privacy Suite
  described 2

R

remove-defunct-server
  described 58
restore
  described 58
review-licence
  described 58

S

sample-data-loader
  described 58
  example of 39
server-state
  described 58
Spring Security
  authentication 58
  authorities 59
  roles 58
  scopes 62
start-broker
  described 58
  example of 51
  running in the foreground 51
status
  described 58
stop-broker
  described 58
  example of 52
  in-core restart 52
storage
  options 62
sum-file-sizes
  described 58
summarize-config
  described 58

T

trace filters
    creating 39

U

UnboundID
    about 5
uninstall tool 53