Space Details

Key:	CROWD
Name:	Crowd
Description:	Single Sign-On and Identity Management
Creator (Creation Date):	justen.stepka@atlassian.com (Sep 28, 2006)
Last Modifier (Mod. Date):	justen.stepka@atlassian.com (Feb 25, 2007)

Available Pages

- Crowd Documentation
  - Crowd Administration Guide
    - 1. Getting Started
      - 1.1 Concepts
        - 1.1.1 Supported Applications and Directories
      - 1.2 About the Crowd Administration Console
    - 2. Managing Directories
      - 2.1 Using the Directory Browser
      - 2.2 Adding a Directory
        - 2.2.1 Configuring an Internal Directory
        - 2.2.2 Configuring an LDAP Directory Connector
          - 2.2.2.1 Microsoft Active Directory
          - 2.2.2.2 SunONE
          - 2.2.2.3 OpenLDAP
          - 2.2.2.4 Apache Directory Server (ApacheDS)
          - 2.2.2.5 Generic LDAP Directories
          - UNPUBLISHED PENDING TESTING & SCREENSHOT- 2.2.2.2 Apple OSX Open Directory
        - 2.2.3 Configuring a Custom Directory Connector
      - 2.3 Specifying Directory Permissions
      - 2.4 Importing Principals and Groups into a Directory
        - 2.4.1 Importing Users from Atlassian Confluence
        - 2.4.2 Importing Users from Atlassian JIRA
        - 2.4.3 Importing Users from Jive Forums
    - 3. Managing Applications
      - 3.1 Using the Application Browser
      - 3.2 Adding an Application
        - 3.2.1 Integrating Crowd with Apache
        - 3.2.2 Integrating Crowd with Atlassian Bamboo
        - 3.2.3 Integrating Crowd with Atlassian Confluence
        - 3.2.4 Integrating Crowd with Atlassian JIRA
        - 3.2.5 Integrating Crowd with Cenqua FishEye

        - 3.2.6 Integrating Crowd with Jive Forums
            - 3.2.6.1 Jive SSO
        - 3.2.7 Integrating Crowd with a Custom Application
    - 3.3 Mapping a Directory to an Application
        - 3.3.1 Specifying the Directory Order for an Application
    - 3.4 Specifying which Groups can access an Application
    - 3.5 Specifying an Application's Address or Hostname
    - 3.6 Managing an Application's Session
    - 3.7 Deleting or Deactivating an Application
- 4. Managing Principals, Groups and Roles
    - 4.1 Using the Principal Browser
    - 4.2 Adding a Principal
    - 4.3 Deleting or Deactivating a Principal
    - 4.4 Managing a Principal's Session
    - 4.5 Editing a Principal's Details and Password
    - 4.6 Specifying a Principal's Attributes
    - 4.7 Editing a Principal's Group and Role Membership
    - 4.8 Using the Group Browser and Role Browser
    - 4.9 Adding a Group or Role
- 5. System Administration
    - 5.1 Viewing Crowd's System Information
    - 5.2 Configuring Server Settings
        - 5.2.1 Licensing
        - 5.2.2 Deployment Title
        - 5.2.3 Domain
        - 5.2.4 Token Seed
        - 5.2.5 Session Timeout
        - 5.2.6 Caching
            - 5.2 Configuring Caching for an Application
    - 5.3 Configuring SMTP Email
        - 5.3.1 Creating an Email Notification Template
    - 5.4 Backing Up and Restoring Data
    - 5.5 Logging
- Crowd Development Hub
    - Creating a Crowd Client for your Custom Application
        - Application Integration Overview
            - Sample Application ('demo')
        - Java Integration Libraries
        - SOAP API
            - Axis Client Stub Generation
            - Microsoft .NET Client
    - Creating a Custom Directory Connector
- Crowd Installation & Upgrade Guide
    - Crowd Release Notes
        - __newreleaseCrowd

        - Crowd 0.2 Beta Release Notes
        - Crowd 0.3 Beta Release Notes
        - Crowd 0.3.2 Beta Release Notes
        - Crowd 0.3.3 Beta Release Notes
        - Crowd 0.4 Beta Release Notes
        - Crowd 0.4.1 Beta Release Notes
        - Crowd 0.4.2 Beta Release Notes
        - Crowd 0.4.3 Beta Release Notes
        - Crowd 0.4.4 Beta Release Notes
        - Crowd 0.4.5 Beta Release Notes
        - Crowd 1.0.0 Release Notes
        - Crowd 1.0.1 Release Notes
        - Crowd 1.0.2 Release Notes
        - Crowd 1.0.3 Release Notes
        - Crowd 1.0.4 Release Notes
        - Crowd 1.0.5 Release Notes
        - Crowd 1.0.6 Release Notes
        - Crowd 1.0.7 Release Notes
    - Installing Crowd
        - 1. System Requirements
        - 2. Installing and Configuring Crowd
            - 2.1 Changing the Port that Crowd uses
        - 3. Connecting Crowd to a Database
            - 3.1 HSQL DB
            - 3.2 MS SQL Server
            - 3.3 MySQL
            - 3.4 Oracle
            - 3.5 PostgreSQL
        - 4. Running the Setup Wizard
    - Upgrading Crowd
        - Crowd 1.0 Upgrade Notes
- Crowd Knowledge Base
    - Deployment FAQ
        - Self Signed Certificate
    - External Resources
    - Integration FAQ
        - IBM Websphere Integration

Crowd Documentation

This page last changed on May 22, 2007 by rosie@atlassian.com.

Crowd 1.0 Documentation	Search
Installation Guide Upgrade Guide Release Notes Administration Guide Development Hub	Type **CTRL-F** to search the full Table of Contents on this page.
	Download
	You can download the Crowd documentation in PDF, HTML or XML formats.
About	**Resources**
Crowd is a web-based single sign-on (SSO) tool that simplifies application provisioning and identity management. Crowd is the perfect solution to: • Give your users the convenience of single sign-on • Manage any number of users, logins and passwords • Centralise user management for applications such as JIRA, Confluence and Bamboo • Connect to multiple LDAP servers, such as Microsoft Active Directory • Integrate or import legacy user repositories • Control access to select applications for every user and group • Easily connect Crowd's application framework to new web applications	If you have a question about using Crowd, please feel free to contact us at support. You may also want to check out the mailing lists forums: • Crowd Announcements • Crowd General Forum • Crowd Developers Forum Other handy links: • Crowd Knowledge Base • Javadoc • JIRA Issue Tracker for Crowd

Crowd Administration Guide — Table of Contents

1. Getting Started

- 1.1 Concepts
  - 1.1.1 Supported Applications and Directories

- 1.2 About the Crowd Administration Console

2. Managing Directories

- 2.1 Using the Directory Browser
- 2.2 Adding a Directory
  - 2.2.1 Configuring an Internal Directory
  - 2.2.2 Configuring an LDAP Directory Connector
    - 2.2.2.1 Microsoft Active Directory
    - 2.2.2.2 SunONE
    - 2.2.2.3 OpenLDAP
    - 2.2.2.4 Apache Directory Server (ApacheDS)
    - 2.2.2.5 Generic LDAP Directories
    - UNPUBLISHED PENDING TESTING & SCREENSHOT- 2.2.2.2 Apple OSX Open Directory
  - 2.2.3 Configuring a Custom Directory Connector
- 2.3 Specifying Directory Permissions
- 2.4 Importing Principals and Groups into a Directory
  - 2.4.1 Importing Users from Atlassian Confluence
  - 2.4.2 Importing Users from Atlassian JIRA
  - 2.4.3 Importing Users from Jive Forums

3. Managing Applications

- 3.1 Using the Application Browser
- 3.2 Adding an Application
  - 3.2.1 Integrating Crowd with Apache
  - 3.2.2 Integrating Crowd with Atlassian Bamboo
  - 3.2.3 Integrating Crowd with Atlassian Confluence
  - 3.2.4 Integrating Crowd with Atlassian JIRA
  - 3.2.5 Integrating Crowd with Cenqua FishEye
  - 3.2.6 Integrating Crowd with Jive Forums
    - 3.2.6.1 Jive SSO
  - 3.2.7 Integrating Crowd with a Custom Application
- 3.3 Mapping a Directory to an Application
  - 3.3.1 Specifying the Directory Order for an Application
- 3.4 Specifying which Groups can access an Application
- 3.5 Specifying an Application's Address or Hostname
- 3.6 Managing an Application's Session
- 3.7 Deleting or Deactivating an Application

4. Managing Principals, Groups and Roles

- 4.1 Using the Principal Browser
- 4.2 Adding a Principal
- 4.3 Deleting or Deactivating a Principal
- 4.4 Managing a Principal's Session
- 4.5 Editing a Principal's Details and Password
- 4.6 Specifying a Principal's Attributes
- 4.7 Editing a Principal's Group and Role Membership
- 4.8 Using the Group Browser and Role Browser

- 4.9 Adding a Group or Role

5. System Administration

- 5.1 Viewing Crowd's System Information
- 5.2 Configuring Server Settings
    - 5.2.1 Licensing
    - 5.2.2 Deployment Title
    - 5.2.3 Domain
    - 5.2.4 Token Seed
    - 5.2.5 Session Timeout
    - 5.2.6 Caching
        - 5.2 Configuring Caching for an Application
- 5.3 Configuring SMTP Email
    - 5.3.1 Creating an Email Notification Template
- 5.4 Backing Up and Restoring Data
- 5.5 Logging

Index

Crowd Administration Guide

This page last changed on Apr 23, 2007 by rosie@atlassian.com.

- 1. Getting Started
  - 1.1 Concepts
    - 1.1.1 Supported Applications and Directories
  - 1.2 About the Crowd Administration Console
- 2. Managing Directories
  - 2.1 Using the Directory Browser
  - 2.2 Adding a Directory
    - 2.2.1 Configuring an Internal Directory
    - 2.2.2 Configuring an LDAP Directory Connector
      - 2.2.2.1 Microsoft Active Directory
      - 2.2.2.2 SunONE
      - 2.2.2.3 OpenLDAP
      - 2.2.2.4 Apache Directory Server (ApacheDS)
      - 2.2.2.5 Generic LDAP Directories
      - UNPUBLISHED PENDING TESTING & SCREENSHOT- 2.2.2.2 Apple OSX Open Directory
    - 2.2.3 Configuring a Custom Directory Connector
  - 2.3 Specifying Directory Permissions
  - 2.4 Importing Principals and Groups into a Directory
    - 2.4.1 Importing Users from Atlassian Confluence
    - 2.4.2 Importing Users from Atlassian JIRA
    - 2.4.3 Importing Users from Jive Forums
- 3. Managing Applications
  - 3.1 Using the Application Browser
  - 3.2 Adding an Application
    - 3.2.1 Integrating Crowd with Apache
    - 3.2.2 Integrating Crowd with Atlassian Bamboo
    - 3.2.3 Integrating Crowd with Atlassian Confluence
    - 3.2.4 Integrating Crowd with Atlassian JIRA
    - 3.2.5 Integrating Crowd with Cenqua FishEye
    - 3.2.6 Integrating Crowd with Jive Forums
      - 3.2.6.1 Jive SSO
    - 3.2.7 Integrating Crowd with a Custom Application
  - 3.3 Mapping a Directory to an Application
    - 3.3.1 Specifying the Directory Order for an Application
  - 3.4 Specifying which Groups can access an Application
  - 3.5 Specifying an Application's Address or Hostname
  - 3.6 Managing an Application's Session
  - 3.7 Deleting or Deactivating an Application
- 4. Managing Principals, Groups and Roles
  - 4.1 Using the Principal Browser
  - 4.2 Adding a Principal
  - 4.3 Deleting or Deactivating a Principal
  - 4.4 Managing a Principal's Session
  - 4.5 Editing a Principal's Details and Password
  - 4.6 Specifying a Principal's Attributes
  - 4.7 Editing a Principal's Group and Role Membership
  - 4.8 Using the Group Browser and Role Browser
  - 4.9 Adding a Group or Role
- 5. System Administration

    - 5.1 Viewing Crowd's System Information
    - 5.2 Configuring Server Settings
        - 5.2.1 Licensing
        - 5.2.2 Deployment Title
        - 5.2.3 Domain
        - 5.2.4 Token Seed
        - 5.2.5 Session Timeout
        - 5.2.6 Caching
            - 5.2 Configuring Caching for an Application
    - 5.3 Configuring SMTP Email
        - 5.3.1 Creating an Email Notification Template
    - 5.4 Backing Up and Restoring Data
    - 5.5 Logging

- Index

1. Getting Started

This page last changed on Mar 12, 2007 by rosie@atlassian.com.

- 1.1 Concepts
  - 1.1.1 Supported Applications and Directories
- 1.2 About the Crowd Administration Console

1.1 Concepts

This page last changed on Apr 23, 2007 by rosie@atlassian.com.

Crowd is an application security framework that handles authentication and authorisation for your web-based applications. With Crowd you can quickly integrate multiple web applications into a single security architecture that supports single sign-on (SSO) and centralised identity management.

Crowd has two components:

- The Crowd Administration Console is a clean and powerful web-interface for managing directories, users (known in Crowd as 'principals') and their security rights ('permissions').
- The Crowd integration API provides a platform-neutral way to integrate web applications into a single security architecture. With the integration API, applications can quickly access user information and perform security checks.

Designed for ease of use, Crowd can be deployed with your existing infrastructure. Crowd supports:

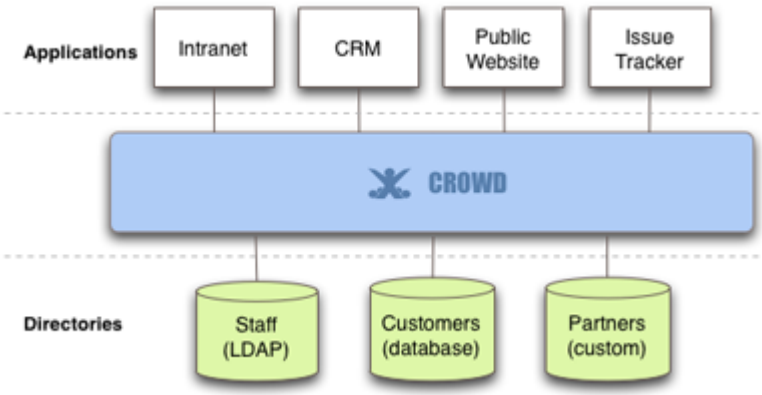
- Java, .NET and PHP applications.
- Popular directory servers such as Microsoft Active Directory, Sun ONE and OpenLDAP. Additionally, custom directory connectors may be developed using the Crowd integration API.

See the list of supported applications and directories.

Architectural Overview

Crowd is a middleware application that integrates web applications into a single security architecture that supports single sign-on and centralised identity management. Crowd works by dispatching authentication and authorisation calls from configured applications to configured directories.

A typical deployment may be similar to the following:



When an application needs to validate a security or authentication request (e.g. when a user attempts to log in to the application), the application will make a simple API call to the Crowd framework, which will then forward the call to the appropriate directory.

About Applications

Crowd integrates and provisions applications. Once defined, an application is mapped to a directory(s), whose users are then granted access to the application. Note that an application can only communicate with Crowd when the application uses a known host address.

About Directories

Crowd supports an unlimited number of user directories. A directory can either be internal to Crowd, or connected to Crowd via an LDAP connector (e.g. for Active Directory) or via a custom directory connector (e.g. for a legacy database).
Once a directory has been defined in Crowd, it can be mapped to applications. Crowd will then delegate authentication and authorisation requests to the directory, for all applications that are mapped to that directory. Modification of directory entities (users, groups and roles) can be done via the Crowd Administration Console or via the application, depending on the application's capabilities.
You can even map multiple directories to an application, providing the application with a single view of multiple directories, in a specified order.

Related Topics

- 1.1 Concepts
  - 1.1.1 Supported Applications and Directories
- 1.2 About the Crowd Administration Console

Crowd Documentation

1.1.1 Supported Applications and Directories

This page last changed on Mar 29, 2007 by rosie@atlassian.com.

Application Connectors

- Atlassian JIRA
- Atlassian Confluence
- Atlassian Bamboo
- Cenqua Fisheye
- Jive Forums
- Jive Wildfire

You can also add your own custom applications.

Directory Connectors

- Microsoft Active Directory
- OpenLDAP
- Sun Java System (SunONE) Directory Server
- Apache Directory Server (ApacheDS)
- Internal Crowd Directory

You can also add your own custom directories.

Related Topics

- 1.1 Concepts
  - 1.1.1 Supported Applications and Directories
- 1.2 About the Crowd Administration Console

Crowd Documentation

1.2 About the Crowd Administration Console

This page last changed on Apr 02, 2007 by rosie@atlassian.com.

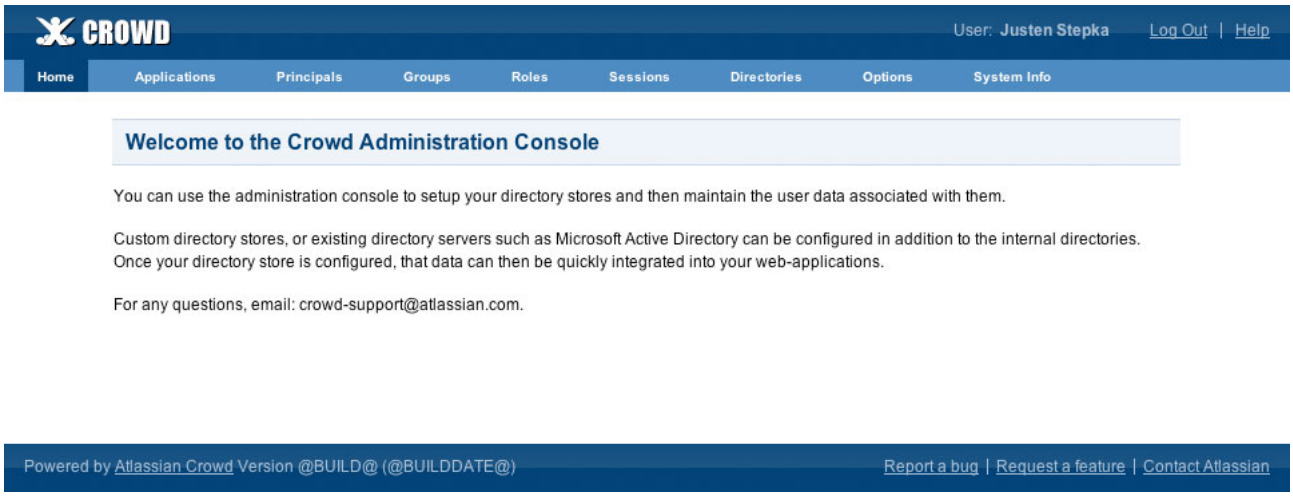
After completing the installation process, you will be able to access the Crowd Administration Console. Here you can:

- Configure applications to access the Crowd framework.
- View active sessions and manually expire sessions.
- Map directories to allow users ('principals') to access integrated applications.
- Create and manage principals along with adjusting group and role membership.
- Adjust server deployment properties configured during the setup process.

To access the Crowd Administration Console,

1. Go to the URL http://localhost:8095/console or http://localhost:8095/crowd/console.

The welcome screen will look similar to the following:



The Crowd Administration Console is a web-application that is provisioned by Crowd — you can see it in the list of applications shown in the Application Browser.

Related Topics

- 1.1 Concepts
  - 1.1.1 Supported Applications and Directories
- 1.2 About the Crowd Administration Console

Crowd Documentation

2. Managing Directories

This page last changed on Mar 28, 2007 by rosie@atlassian.com.

Crowd supports an unlimited number of user directories. A directory can either be internal to Crowd, or connected to Crowd via an LDAP connector (e.g. for Active Directory) or via a custom directory connector (e.g. for a legacy database).
Once a directory has been defined in Crowd, it can be mapped to applications. Crowd will then delegate authentication and authorisation requests to the directory, for all applications that are mapped to that directory. Modification of directory entities (users, groups and roles) can be done via the Crowd Administration Console or via the application, depending on the application's capabilities.
You can even map multiple directories to an application, providing the application with a single view of multiple directories, in a specified order.

- 2.1 Using the Directory Browser
- 2.2 Adding a Directory
    - 2.2.1 Configuring an Internal Directory
    - 2.2.2 Configuring an LDAP Directory Connector
        - 2.2.2.1 Microsoft Active Directory
        - 2.2.2.2 SunONE
        - 2.2.2.3 OpenLDAP
        - 2.2.2.4 Apache Directory Server (ApacheDS)
        - 2.2.2.5 Generic LDAP Directories
        - UNPUBLISHED PENDING TESTING & SCREENSHOT- 2.2.2.2 Apple OSX Open Directory
    - 2.2.3 Configuring a Custom Directory Connector
- 2.3 Specifying Directory Permissions
- 2.4 Importing Principals and Groups into a Directory
    - 2.4.1 Importing Users from Atlassian Confluence
    - 2.4.2 Importing Users from Atlassian JIRA
    - 2.4.3 Importing Users from Jive Forums

2.1 Using the Directory Browser

This page last changed on Mar 27, 2007 by rosie@atlassian.com.

About Directories

Crowd supports an unlimited number of user directories. A directory can either be internal to Crowd, or connected to Crowd via an LDAP connector (e.g. for Active Directory) or via a custom directory connector (e.g. for a legacy database).
Once a directory has been defined in Crowd, it can be mapped to applications. Crowd will then delegate authentication and authorisation requests to the directory, for all applications that are mapped to that directory. Modification of directory entities (users, groups and roles) can be done via the Crowd Administration Console or via the application, depending on the application's capabilities.
You can even map multiple directories to an application, providing the application with a single view of multiple directories, in a specified order.

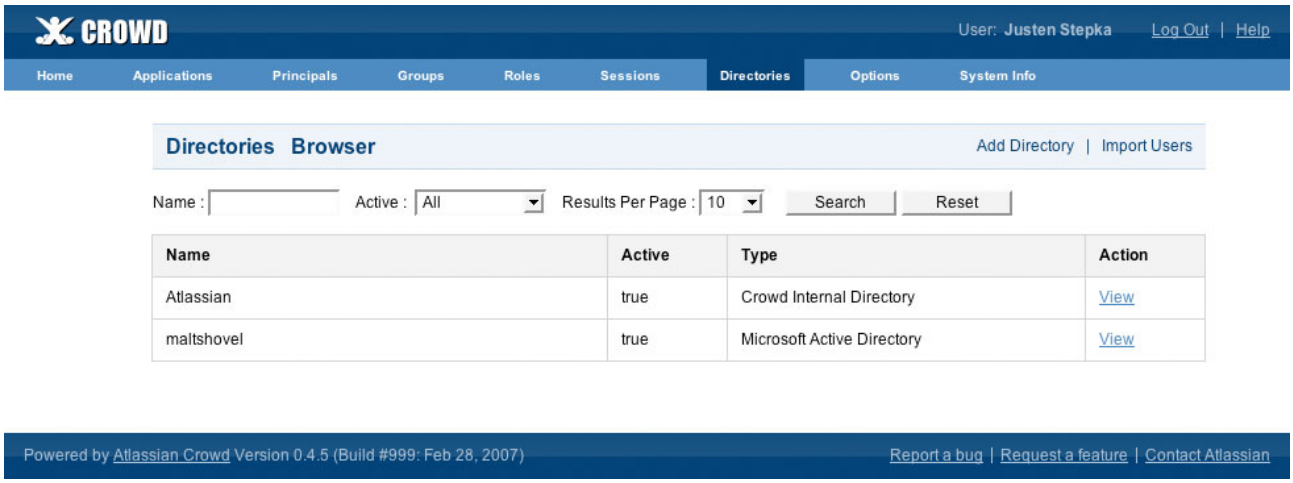
About the Directory Browser

The Directory Browser allows you to view and search for configured directories.

To use the Directory Browser,

1. Login to the Crowd Administration Console.
2. Click the 'Directories' link in the top navigation bar.
3. This will display the Directory Browser, showing all the directories that exist in your Crowd system. You can refine your search by specifying a 'Name' (note that this is case-sensitive), or 'Active'/'Inactive' directories.
   An 'Inactive' directory is unable to be used by any applications, regardless of whether or not they are mapped to it.
4. To view/edit a directory's details, click the 'View' link.

Screenshot: 'Directory Browser'



Related Topics

- 2.1 Using the Directory Browser
- 2.2 Adding a Directory
    - 2.2.1 Configuring an Internal Directory
    - 2.2.2 Configuring an LDAP Directory Connector
        - 2.2.2.1 Microsoft Active Directory
        - 2.2.2.2 SunONE
        - 2.2.2.3 OpenLDAP
        - 2.2.2.4 Apache Directory Server (ApacheDS)
        - 2.2.2.5 Generic LDAP Directories
        - UNPUBLISHED PENDING TESTING & SCREENSHOT- 2.2.2.2 Apple OSX Open Directory
    - 2.2.3 Configuring a Custom Directory Connector
- 2.3 Specifying Directory Permissions
- 2.4 Importing Principals and Groups into a Directory
    - 2.4.1 Importing Users from Atlassian Confluence
    - 2.4.2 Importing Users from Atlassian JIRA
    - 2.4.3 Importing Users from Jive Forums

Crowd Documentation

2.2 Adding a Directory

This page last changed on Mar 20, 2007 by rosie@atlassian.com.

Directories contain authentication and authorisation information about users, groups and roles. Crowd supports an unlimited number of directories, which allows administrators to create silos of users (e.g. 'customers' and 'employees').
Crowd supports three types of directories:

- Crowd Internal Directory — Internal directories use the Crowd database to store user, group and role information. Internal directories are stored in Crowd's database server.
- LDAP Directory Connector — Crowd provides built-in connectors for the most popular LDAP directory servers (Micrsoft Active Directory, SunONE, OpenLDAP, Apache Directory). These LDAP connectors enable you to quickly integrate existing desktop logins with web-applications.
- Custom Directory Connector — Custom directory connectors allow developers to connect Crowd to custom user-stores, such as existing databases or legacy system.

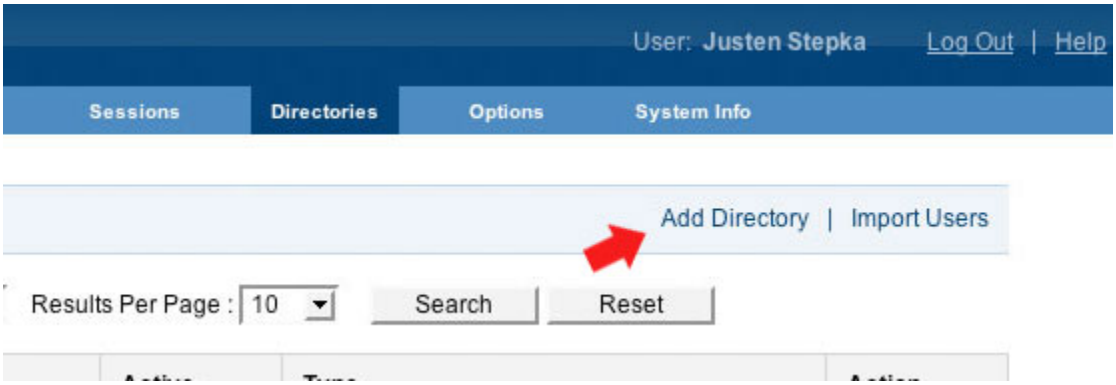
You can add as many (or as few) directories of each type as you need.

To add a directory,

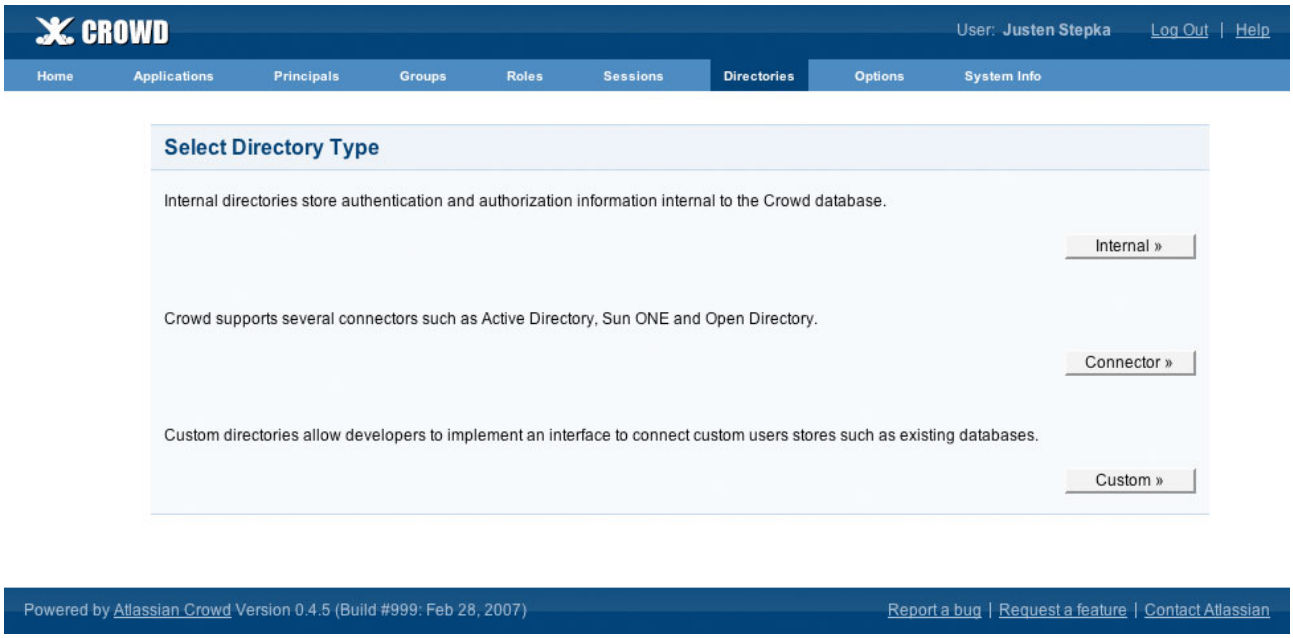
1. Login to the Crowd Administration Console.
2. Click the 'Directories' link in the top navigation bar.
3. This will display the Directory Browser. Click the 'Add Directory' link.
4. This will display the 'Select Directory Type' screen (see below). Click the button corresponding to the type of directory you want to add:
    - 'Internal' — see 2.2.1 Configuring an Internal Directory
    - 'Connector' — see 2.2.2 Configuring an LDAP Directory Connector (e.g. Microsoft Active Directory)
    - 'Custom' — see 2.2.3 Configuring a Custom Directory Connector

Once a directory has been configured, you will need to specify permissions for its users. You can then map the directory to appropriate applications.

Screenshot 1: 'Add Directory'



Screenshot 2: 'Select Directory Type'



Related Topics

- 2.1 Using the Directory Browser
- 2.2 Adding a Directory
  - 2.2.1 Configuring an Internal Directory
  - 2.2.2 Configuring an LDAP Directory Connector
    - 2.2.2.1 Microsoft Active Directory
    - 2.2.2.2 SunONE
    - 2.2.2.3 OpenLDAP
    - 2.2.2.4 Apache Directory Server (ApacheDS)
    - 2.2.2.5 Generic LDAP Directories
    - UNPUBLISHED PENDING TESTING & SCREENSHOT- 2.2.2.2 Apple OSX Open Directory
  - 2.2.3 Configuring a Custom Directory Connector
- 2.3 Specifying Directory Permissions
- 2.4 Importing Principals and Groups into a Directory
  - 2.4.1 Importing Users from Atlassian Confluence
  - 2.4.2 Importing Users from Atlassian JIRA
  - 2.4.3 Importing Users from Jive Forums

Crowd Documentation

2.2.1 Configuring an Internal Directory

This page last changed on Apr 02, 2007 by justin.

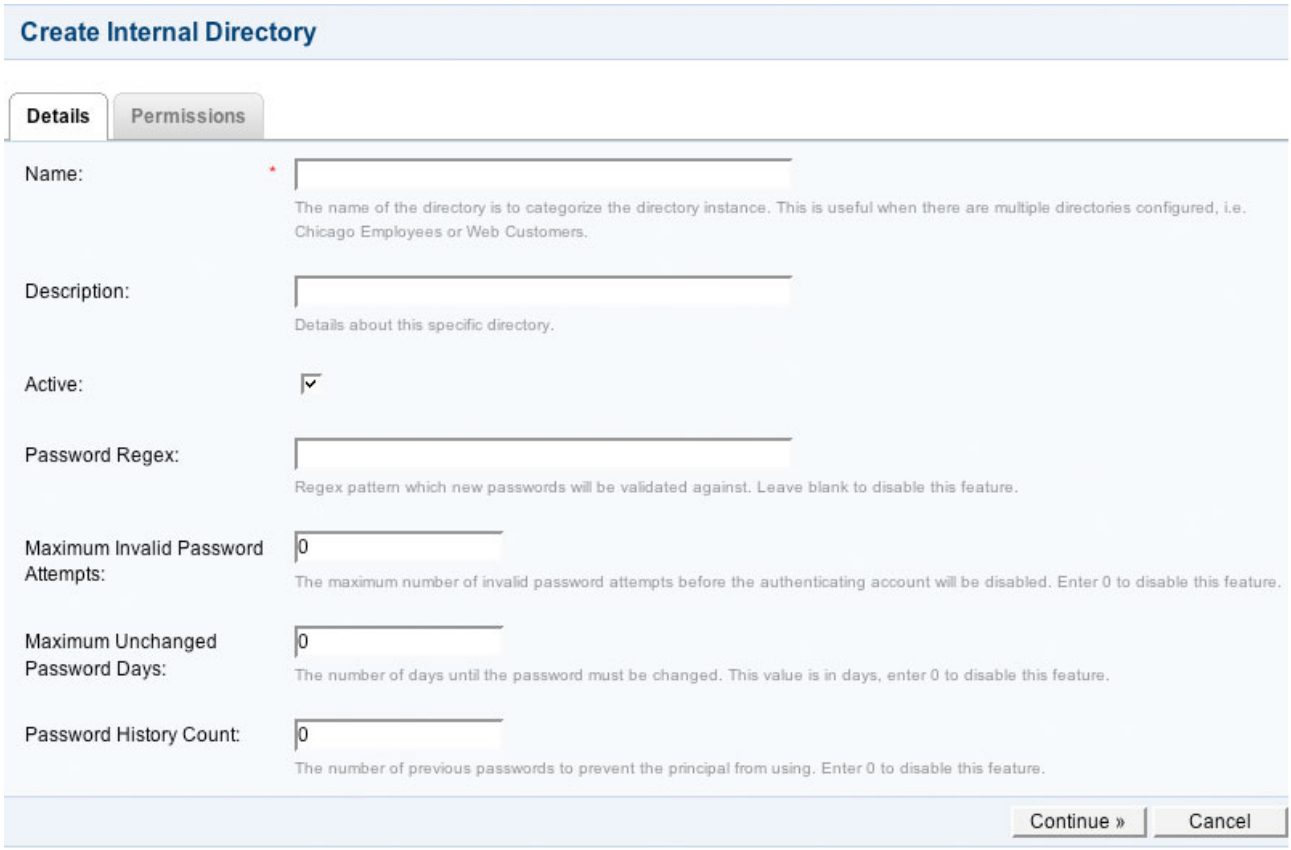
Internal directories use the Crowd database to store user, group and role information. Internal directories are stored in Crowd's database server.

To configure an Internal Directory,

1. Login to the Crowd Administration Console.
2. Click the 'Directories' link in the top navigation bar.
3. This will display the Directory Browser. Click the 'Add Directory' link.
4. Click the 'Internal' button.
5. Complete the fields as described in the table below.
6. Click the 'Continue' button to configure the directory's permissions.

Once you have configured the directory's permissions, you will have finished configuring your new directory. You can then map the directory to appropriate applications.

Screenshot: 'Create Internal Directory'



Internal Directory Attributes	Description
Name	The name used to identify the directory within Crowd. This is useful when there are multiple directories configured, e.g. Chicago Employees or

	Web Customers.
Description	Details about this specific directory.
Active	Only deselect this if you wish to prevent all users ('principals') within the directory from accessing all mapped applications.
Password Regex	Regex pattern which new passwords will be validated against. Leave blank to disable this feature.
Maximum Invalid Password Attempts	The maximum number of invalid password attempts before the authenticating account will be disabled. Enter 0 to disable this feature.
Maimum Unchanged Password Days	The number of days until the password must be changed. This value is in days, enter 0 to disable this feature.
Password History Count	The number of previous passwords to prevent the principal from using. Enter 0 to disable this feature.

Next Step:

See 2.3 Specifying Directory Permissions

Related Topics

- 2.1 Using the Directory Browser
- 2.2 Adding a Directory
  - 2.2.1 Configuring an Internal Directory
  - 2.2.2 Configuring an LDAP Directory Connector
    - 2.2.2.1 Microsoft Active Directory
    - 2.2.2.2 SunONE
    - 2.2.2.3 OpenLDAP
    - 2.2.2.4 Apache Directory Server (ApacheDS)
    - 2.2.2.5 Generic LDAP Directories
    - UNPUBLISHED PENDING TESTING & SCREENSHOT- 2.2.2.2 Apple OSX Open Directory
  - 2.2.3 Configuring a Custom Directory Connector
- 2.3 Specifying Directory Permissions
- 2.4 Importing Principals and Groups into a Directory
  - 2.4.1 Importing Users from Atlassian Confluence
  - 2.4.2 Importing Users from Atlassian JIRA
  - 2.4.3 Importing Users from Jive Forums

Crowd Documentation

2.2.2 Configuring an LDAP Directory Connector

This page last changed on Apr 23, 2007 by rosie@atlassian.com.

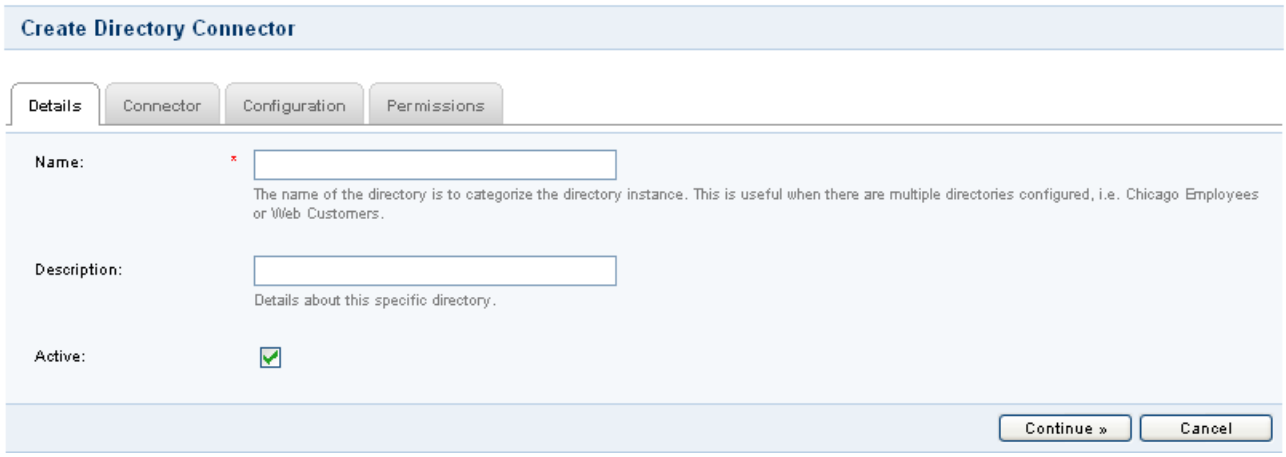
Crowd provides built-in connectors for the most popular LDAP directory servers (Micrsoft Active Directory, SunONE, OpenLDAP, Apache Directory). These LDAP connectors enable you to quickly integrate existing desktop logins with web-applications.
To configure an LDAP Directory Connector,

1. Login to the Crowd Administration Console.
2. Click the 'Directories' link in the top navigation bar.
3. This will display the Directory Browser. Click the 'Add Directory' link.
4. This will display the 'Select Directory Type' screen. Click the 'Connector' button.
5. This will display the 'Details' tab (see Screenshot 1 below). Enter the 'Name' and 'Description' fields (see table below), then click the 'Continue' button.
6. This will display the 'Connector' tab (see Screenshot 2 below). Select the relevant connector type, and fill in the basic connection information for your directory server. For details, please see:
   - 2.2.2.1 Microsoft Active Directory
   - 2.2.2.2 SunONE
   - 2.2.2.3 OpenLDAP
   - 2.2.2.4 Apache Directory Server (ApacheDS)
   - 2.2.2.5 Generic LDAP Directories
   - UNPUBLISHED PENDING TESTING & SCREENSHOT- 2.2.2.2 Apple OSX Open Directory

   Then click the 'Continue' button.
7. This will display the 'Configuration' tab (see Screenshot 3 below). Fill in the configuration details as described in the tables below the screenshot. Also please see LDAP Object Structures (below).
8. Click the 'Continue' button to configure the directory's permissions.

Once you have configured the directory's permissions, you will have finished configuring your new directory. You can then map the directory to appropriate applications.

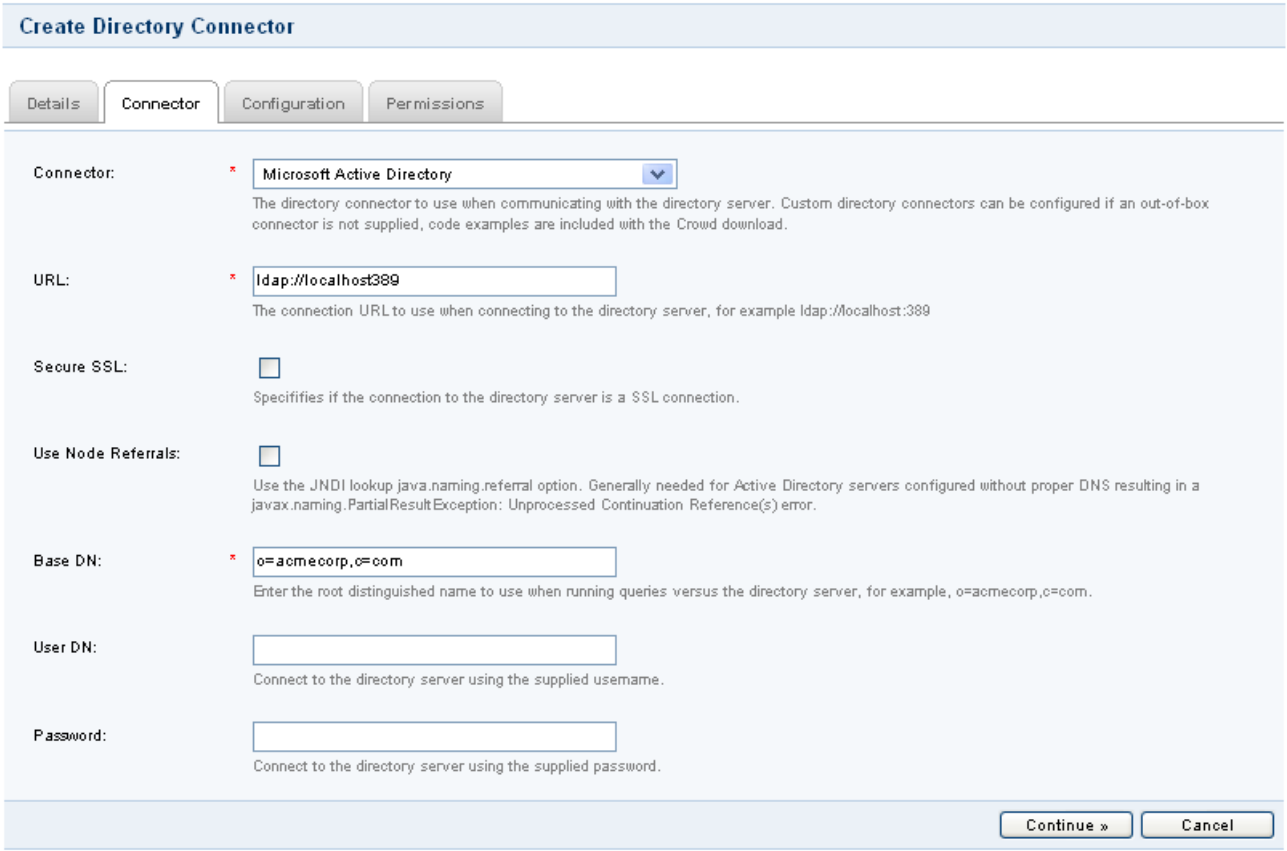
Screenshot 1: 'Details'



Attribute	Description

Name	The name used to identify the directory within Crowd. This is useful when there are multiple directories configured, e.g. 'Chicago Employees' or 'Web Customers'.
Description	Details about this specific directory.
Active	Only deselect this if you wish to prevent all users ('principals') within the directory from accessing all mapped applications.

Screenshot 2: 'Connector'



Attribute	Description
Connector	The directory connector to use when communicating with the directory server.
URL	The connection URL to use when connecting to the directory server, e.g.: `ldap://localhost:389`, or `port 639` for SSL.
Secure SSL	Specifies if the connection to the directory server is a SSL connection.
Use Node Referrals	Use the JNDI lookup java.naming.referral option. Generally needed for Active Directory servers configured without proper DNS, to prevent a 'javax.naming.PartialResultException:

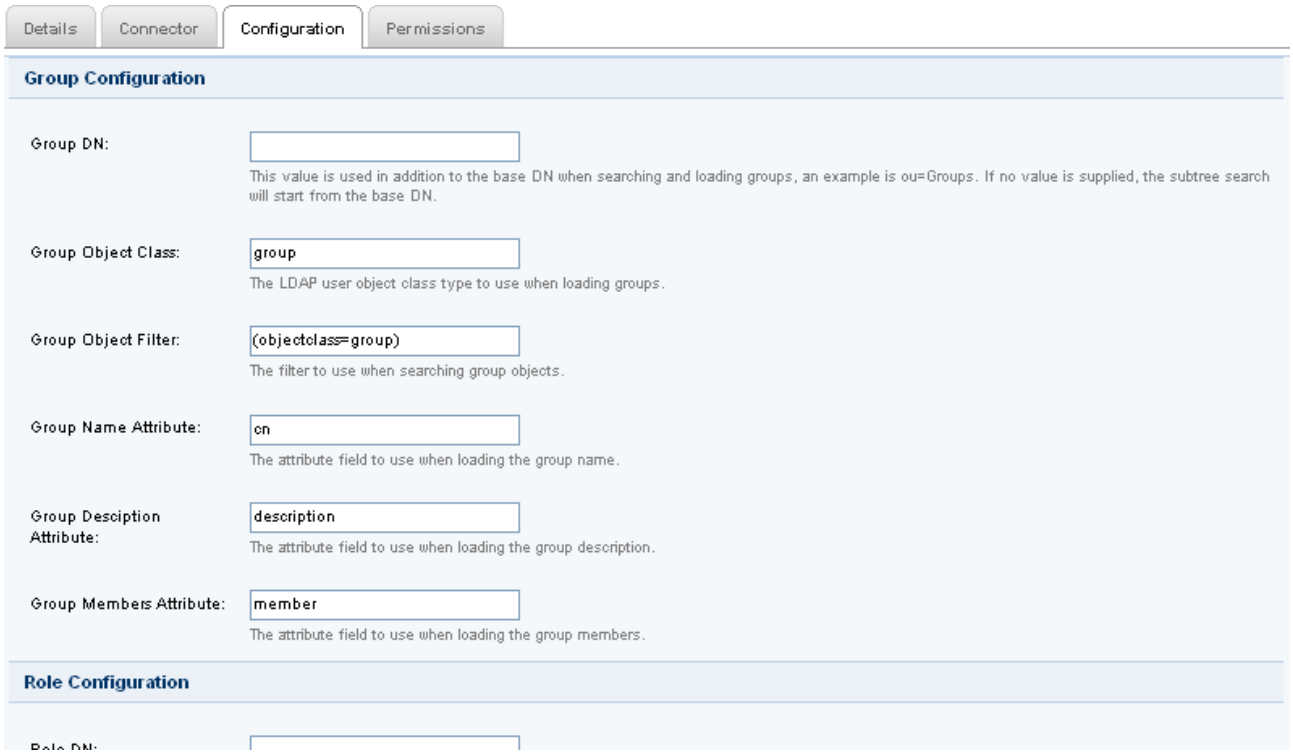
	Unprocessed Continuation Reference(s)' error.
Use Paged Results	Use the LDAP control extension for simple paged results option. Retrieves chunks of data rather than all of the results at once. This feature may be necessary when using Microsoft Active Directory if more than 999 results are returned for any given search.
Base DN	Enter the root distinguished name to use when running queries versus the directory server, e.g.: `o=acmecorp,c=com`.
User DN	The username that Crowd will use when connecting to the directory server.
Password	The password that Crowd will use when connecting to the directory server.

For details about the settings for your specific directory server, please see:

- 2.2.2.1 Microsoft Active Directory
- 2.2.2.2 SunONE
- 2.2.2.3 OpenLDAP
- 2.2.2.4 Apache Directory Server (ApacheDS)
- 2.2.2.5 Generic LDAP Directories
- UNPUBLISHED PENDING TESTING & SCREENSHOT- 2.2.2.2 Apple OSX Open Directory

To help you identify your LDAP structure, JXplorer is a free tool that allows you to browse your LDAP tree.

Screenshot 3: 'Configuration'



Once you have selected a Connector, various LDAP object and attribute settings of the specific LDAP server may be modified. Generic default settings have been provided based on the Connector selected. When configuring your LDAP connector, if you are using non-standard object types, you will need to adjust the default filter and object type configurations. Default values are configured for the predefined LDAP servers. If your connector is added successfully, but you are unable to see any data when browsing your LDAP directory, it is likely that your object and filters are configured incorrectly.

Group Configuration

Attribute	Description
Group DN	This value is used in addition to the base DN when searching and loading groups, an example is ou=Groups. If no value is supplied, the subtree search will start from the base DN.
Group Object Class	This value is used in addition to the base DN when searching and loading groups, an example is ou=Groups. If no value is supplied, the subtree search.
Group Object Filter	The filter to use when searching group objects.
Group Name Attribute	The attribute field to use when loading the group's name.
Group Desciption Attribute	The attribute field to use when loading the group's description.
Group Members Attribute	The attribute field to use when loading the group's members.

Role Configuration

Attribute	Description
Role DN	This value is used in addition to the base DN when searching and loading roles, an example is ou=Roles. If no value is supplied, the subtree search will start from the base DN.
Role Object Class	This value is used in addition to the base DN when searching and loading roles, an example is ou=Roles. If no value is supplied, the subtree search.
Role Object Filter	The filter to use when searching role objects.
Role Name Attribute	The attribute field to use when loading the role's name.
Role Desciption Attribute	The attribute field to use when loading the role's description.
Role Members Attribute	The attribute field to use when loading the role's members.

Principal Configuration

(In Crowd, users are known as principals.)

Attribute	Description
User DN	This value is used in addition to the base DN when searching and loading users, an example is ou=Users. If no value is supplied, the subtree search will start from the base DN.
User Object Class	The LDAP user object class type to use when loading principals.
User Object Filter	The filter to use when searching user objects.
User Name	The attribute field to use when loading the principal's username.
User First Name	The attribute field to use when loading the principal's first name.
User Last Name	The attribute field to use when loading the principal's last name.
User Email	The attribute field to use when loading the principal's email.
User Group	The attribute field to use when loading the principal's groups.
User Password	The attribute field to use when manipulating a principal's password.

Only the User First Name, User Last Name and User Email attributes can be updated via the Crowd LDAP connectors. With a license purchase, full source is available and the LDAP connectors can be modified to support any number of attributes.

LDAP Object Structures

The Crowd LDAP connectors assume that all container objects (groups and roles) have the full DN to the associated member. Currently, the membership attributes on a Principal object are not used by Crowd; however, in the future these associations may be used to assist with performance when looking up memberships.
Supported Object Types:

- groupOfUniqueNames
- inetorgperson

Non-supported Object types:

The following object types are not supported because of the required `guiNumber` attribute.

- posixGroup
- posixUser

Zimbra Mail Server
Principal objects have been tested and are known to work with the `zimbraAccount` LDAP object types.

Microsoft Active Directory:
The Active Directory LDAP connector assumes that all LDAP object types are of the default structure. Any changes to the default object structure of the `User` and `Group` objects will require a custom connector to be coded.

Next Step:

See 2.3 Specifying Directory Permissions

Related Topics

- 2.1 Using the Directory Browser
- 2.2 Adding a Directory
  - 2.2.1 Configuring an Internal Directory
  - 2.2.2 Configuring an LDAP Directory Connector
    - 2.2.2.1 Microsoft Active Directory
    - 2.2.2.2 SunONE
    - 2.2.2.3 OpenLDAP
    - 2.2.2.4 Apache Directory Server (ApacheDS)
    - 2.2.2.5 Generic LDAP Directories
    - UNPUBLISHED PENDING TESTING & SCREENSHOT- 2.2.2.2 Apple OSX Open Directory
  - 2.2.3 Configuring a Custom Directory Connector
- 2.3 Specifying Directory Permissions
- 2.4 Importing Principals and Groups into a Directory
  - 2.4.1 Importing Users from Atlassian Confluence

- 2.4.2 Importing Users from Atlassian JIRA
- 2.4.3 Importing Users from Jive Forums

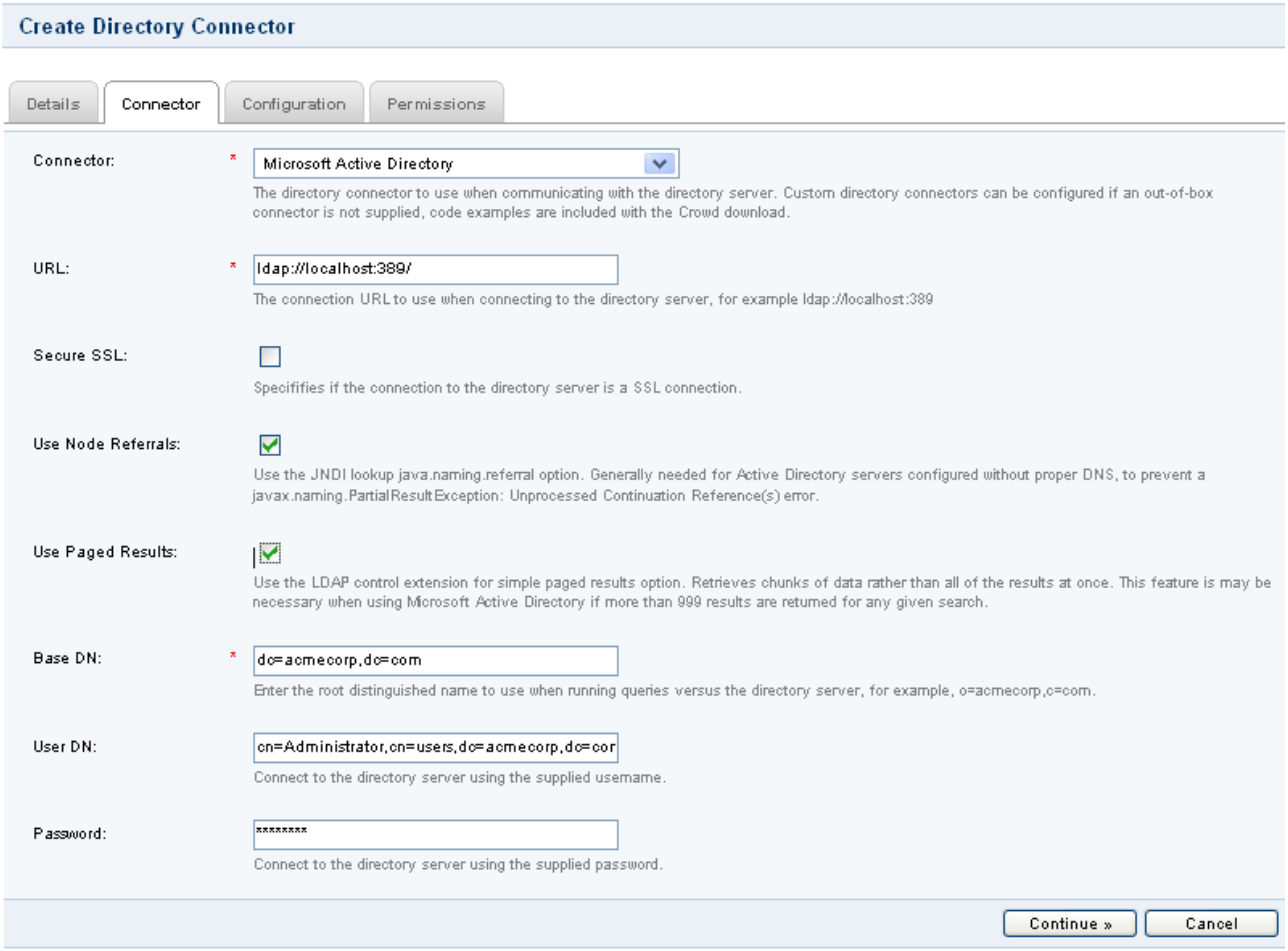
Crowd Documentation

2.2.2.1 Microsoft Active Directory

This page last changed on Mar 19, 2007 by rosie@atlassian.com.

This page provides configuration notes for Microsoft Active Directory, in relation to 2.2.2 Configuring an LDAP Directory Connector.

Screenshot: 'Connector — Microsoft Active Directory'



Attribute	Description
Connector	The directory connector to use when communicating with the directory server.
URL	The connection URL to use when connecting to the directory server, e.g.: `ldap://localhost:389`, or `port 639` for SSL.
Secure SSL	Specifies if the connection to the directory server is a SSL connection.
Use Node Referrals	Use the JNDI lookup java.naming.referral option. Generally needed for Active Directory servers configured without proper DNS, to prevent a 'javax.naming.PartialResultException:

	Unprocessed Continuation Reference(s)' error.
Use Paged Results	Use the LDAP control extension for simple paged results option. Retrieves chunks of data rather than all of the results at once. This feature may be necessary when using Microsoft Active Directory if more than 999 results are returned for any given search.
Base DN	Enter the root distinguished name to use when running queries versus the directory server, e.g.: `o=acmecorp,c=com`.
User DN	The username that Crowd will use when connecting to the directory server.
Password	The password that Crowd will use when connecting to the directory server.

Configuration notes for Microsoft Active Directory

Active Directory Attribute Example	Value
Base DN	`cn=users,dc=ad,dc=acmecorp,dc=com`
User DN	`administrator@ad.acmecorp.com`

For Microsoft Active Directory, specify the Base DN in the following format: `dc=domain1,dc=local`. You will need to replace the `domain1` and `local` for your specific configuration. Microsoft Server provides a tool called `ldp.exe` which is useful for finding out and configuring the the LDAP structure of your server.

The URL for Microsoft Active Directory should be in the following format: `ldap://domainname`.

Next Step

Go back to 2.2.2 Configuring an LDAP Directory Connector

Related Topics

- 2.1 Using the Directory Browser
- 2.2 Adding a Directory
  - 2.2.1 Configuring an Internal Directory
  - 2.2.2 Configuring an LDAP Directory Connector
    - 2.2.2.1 Microsoft Active Directory
    - 2.2.2.2 SunONE
    - 2.2.2.3 OpenLDAP
    - 2.2.2.4 Apache Directory Server (ApacheDS)
    - 2.2.2.5 Generic LDAP Directories
    - UNPUBLISHED PENDING TESTING & SCREENSHOT- 2.2.2.2 Apple OSX Open Directory

    - 2.2.3 Configuring a Custom Directory Connector
- 2.3 Specifying Directory Permissions
- 2.4 Importing Principals and Groups into a Directory
    - 2.4.1 Importing Users from Atlassian Confluence
    - 2.4.2 Importing Users from Atlassian JIRA
    - 2.4.3 Importing Users from Jive Forums

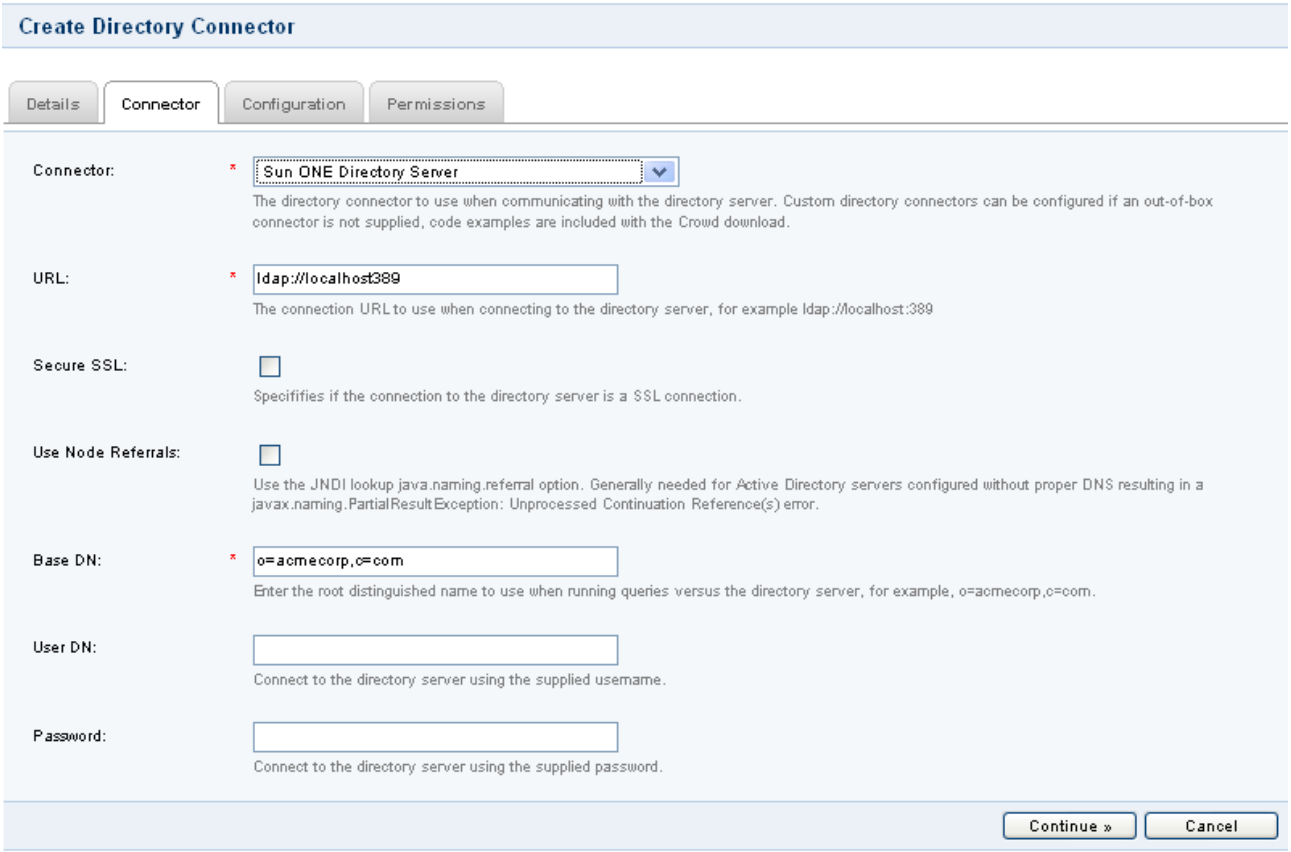
Crowd Documentation

2.2.2.2 SunONE

This page last changed on Mar 15, 2007 by rosie@atlassian.com.

This page provides configuration notes for SunONE Directory Server, in relation to 2.2.2 Configuring an LDAP Directory Connector.

Screenshot: 'Connector — SunONE Directory Server'



Attribute	Description
Connector	The directory connector to use when communicating with the directory server.
URL	The connection URL to use when connecting to the directory server, e.g.: `ldap://localhost:389`, or `port 639` for SSL.
Secure SSL	Specifies if the connection to the directory server is a SSL connection.
Use Node Referrals	Use the JNDI lookup java.naming.referral option. Generally needed for Active Directory servers configured without proper DNS, to prevent a 'javax.naming.PartialResultException: Unprocessed Continuation Reference(s)' error.
Use Paged Results	Use the LDAP control extension for simple paged results option. Retrieves chunks of data rather

	than all of the results at once. This feature may be necessary when using Microsoft Active Directory if more than 999 results are returned for any given search.
Base DN	Enter the root distinguished name to use when running queries versus the directory server, e.g.: `o=acmecorp,c=com`.
User DN	The username that Crowd will use when connecting to the directory server.
Password	The password that Crowd will use when connecting to the directory server.

Configuration details for SunONE

SunONE Example	Value
Base DN	dc=acmecorp,dc=com
User DN	cn=Directory Manager

Next Step

Go back to 2.2.2 Configuring an LDAP Directory Connector

Related Topics

- 2.1 Using the Directory Browser
- 2.2 Adding a Directory
  - 2.2.1 Configuring an Internal Directory
  - 2.2.2 Configuring an LDAP Directory Connector
    - 2.2.2.1 Microsoft Active Directory
    - 2.2.2.2 SunONE
    - 2.2.2.3 OpenLDAP
    - 2.2.2.4 Apache Directory Server (ApacheDS)
    - 2.2.2.5 Generic LDAP Directories
    - UNPUBLISHED PENDING TESTING & SCREENSHOT- 2.2.2.2 Apple OSX Open Directory
  - 2.2.3 Configuring a Custom Directory Connector
- 2.3 Specifying Directory Permissions
- 2.4 Importing Principals and Groups into a Directory
  - 2.4.1 Importing Users from Atlassian Confluence
  - 2.4.2 Importing Users from Atlassian JIRA
  - 2.4.3 Importing Users from Jive Forums

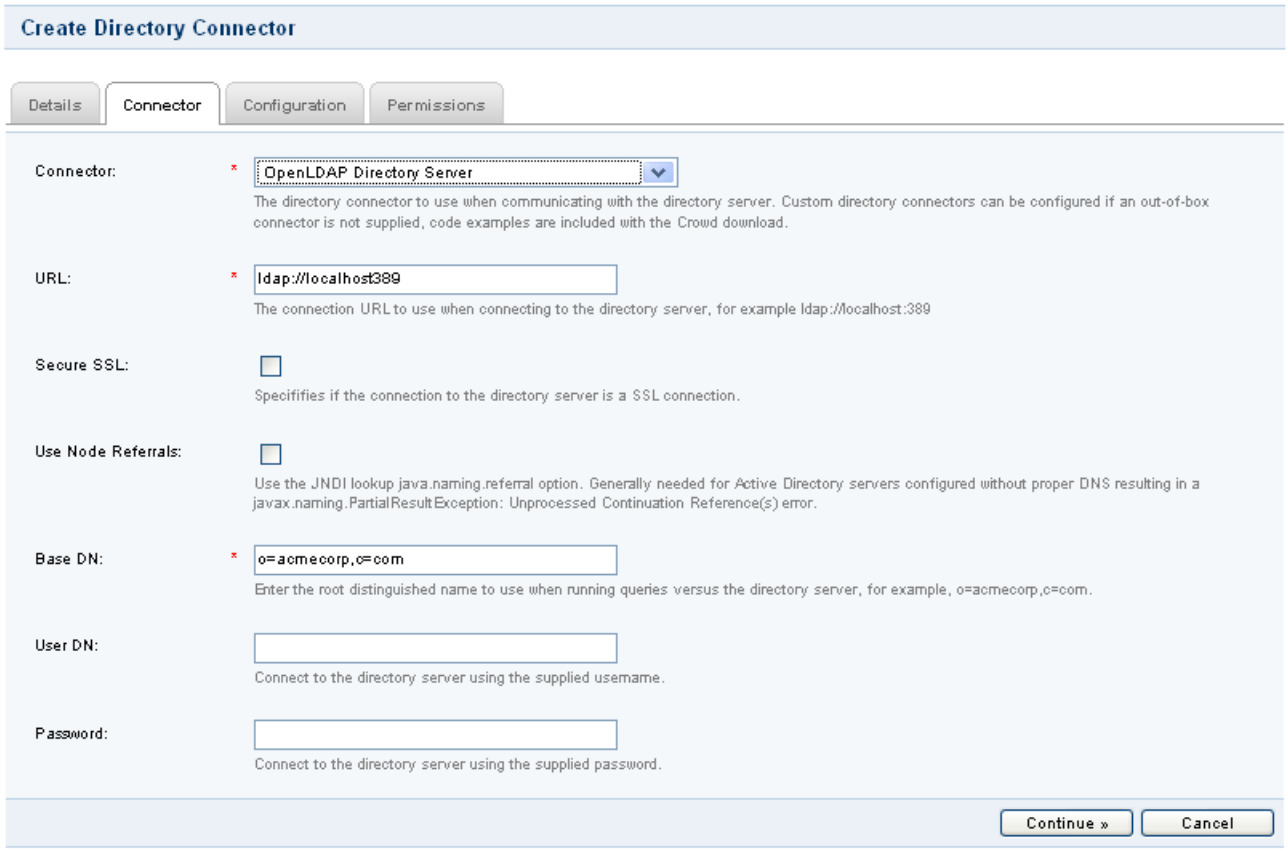
Crowd Documentation

2.2.2.3 OpenLDAP

This page last changed on Apr 02, 2007 by justen.stepka@atlassian.com.

This page provides configuration notes for OpenLDAP, in relation to 2.2.2 Configuring an LDAP Directory Connector.

Screenshot: 'Connector — OpenLDAP'



Attribute	Description
Connector	The directory connector to use when communicating with the directory server.
URL	The connection URL to use when connecting to the directory server, e.g.: `ldap://localhost:389`, or `port 639` for SSL.
Secure SSL	Specifies if the connection to the directory server is a SSL connection.
Use Node Referrals	Use the JNDI lookup java.naming.referral option. Generally needed for Active Directory servers configured without proper DNS, to prevent a 'javax.naming.PartialResultException: Unprocessed Continuation Reference(s)' error.
Use Paged Results	Use the LDAP control extension for simple paged results option. Retrieves chunks of data rather

	than all of the results at once. This feature may be necessary when using Microsoft Active Directory if more than 999 results are returned for any given search.
Base DN	Enter the root distinguished name to use when running queries versus the directory server, e.g.: `o=acmecorp,c=com`.
User DN	The username that Crowd will use when connecting to the directory server.
Password	The password that Crowd will use when connecting to the directory server.

Configuration details for OpenLDAP

OpenLDAP Directory Example	Value
Base DN	dc=exampel,dc=com
User DN	cn=Manager,dc=example,dc=com

Next Step

Go back to 2.2.2 Configuring an LDAP Directory Connector

Related Topics

- 2.1 Using the Directory Browser
- 2.2 Adding a Directory
  - 2.2.1 Configuring an Internal Directory
  - 2.2.2 Configuring an LDAP Directory Connector
    - 2.2.2.1 Microsoft Active Directory
    - 2.2.2.2 SunONE
    - 2.2.2.3 OpenLDAP
    - 2.2.2.4 Apache Directory Server (ApacheDS)
    - 2.2.2.5 Generic LDAP Directories
    - UNPUBLISHED PENDING TESTING & SCREENSHOT- 2.2.2.2 Apple OSX Open Directory
  - 2.2.3 Configuring a Custom Directory Connector
- 2.3 Specifying Directory Permissions
- 2.4 Importing Principals and Groups into a Directory
  - 2.4.1 Importing Users from Atlassian Confluence
  - 2.4.2 Importing Users from Atlassian JIRA
  - 2.4.3 Importing Users from Jive Forums

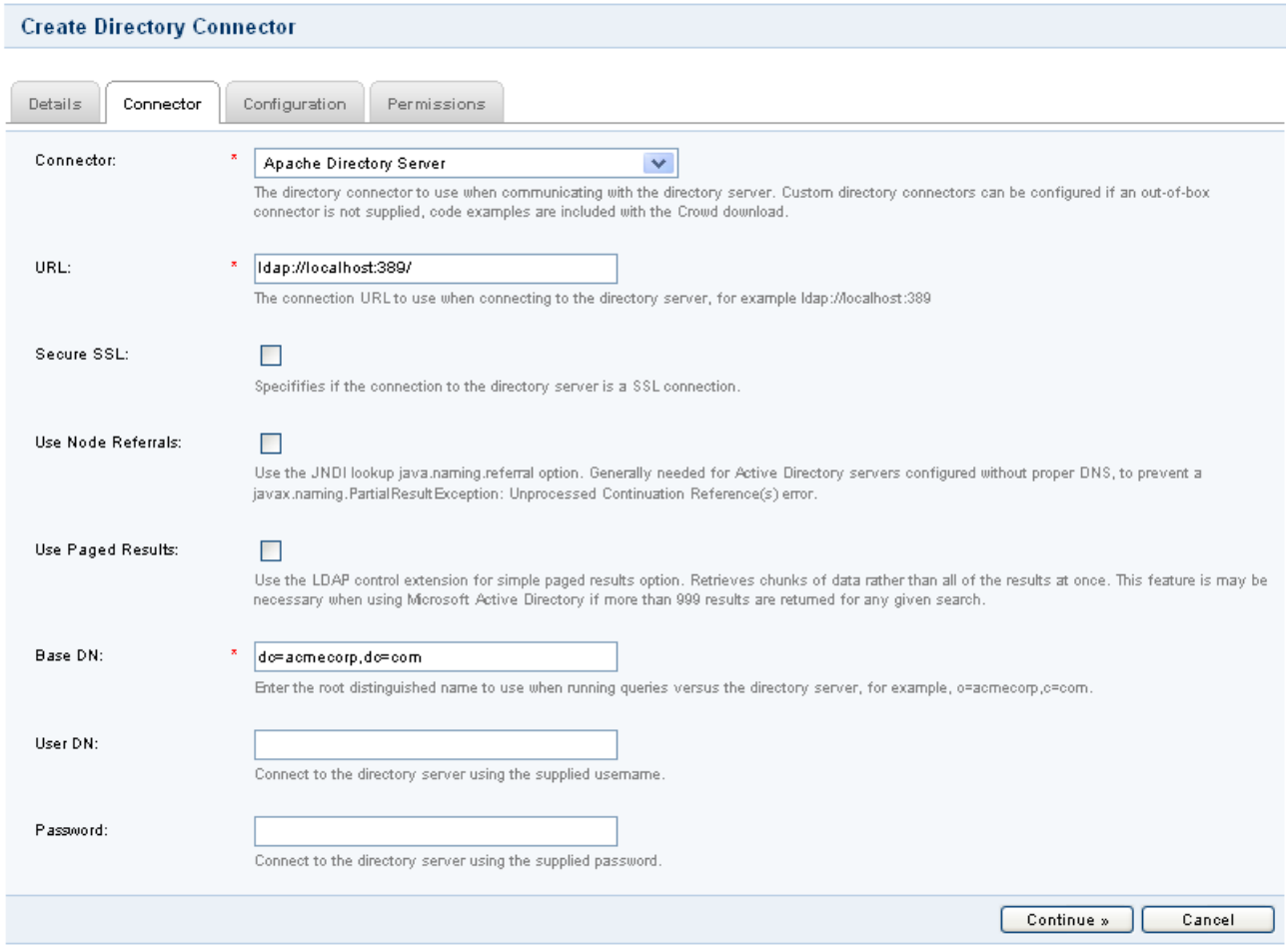
Crowd Documentation

2.2.2.4 Apache Directory Server (ApacheDS)

This page last changed on Mar 26, 2007 by rosie@atlassian.com.

This page provides configuration notes for Apache Directory Server, in relation to 2.2.2 Configuring an LDAP Directory Connector.

Screenshot: 'Connector — Apache '



Attribute	Description
Connector	The directory connector to use when communicating with the directory server.
URL	The connection URL to use when connecting to the directory server, e.g.: `ldap://localhost:389`, or `port 639` for SSL.
Secure SSL	Specifies if the connection to the directory server is a SSL connection.
Use Node Referrals	Use the JNDI lookup java.naming.referral option. Generally needed for Active Directory servers configured without proper DNS, to prevent a 'javax.naming.PartialResultException: Unprocessed Continuation Reference(s)' error.

Use Paged Results	Use the LDAP control extension for simple paged results option. Retrieves chunks of data rather than all of the results at once. This feature may be necessary when using Microsoft Active Directory if more than 999 results are returned for any given search.
Base DN	Enter the root distinguished name to use when running queries versus the directory server, e.g.: `o=acmecorp,c=com`.
User DN	The username that Crowd will use when connecting to the directory server.
Password	The password that Crowd will use when connecting to the directory server.

Configuration details for ApacheDS

OpenLDAP Directory Example	Value
Base DN	dc=example,dc=com

Next Step

Go back to 2.2.2 Configuring an LDAP Directory Connector

Related Topics

- 2.1 Using the Directory Browser
- 2.2 Adding a Directory
  - 2.2.1 Configuring an Internal Directory
  - 2.2.2 Configuring an LDAP Directory Connector
    - 2.2.2.1 Microsoft Active Directory
    - 2.2.2.2 SunONE
    - 2.2.2.3 OpenLDAP
    - 2.2.2.4 Apache Directory Server (ApacheDS)
    - 2.2.2.5 Generic LDAP Directories
    - UNPUBLISHED PENDING TESTING & SCREENSHOT- 2.2.2.2 Apple OSX Open Directory
  - 2.2.3 Configuring a Custom Directory Connector
- 2.3 Specifying Directory Permissions
- 2.4 Importing Principals and Groups into a Directory
  - 2.4.1 Importing Users from Atlassian Confluence
  - 2.4.2 Importing Users from Atlassian JIRA
  - 2.4.3 Importing Users from Jive Forums

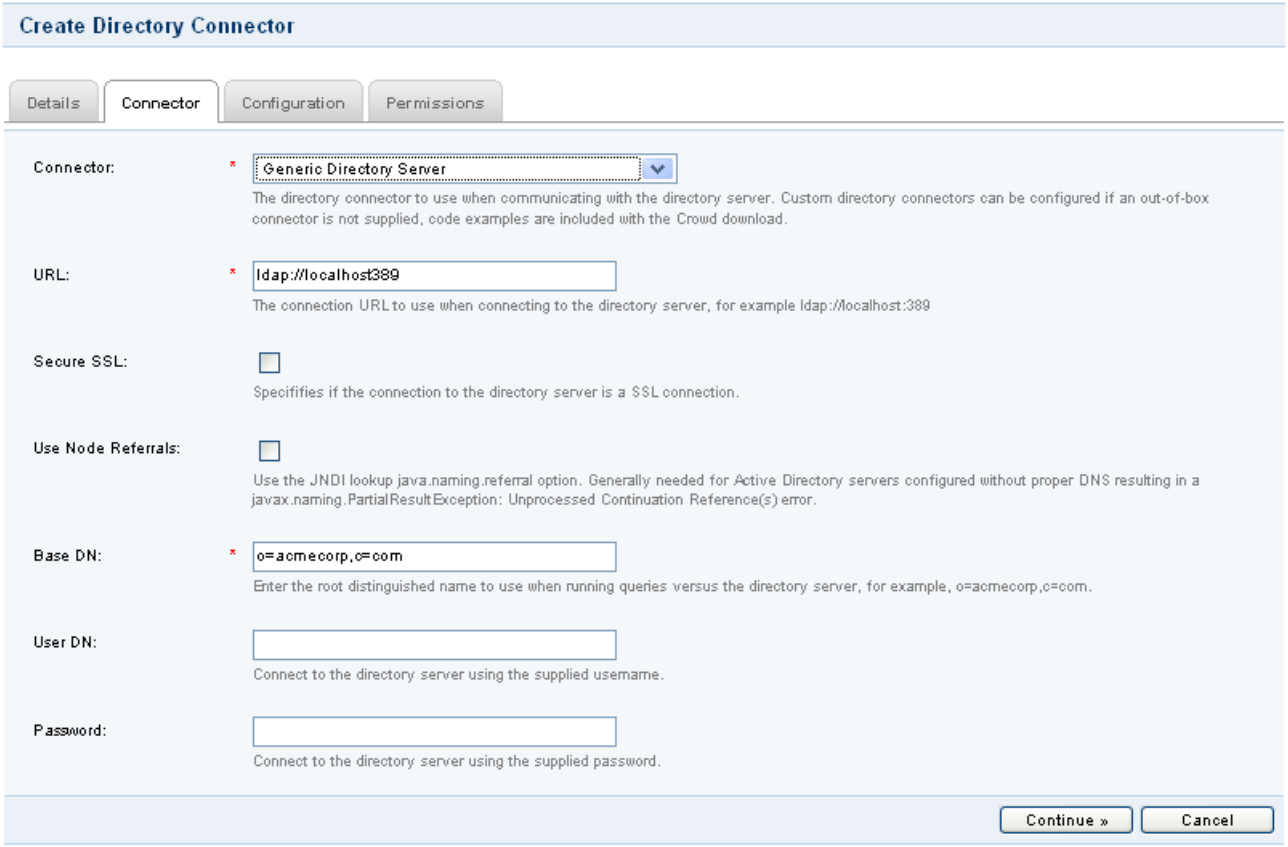
Crowd Documentation

2.2.2.5 Generic LDAP Directories

This page last changed on Mar 26, 2007 by rosie@atlassian.com.

This page provides configuration notes for generic LDAP directories, in relation to 2.2.2 Configuring an LDAP Directory Connector.

Screenshot: 'Connector — Generic Directory Server'



Attribute	Description
Connector	The directory connector to use when communicating with the directory server.
URL	The connection URL to use when connecting to the directory server, e.g.: `ldap://localhost:389`, or `port 639` for SSL.
Secure SSL	Specifies if the connection to the directory server is a SSL connection.
Use Node Referrals	Use the JNDI lookup java.naming.referral option. Generally needed for Active Directory servers configured without proper DNS, to prevent a 'javax.naming.PartialResultException: Unprocessed Continuation Reference(s)' error.
Use Paged Results	Use the LDAP control extension for simple paged results option. Retrieves chunks of data rather than all of the results at once. This feature may be

	necessary when using Microsoft Active Directory if more than 999 results are returned for any given search.
Base DN	Enter the root distinguished name to use when running queries versus the directory server, e.g.: `o=acmecorp,c=com`.
User DN	The username that Crowd will use when connecting to the directory server.
Password	The password that Crowd will use when connecting to the directory server.

Next Step

Go back to 2.2.2 Configuring an LDAP Directory Connector

Related Topics

- 2.1 Using the Directory Browser
- 2.2 Adding a Directory
  - 2.2.1 Configuring an Internal Directory
  - 2.2.2 Configuring an LDAP Directory Connector
    - 2.2.2.1 Microsoft Active Directory
    - 2.2.2.2 SunONE
    - 2.2.2.3 OpenLDAP
    - 2.2.2.4 Apache Directory Server (ApacheDS)
    - 2.2.2.5 Generic LDAP Directories
    - UNPUBLISHED PENDING TESTING & SCREENSHOT- 2.2.2.2 Apple OSX Open Directory
  - 2.2.3 Configuring a Custom Directory Connector
- 2.3 Specifying Directory Permissions
- 2.4 Importing Principals and Groups into a Directory
  - 2.4.1 Importing Users from Atlassian Confluence
  - 2.4.2 Importing Users from Atlassian JIRA
  - 2.4.3 Importing Users from Jive Forums

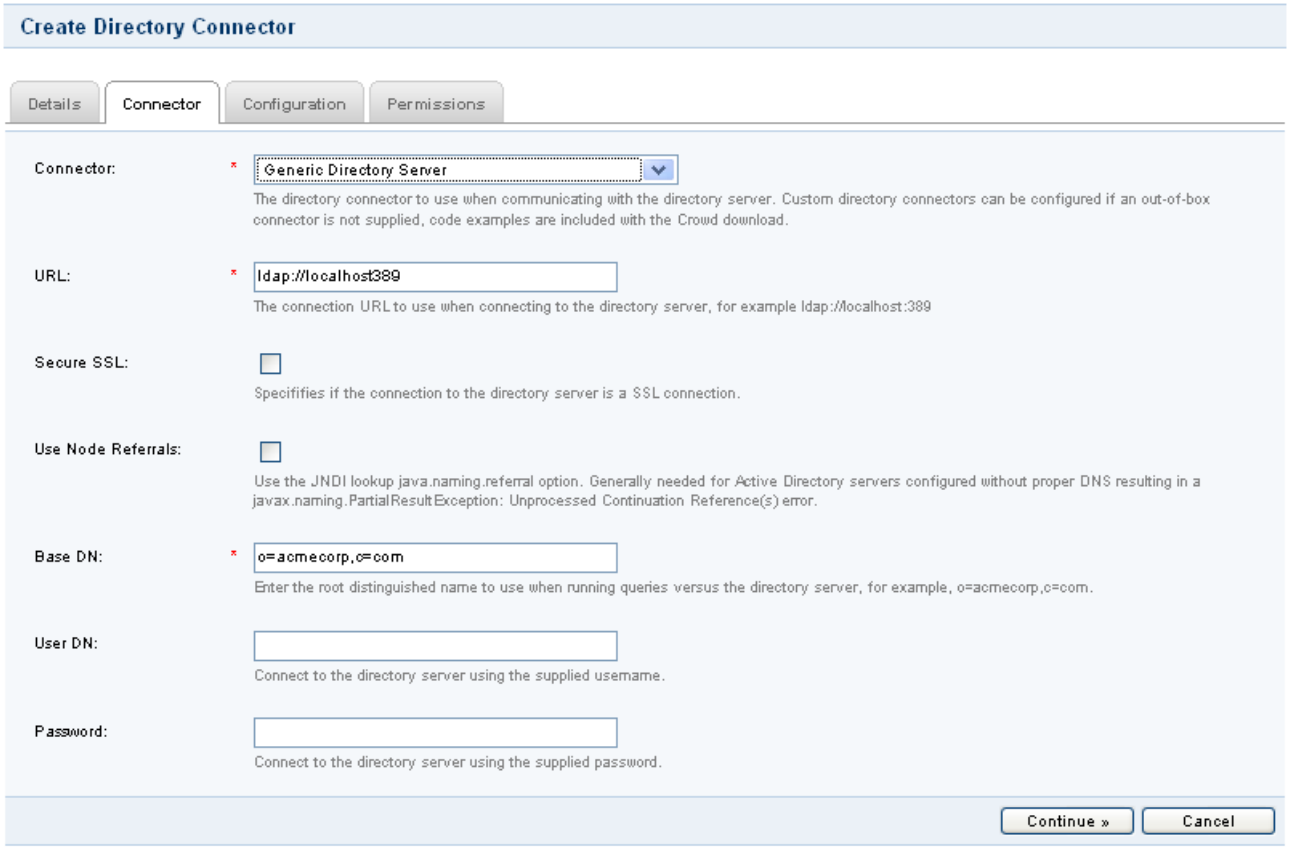
Crowd Documentation

UNPUBLISHED PENDING TESTING & SCREENSHOT- 2.2.2.2 Apple OSX Open Directory

This page last changed on Mar 15, 2007 by rosie@atlassian.com.

This page provides configuration notes for Apple OSX Open Directory, in relation to 2.2.2 Configuring an LDAP Directory Connector.

Screenshot: 'Connector — Generic Directory Server'



Attribute	Description
Connector	The directory connector to use when communicating with the directory server.
URL	The connection URL to use when connecting to the directory server, e.g.: `ldap://localhost:389`, or `port 639` for SSL.
Secure SSL	Specifies if the connection to the directory server is a SSL connection.
Use Node Referrals	Use the JNDI lookup java.naming.referral option. Generally needed for Active Directory servers configured without proper DNS, to prevent a 'javax.naming.PartialResultException: Unprocessed Continuation Reference(s)' error.
Use Paged Results	Use the LDAP control extension for simple paged results option. Retrieves chunks of data rather

	than all of the results at once. This feature may be necessary when using Microsoft Active Directory if more than 999 results are returned for any given search.
Base DN	Enter the root distinguished name to use when running queries versus the directory server, e.g.: `o=acmecorp,c=com`.
User DN	The username that Crowd will use when connecting to the directory server.
Password	The password that Crowd will use when connecting to the directory server.

Configuration details for Apple OSX Open Directory

Apple OS X Open Directory Example	Value
Base DN	dc=acmecorp,dc=com
User DN	cn=Manager,dc=acmecorp,dc=com

Next Step

Go back to 2.2.2 Configuring an LDAP Directory Connector

Related Topics

- 2.1 Using the Directory Browser
- 2.2 Adding a Directory
  - 2.2.1 Configuring an Internal Directory
  - 2.2.2 Configuring an LDAP Directory Connector
    - 2.2.2.1 Microsoft Active Directory
    - 2.2.2.2 SunONE
    - 2.2.2.3 OpenLDAP
    - 2.2.2.4 Apache Directory Server (ApacheDS)
    - 2.2.2.5 Generic LDAP Directories
    - UNPUBLISHED PENDING TESTING & SCREENSHOT- 2.2.2.2 Apple OSX Open Directory
  - 2.2.3 Configuring a Custom Directory Connector
- 2.3 Specifying Directory Permissions
- 2.4 Importing Principals and Groups into a Directory
  - 2.4.1 Importing Users from Atlassian Confluence
  - 2.4.2 Importing Users from Atlassian JIRA
  - 2.4.3 Importing Users from Jive Forums

Crowd Documentation

2.2.3 Configuring a Custom Directory Connector

This page last changed on Mar 28, 2007 by rosie@atlassian.com.

Custom directory connectors allow developers to connect Crowd to custom user-stores, such as existing databases or legacy system.

The simplest way to accomplish this is to add a JAR file with the necessary classes to the Crowd `WEB-INF/lib` folder. For details, please see Creating a Custom Directory Connector

Once you have added your JAR file to the Crowd `WEB-INF/lib` folder, you are ready to configure a Custom Directory Connector.

To configure a Custom Directory Connector,

1. Login to the Crowd Administration Console.
2. Click the 'Directories' link in the top navigation bar.
3. This will display the Directory Browser. Click the 'Add Directory' link.
4. Click the 'Custom' button.
5. Complete the fields as described in the table below.
6. Click the 'Continue' button to configure the directory's permissions.

Once you have configured the directory's permissions, you will have finished configuring your new directory. You can then map the directory to appropriate applications.

Screenshot: 'Create Custom Directory'

|| Custom Directory Store Attributes || Description ||

Name	The name used to identify the directory within Crowd. This is useful when there are multiple directories configured, e.g. Chicago Employees or Web Customers.

Description	Details about this specific directory.
Active	Only deselect this if you wish to prevent all users ('principals') within the directory from accessing all mapped applications.
Implementation Class	Implementation of `com.atlassian.crowd.integration.directory.RemoteDirecto` Java interface. Must be in the Crowd CLASSPATH.

Next Step:

See 2.3 Specifying Directory Permissions

Related Topics

- 2.1 Using the Directory Browser
- 2.2 Adding a Directory
  - 2.2.1 Configuring an Internal Directory
  - 2.2.2 Configuring an LDAP Directory Connector
    - 2.2.2.1 Microsoft Active Directory
    - 2.2.2.2 SunONE
    - 2.2.2.3 OpenLDAP
    - 2.2.2.4 Apache Directory Server (ApacheDS)
    - 2.2.2.5 Generic LDAP Directories
    - UNPUBLISHED PENDING TESTING & SCREENSHOT- 2.2.2.2 Apple OSX Open Directory
  - 2.2.3 Configuring a Custom Directory Connector
- 2.3 Specifying Directory Permissions
- 2.4 Importing Principals and Groups into a Directory
  - 2.4.1 Importing Users from Atlassian Confluence
  - 2.4.2 Importing Users from Atlassian JIRA
  - 2.4.3 Importing Users from Jive Forums

Crowd Documentation

2.3 Specifying Directory Permissions

This page last changed on Mar 30, 2007 by rosie@atlassian.com.

Directory permissions allow you to restrict the way in which directories can be used by mapped applications. Often, administrators need to limit applications to only being able to read — not modify — directory entity data, i.e. the users ('principals') groups and roles contained within the directory. You can achieve this by disabling the relevant directory permissions.

Permission	Description
Add Group	Allows applications to add groups to the directory.
Add Principal	Allows applications to add principals to the directory.
Add Role	Allows applications to add roles to the directory.
Modify Group	Allows applications to modify groups in the directory.
Modify Principal	Allows applications to modify principals in the directory.
Modify Role	Allows applications to modify roles in the directory.
Remove Group	Allows applications to delete groups from the directory.
Remove Principal	Allows applications to delete principals from the directory.
Remove Role	Allows applications to delete roles from the directory.

When you add a new directory, all of its permissions are enabled by default.

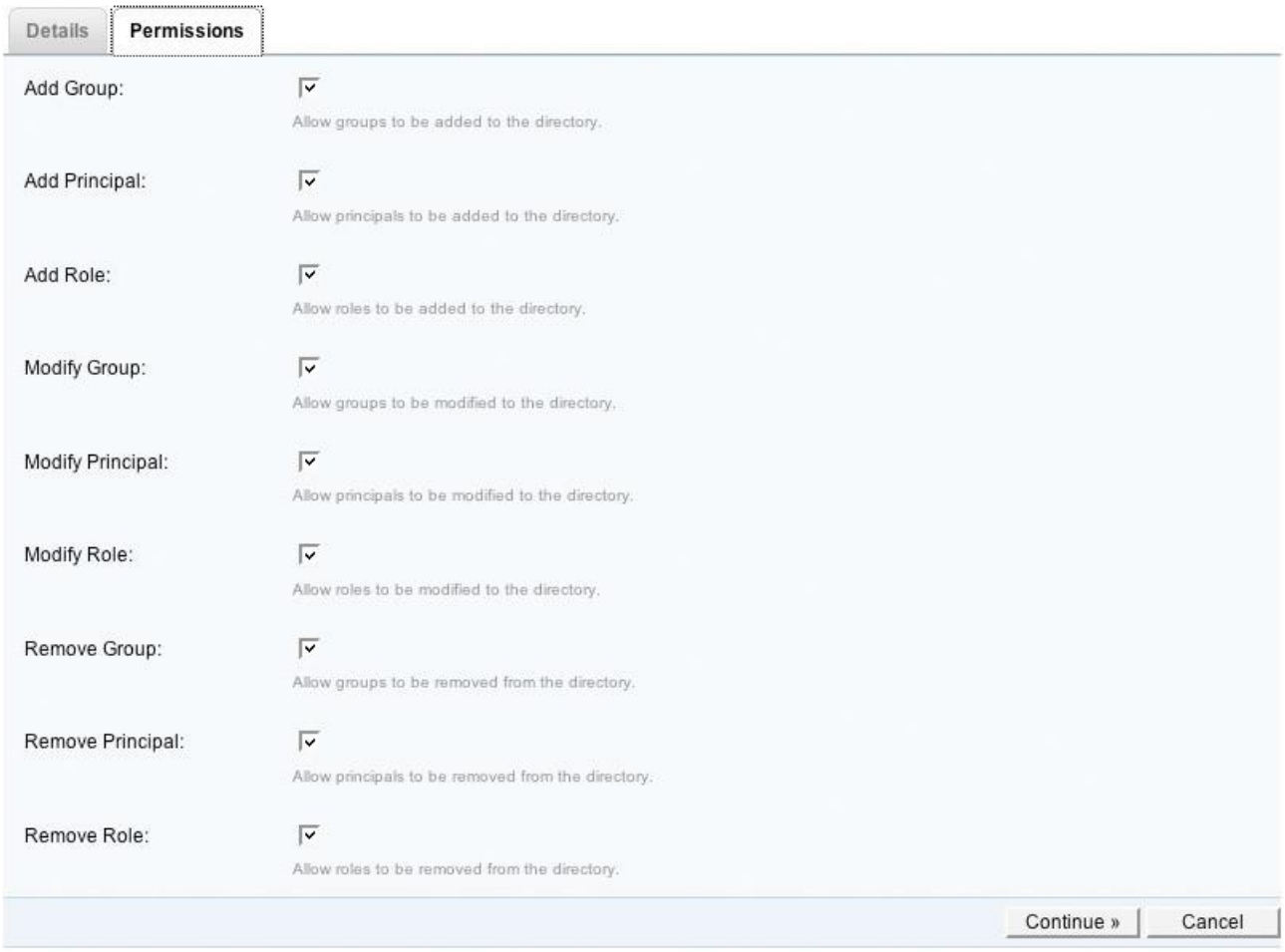
⚠ Directory permissions apply to all mapped applications, including the Crowd Administration Console. If you disable a directory's permissions, some of the functionality described in 4. Managing Principals, Groups and Roles may be unavailable.

To specify directory permissions,

1. Configure a new directory as described in 2.2 Adding a Directory or select an existing directory from the Directory Browser.
2. Click the 'Permissions' tab. This will display a list of permissions as shown in the screenshot below.

- To enable a directory permission, select the corresponding check-box.
- To disable a directory permission, deselect the corresponding check-box.

Screenshot: 'Directory Permissions'



See Also

To control which users within a directory may access a mapped application, see 3.4 Specifying which Groups can access an Application.

Related Topics

- 2.1 Using the Directory Browser
- 2.2 Adding a Directory
  - 2.2.1 Configuring an Internal Directory
  - 2.2.2 Configuring an LDAP Directory Connector
    - 2.2.2.1 Microsoft Active Directory
    - 2.2.2.2 SunONE
    - 2.2.2.3 OpenLDAP
    - 2.2.2.4 Apache Directory Server (ApacheDS)
    - 2.2.2.5 Generic LDAP Directories
    - UNPUBLISHED PENDING TESTING & SCREENSHOT- 2.2.2.2 Apple OSX Open Directory
  - 2.2.3 Configuring a Custom Directory Connector
- 2.3 Specifying Directory Permissions
- 2.4 Importing Principals and Groups into a Directory
  - 2.4.1 Importing Users from Atlassian Confluence
  - 2.4.2 Importing Users from Atlassian JIRA

- 2.4.3 Importing Users from Jive Forums

Crowd Documentation

2.4 Importing Principals and Groups into a Directory

This page last changed on Mar 29, 2007 by rosie@atlassian.com.

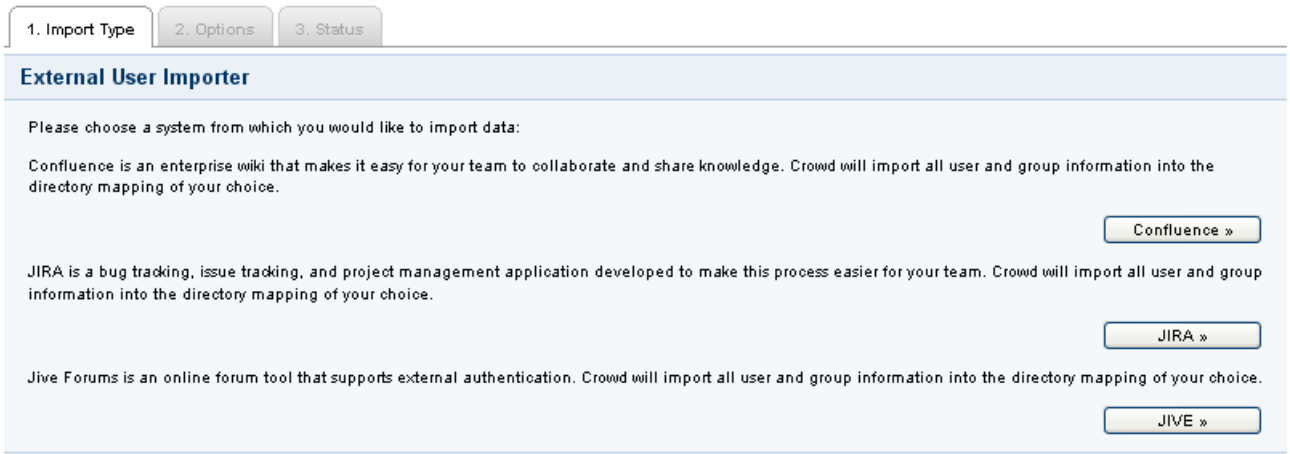
Once you have added a directory, you can import groups and users (or 'principals', as they are known in Crowd) into it from external user-stores. This can reduce the number of user-stores within your organisation, and give you a consolidated, centralised point of user management. Once you have imported users into a Crowd directory, you can manage them via the Crowd Administration Console (assuming the directory's permissions allow this).
For example, your organisation might currently have user IDs for Atlassian JIRA users stored within JIRA's database, and user IDs for Jive Forums users stored within Jive's database. You could use Crowd to import all the user IDs from both places into Microsoft Active Directory.

You can import from different user-stores into a single Crowd directory, or into different Crowd directories, depending on your needs.

To import users into a directory,

1. Login to the Crowd Administration Console.
2. Click the 'Principals' link in the top navigation bar.
3. This will display the Principal Browser. Click the 'Import Users' link.
4. This will display the 'Import Type' screen (see below). Click the button corresponding to the type of user-store from which you want to import external users into Crowd:
   - 'Confluence' — see 2.4.1 Importing Users from Atlassian Confluence
   - 'JIRA' — see 2.4.2 Importing Users from Atlassian JIRA
   - 'Jive Forums' — see 2.4.3 Importing Users from Jive Forums

Screenshot: 'Select Import Type'



Related Topics

- 2.1 Using the Directory Browser
- 2.2 Adding a Directory
  - 2.2.1 Configuring an Internal Directory
  - 2.2.2 Configuring an LDAP Directory Connector
    - 2.2.2.1 Microsoft Active Directory

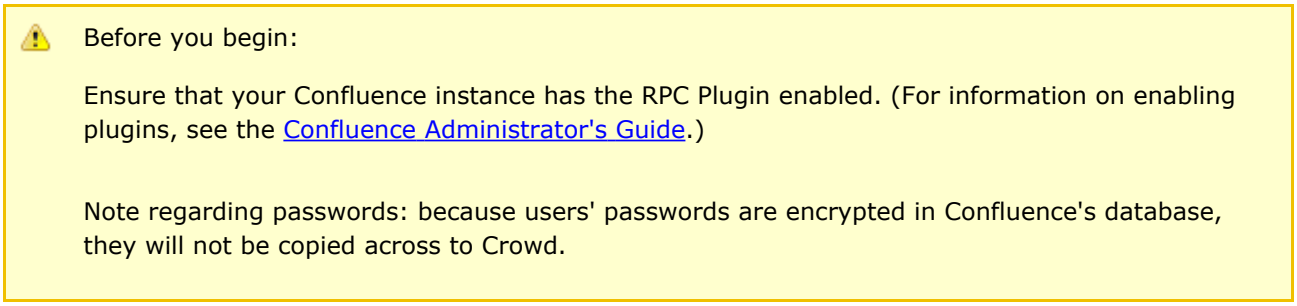
        - 2.2.2.2 SunONE
        - 2.2.2.3 OpenLDAP
        - 2.2.2.4 Apache Directory Server (ApacheDS)
        - 2.2.2.5 Generic LDAP Directories
        - UNPUBLISHED PENDING TESTING & SCREENSHOT- 2.2.2.2 Apple OSX Open Directory
    - 2.2.3 Configuring a Custom Directory Connector
- 2.3 Specifying Directory Permissions
- 2.4 Importing Principals and Groups into a Directory
    - 2.4.1 Importing Users from Atlassian Confluence
    - 2.4.2 Importing Users from Atlassian JIRA
    - 2.4.3 Importing Users from Jive Forums

Crowd Documentation

2.4.1 Importing Users from Atlassian Confluence

This page last changed on Apr 12, 2007 by justen.stepka@atlassian.com.

If you have already been using Atlassian Confluence, and are now configuring Confluence as a Crowd application, you will probably want to import your existing Confluence users and groups into a Crowd directory.



Before you begin:

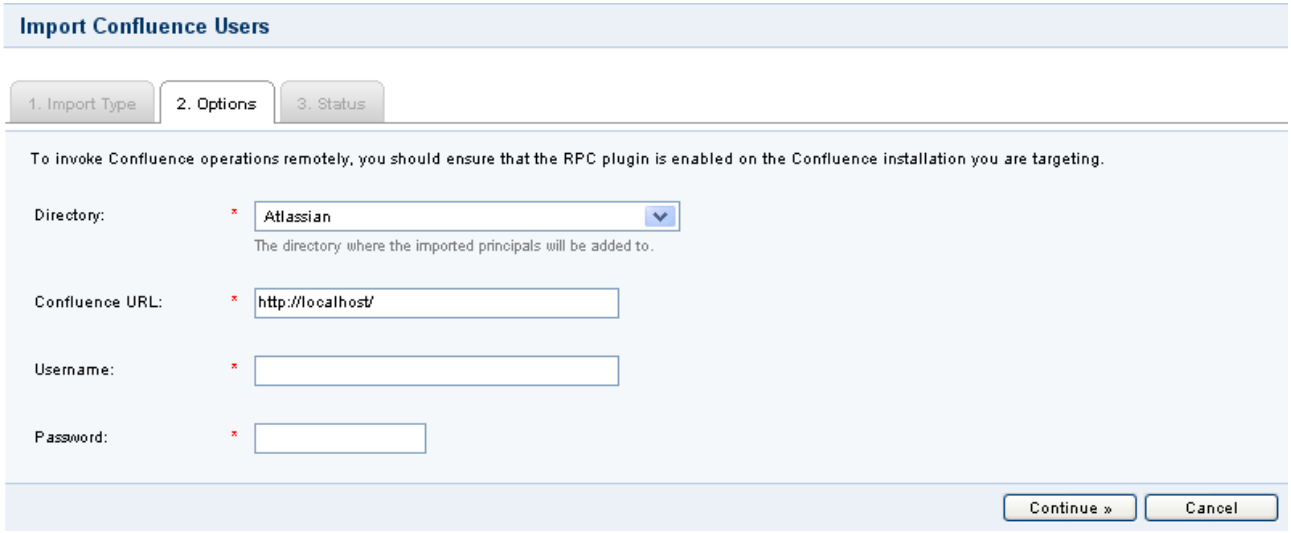
Ensure that your Confluence instance has the RPC Plugin enabled. (For information on enabling plugins, see the Confluence Administrator's Guide.)

Note regarding passwords: because users' passwords are encrypted in Confluence's database, they will not be copied across to Crowd.

To import users and groups from Atlassian Confluence into a Crowd directory,

1. Login to the Crowd Administration Console.
2. Click the 'Principals' link in the top navigation bar.
3. This will display the Principal Browser. Click the 'Import Users' link.
4. This will display the 'Import Type' screen. Click the 'Confluence' button.
5. This will display the 'Options' screen. Complete the fields as follows:
   - 'Directory' — select the directory that is mapped to the Confluence application.
   - 'Confluence URL' — type the URL of your Confluence instance.
   - 'Username' — type the Confluence user ID that Crowd will use to login to your Confluence instance. Note that this user ID should have Confluence administration rights.
   - 'Password' — type the password of the Confluence user ID that Crowd will use to login to your Confluence instance.
6. Click the 'Continue' button to import the users from your Confluence instance into your Crowd directory.
7. The 'Status' screen will be displayed, showing how many users and groups have been imported into your Crowd directory.
8. Click the 'Principals' button to view and manage the imported users and groups via the Crowd Administration Console (assuming the directory's permissions allow this).

Screenshot: 'Import Confluence Users'



Next Step

To give the imported groups access to the Confluence application, see 3.4 Specifying which Groups can access an Application.

Related Topics

- 2.1 Using the Directory Browser
- 2.2 Adding a Directory
  - 2.2.1 Configuring an Internal Directory
  - 2.2.2 Configuring an LDAP Directory Connector
    - 2.2.2.1 Microsoft Active Directory
    - 2.2.2.2 SunONE
    - 2.2.2.3 OpenLDAP
    - 2.2.2.4 Apache Directory Server (ApacheDS)
    - 2.2.2.5 Generic LDAP Directories
    - UNPUBLISHED PENDING TESTING & SCREENSHOT- 2.2.2.2 Apple OSX Open Directory
  - 2.2.3 Configuring a Custom Directory Connector
- 2.3 Specifying Directory Permissions
- 2.4 Importing Principals and Groups into a Directory
  - 2.4.1 Importing Users from Atlassian Confluence
  - 2.4.2 Importing Users from Atlassian JIRA
  - 2.4.3 Importing Users from Jive Forums

Crowd Documentation

2.4.2 Importing Users from Atlassian JIRA

This page last changed on Mar 29, 2007 by rosie@atlassian.com.

If you have already been using Atlassian JIRA, and are now configuring JIRA as a Crowd application, you will probably want to import your existing JIRA users and groups into a Crowd directory.

> Before you begin:
>
> You will need to have installed the JIRA database's JDBC drivers in the Crowd CLASS-PATH.
>
> Note regarding passwords: because users' passwords are encrypted in JIRA's database, they will not be copied across to Crowd.

To import users and groups from Atlassian JIRA into a Crowd directory,

1. Login to the Crowd Administration Console.
2. Click the 'Principals' link in the top navigation bar.
3. This will display the Principal Browser. Click the 'Import Users' link.
4. This will display the 'Import Type' screen. Click the 'JIRA' button.
5. This will display the 'Options' screen. Complete the fields as follows:
   - 'Directory' — select the directory that is mapped to the JIRA application.
   - 'DB URL' — type the URL of your JIRA instance's database.
   - 'DB Driver' — type the name of your JIRA instance's database JDBC driver.
   - 'Username' — type the username of the database user that Crowd will use to login to your JIRA instance's database.
   - 'Password' — type the password of the database user Crowd will use to login to your JIRA instance's database.

     The import process will log in to the database, not to JIRA.
6. Click the 'Continue' button to import the users from your JIRA instance into your Crowd directory.
7. The 'Status' screen will be displayed, showing how many users and groups have been imported into your Crowd directory.
8. Click the 'Principals' button to view and manage the imported users and groups via the Crowd Administration Console (assuming the directory's permissions allow this).

Screenshot: 'Import JIRA Users'

Next Step

To give the imported groups access to the JIRA application, see 3.4 Specifying which Groups can access an Application.

Related Topics

- 2.1 Using the Directory Browser
- 2.2 Adding a Directory
  - 2.2.1 Configuring an Internal Directory
  - 2.2.2 Configuring an LDAP Directory Connector
    - 2.2.2.1 Microsoft Active Directory
    - 2.2.2.2 SunONE
    - 2.2.2.3 OpenLDAP
    - 2.2.2.4 Apache Directory Server (ApacheDS)
    - 2.2.2.5 Generic LDAP Directories
    - UNPUBLISHED PENDING TESTING & SCREENSHOT- 2.2.2.2 Apple OSX Open Directory
  - 2.2.3 Configuring a Custom Directory Connector
- 2.3 Specifying Directory Permissions
- 2.4 Importing Principals and Groups into a Directory
  - 2.4.1 Importing Users from Atlassian Confluence
  - 2.4.2 Importing Users from Atlassian JIRA
  - 2.4.3 Importing Users from Jive Forums

Crowd Documentation

2.4.3 Importing Users from Jive Forums

This page last changed on Mar 29, 2007 by rosie@atlassian.com.

If you have already been using Jive Forums, and are now configuring Jive Forms as a Crowd application, you will probably want to import your existing Jive users and groups into a Crowd directory.

Before you begin:

The database drivers for the Jive Forums database will need to be on Crowd's classpath. To do this, simply copy the database driver JAR for your particular Jive database across to `CROWD/apache-tomcat-5.5.20/common/lib` and restart Crowd.

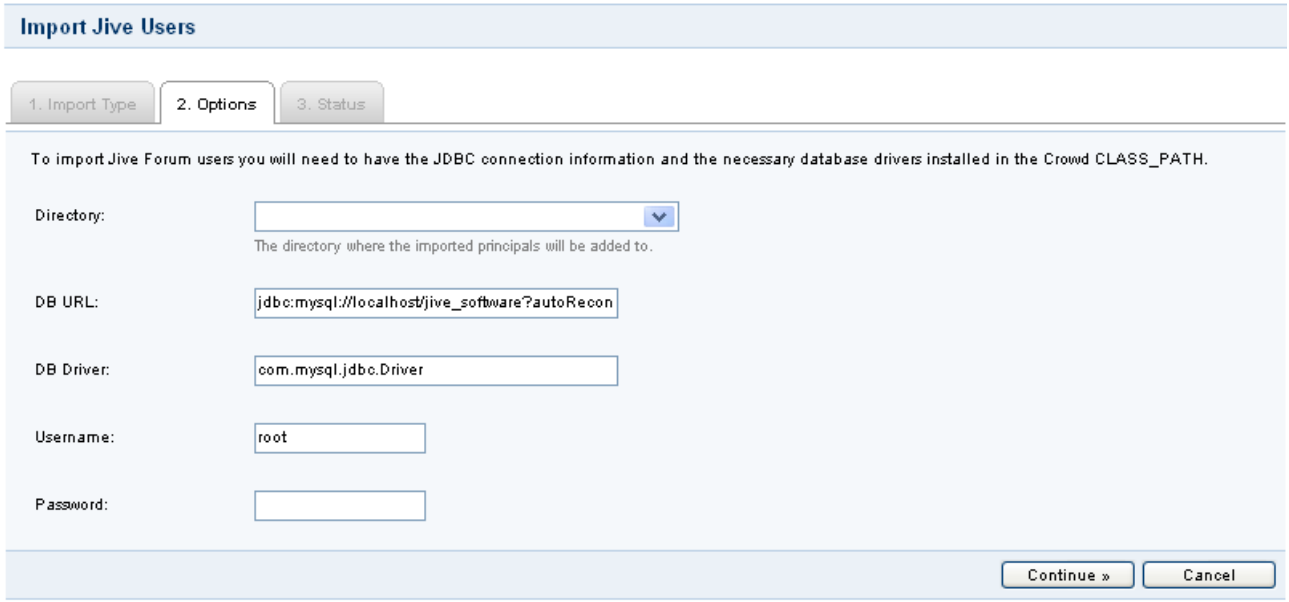
Note: the passwords for users in Jive will not be copied across to Crowd as they are stored as hashes in Jive's internal database.

To import users and groups from Jive Forums into a Crowd directory,

1. Login to the Crowd Administration Console.
2. Click the 'Principals' link in the top navigation bar.
3. This will display the Principal Browser. Click the 'Import Users' link.
4. This will display the 'Import Type' screen. Click the 'JIRA' button.
5. This will display the 'Options' screen. Complete the fields as follows:
    - 'Directory' — select the directory that is mapped to the Jive Forums application.
    - 'DB URL' — type the URL of Jive's database.
    - 'DB Driver' — type the name of Jive's database JDBC driver.
    - 'Username' — type the username of the database user that Crowd will use to login to Jive's database.
    - 'Password' — type the password of the database user Crowd will use to login to Jive's database.

     The import process will log in to the database, not to Jive Forums.
6. Click the 'Continue' button to import the users from Jive Forums into your Crowd directory.
7. The 'Status' screen will be displayed, showing how many users and groups have been imported into your Crowd directory.
8. Click the 'Principals' button to view and manage the imported users and groups via the Crowd Administration Console (assuming the directory's permissions allow this).

Screenshot: 'Import Jive Users'



Next Step

To give the imported groups access to the Jive Forums application, see 3.4 Specifying which Groups can access an Application.

Related Topics

- 2.1 Using the Directory Browser
- 2.2 Adding a Directory
  - 2.2.1 Configuring an Internal Directory
  - 2.2.2 Configuring an LDAP Directory Connector
    - 2.2.2.1 Microsoft Active Directory
    - 2.2.2.2 SunONE
    - 2.2.2.3 OpenLDAP
    - 2.2.2.4 Apache Directory Server (ApacheDS)
    - 2.2.2.5 Generic LDAP Directories
    - UNPUBLISHED PENDING TESTING & SCREENSHOT- 2.2.2.2 Apple OSX Open Directory
  - 2.2.3 Configuring a Custom Directory Connector
- 2.3 Specifying Directory Permissions
- 2.4 Importing Principals and Groups into a Directory
  - 2.4.1 Importing Users from Atlassian Confluence
  - 2.4.2 Importing Users from Atlassian JIRA
  - 2.4.3 Importing Users from Jive Forums

Crowd Documentation

3. Managing Applications

This page last changed on Mar 30, 2007 by rosie@atlassian.com.

Crowd integrates and provisions applications. Once defined, an application is mapped to a directory(s), whose users are then granted access to the application. Note that an application can only communicate with Crowd when the application uses a known host address.

- 3.1 Using the Application Browser
- 3.2 Adding an Application
    - 3.2.1 Integrating Crowd with Apache
    - 3.2.2 Integrating Crowd with Atlassian Bamboo
    - 3.2.3 Integrating Crowd with Atlassian Confluence
    - 3.2.4 Integrating Crowd with Atlassian JIRA
    - 3.2.5 Integrating Crowd with Cenqua FishEye
    - 3.2.6 Integrating Crowd with Jive Forums
        - 3.2.6.1 Jive SSO
    - 3.2.7 Integrating Crowd with a Custom Application
- 3.3 Mapping a Directory to an Application
    - 3.3.1 Specifying the Directory Order for an Application
- 3.4 Specifying which Groups can access an Application
- 3.5 Specifying an Application's Address or Hostname
- 3.6 Managing an Application's Session
- 3.7 Deleting or Deactivating an Application

3.1 Using the Application Browser

This page last changed on Apr 02, 2007 by rosie@atlassian.com.

About Applications

Crowd integrates and provisions applications. Once defined, an application is mapped to a directory(s), whose users are then granted access to the application. Note that an application can only communicate with Crowd when the application uses a known host address.

Default Applications

When you first use the Application Browser, you will see two default applications:

- 'crowd' — this is the Crowd Administration Console (i.e. the Crowd Administration Console is itself a web-application that is provisioned by Crowd). The 'crowd' application is mapped to the default directory which you defined during setup, and can be accessed by members of the `crowd-administrators` group.
- 'demo' — this is the 'demo' application which you (optionally) installed during setup. Its main purpose is to provide an example of how to integrate custom applications with Crowd. The page 3.2.7.1.1 Sample Application ('demo') does not exist.

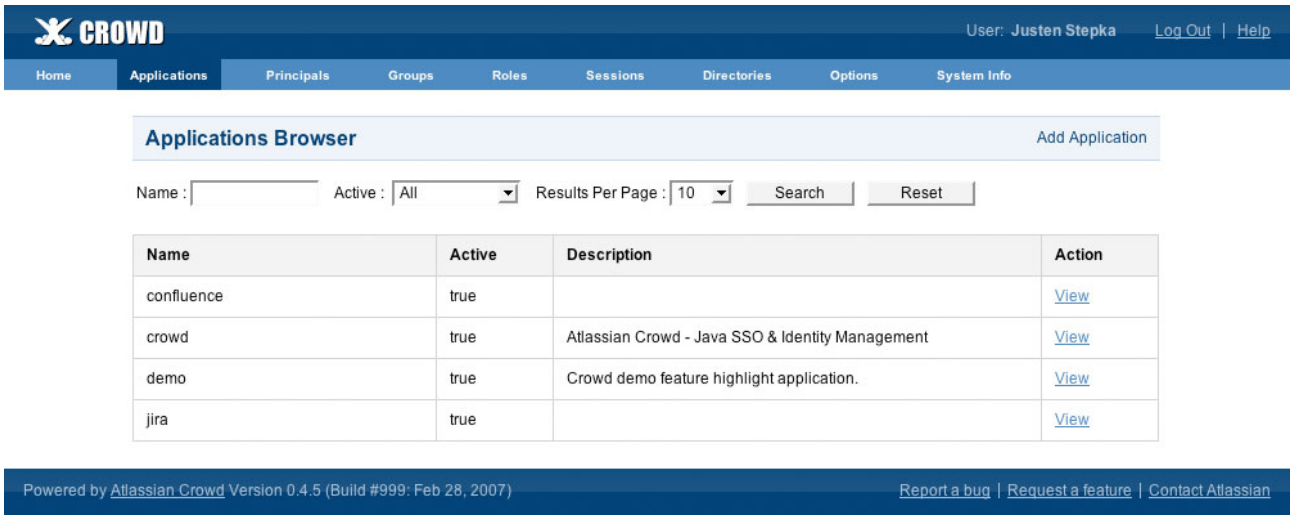
About the Application Browser

The Application Browser allows you to view and search for integrated applications.

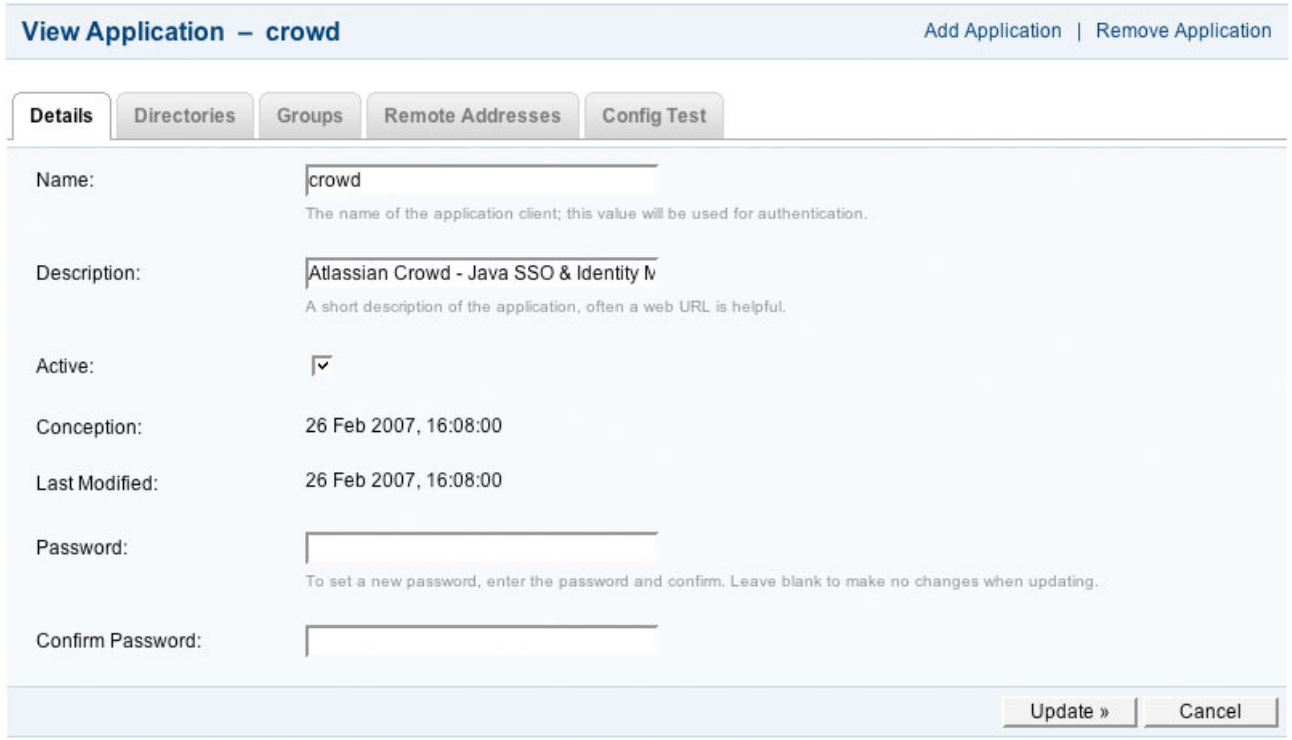
To use the Application Browser,

1. Login to the Crowd Administration Console.
2. Click the 'Applications' link in the top navigation bar.
3. This will display the Application Browser, showing all the applications that exist in your Crowd system. You can refine your search by specifying a 'Name' (note that this is case-sensitive), or 'Active'/'Inactive' applications.
4. To view/edit an applications's details, click the 'View' link.

Screenshot 1: 'Application Browser'



Screenshot 2: 'View Application'



Related Topics

- 3.1 Using the Application Browser
- 3.2 Adding an Application
  - 3.2.1 Integrating Crowd with Apache
  - 3.2.2 Integrating Crowd with Atlassian Bamboo
  - 3.2.3 Integrating Crowd with Atlassian Confluence
  - 3.2.4 Integrating Crowd with Atlassian JIRA
  - 3.2.5 Integrating Crowd with Cenqua FishEye
  - 3.2.6 Integrating Crowd with Jive Forums
    - 3.2.6.1 Jive SSO
  - 3.2.7 Integrating Crowd with a Custom Application
- 3.3 Mapping a Directory to an Application
  - 3.3.1 Specifying the Directory Order for an Application

- 3.4 Specifying which Groups can access an Application
- 3.5 Specifying an Application's Address or Hostname
- 3.6 Managing an Application's Session
- 3.7 Deleting or Deactivating an Application

Crowd Documentation

3.2 Adding an Application

This page last changed on Apr 04, 2007 by mjensen.

There are two overall steps to integrating an application with Crowd:

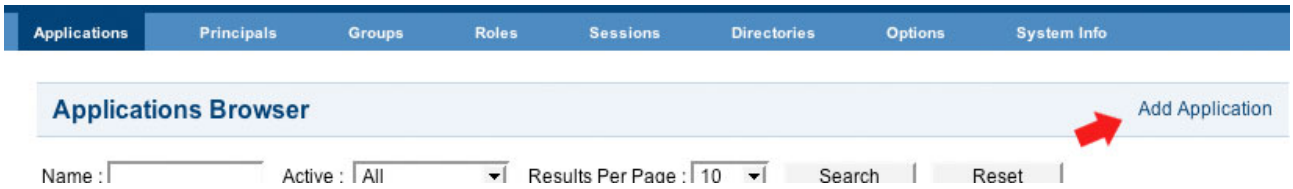
- Step 1. Configure Crowd to talk to the application — that is, add the application to Crowd via the Crowd Administration Console (see below). The application will then be allowed to authenticate against Crowd.

- Step 2. Configure the application to talk to Crowd — that is, install the Crowd Client into the application and configure the application to forward users' authentication and security requests to Crowd. Please see details for your specific application:
  - 3.2.1 Integrating Crowd with Apache
  - 3.2.2 Integrating Crowd with Atlassian Bamboo
  - 3.2.3 Integrating Crowd with Atlassian Confluence
  - 3.2.4 Integrating Crowd with Atlassian JIRA
  - 3.2.5 Integrating Crowd with Cenqua FishEye
  - 3.2.6 Integrating Crowd with Jive Forums
  - 3.2.7 Integrating Crowd with a Custom Application

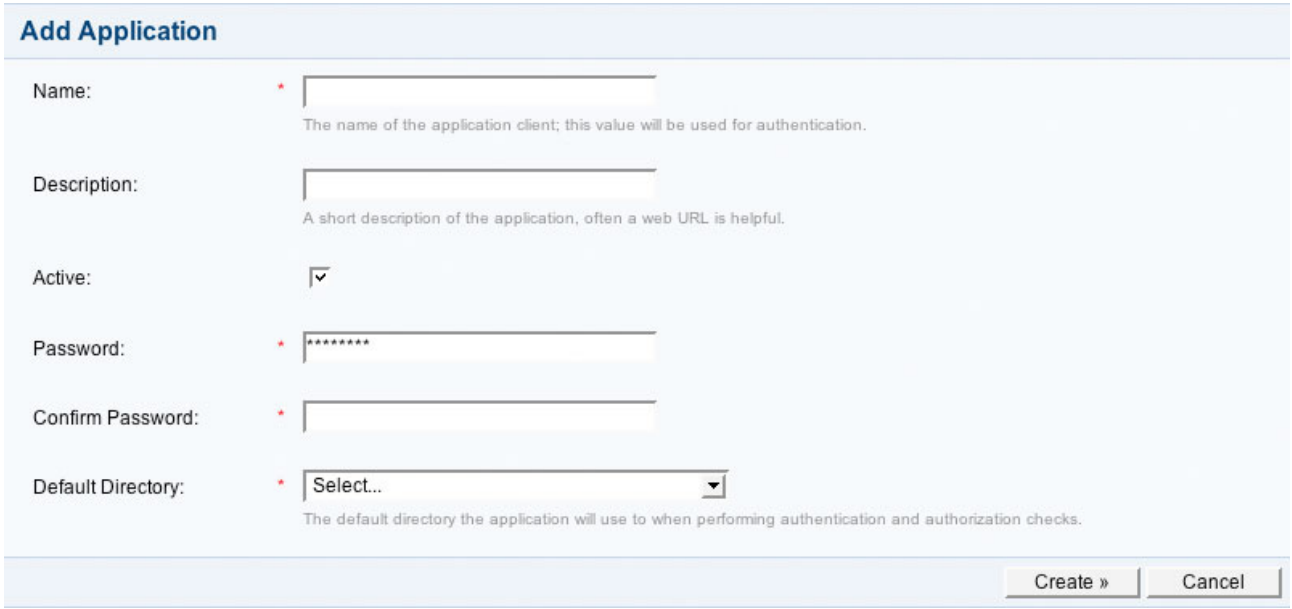
To add an application to Crowd,

1. Login to the Crowd Administration Console.
2. Click the 'Applications link in the top navigation bar.
3. This will display the Application Browser. Click the 'Add Application' link.
4. This will display the 'Add Application' screen (see screenshot). Complete the fields as described in the table below. Note that you will need to select a suitable directory to contain the application's users.
5. Click the 'Create' button to create the application. A number of tabs will now be displayed
6. To choose which users within the directory may authenticate against the application, either:
   - Click the 'Groups' tab and select one or more groups of users, then click the 'Add' button; OR
   - Click the 'Directories' tab and change 'Allow all to authenticate' to 'True' (the default is 'False').
7. Click the 'Remote Addresses' tab and specify the IP address or hostname of the application (the default is 'localhost').
8. Click the 'Config Test' tab.
9. Enter the 'Username' and 'Password' that you have just defined, then click the 'Update' button. This will (a) save your Groups/Directories and Remote Addresses; and (b) verify that the application can successfully login to the Crowd framework.

Screenshot 1: 'Application Browser'



Screenshot 2: 'Add Application'



Attribute	Description
Name	The username which the application will use when it authenticates against the Crowd framework as a client. This value must be unique, i.e. it cannot be used by more than one application client.
Description	A short description of the application. Note: a web URL is often helpful.
Active	Only deselect this if you wish to prevent all users (from all directories) from accessing this application.
Password	The password which the application will use when it authenticates against the Crowd framework as a client.
Default Directory	A directory that contains relevant users. Note: additional directories can be added later.

Related Topics

- 3.1 Using the Application Browser
- 3.2 Adding an Application
  - 3.2.1 Integrating Crowd with Apache
  - 3.2.2 Integrating Crowd with Atlassian Bamboo
  - 3.2.3 Integrating Crowd with Atlassian Confluence
  - 3.2.4 Integrating Crowd with Atlassian JIRA
  - 3.2.5 Integrating Crowd with Cenqua FishEye
  - 3.2.6 Integrating Crowd with Jive Forums
    - 3.2.6.1 Jive SSO
  - 3.2.7 Integrating Crowd with a Custom Application
- 3.3 Mapping a Directory to an Application
  - 3.3.1 Specifying the Directory Order for an Application

- 3.4 Specifying which Groups can access an Application
- 3.5 Specifying an Application's Address or Hostname
- 3.6 Managing an Application's Session
- 3.7 Deleting or Deactivating an Application

Crowd Documentation

3.2.1 Integrating Crowd with Apache

This page last changed on May 08, 2007 by justin.

IN DEVELOPMENT

The attached Perl module is in beta, use the module with caution.

- (Apache-CrowdAuth-0.03.zip) - Works with Perl 1.99 and 2.

If you find any problems with the module please comment on the following ticket:

- http://jira.atlassian.com/browse/CWD-97

Introduction

This documentation describes how to configure Crowd to authenticate HTTP Authentication requests made to an Apache webserver.

- These instructions assume some Unix system and Apache configuration knowledge.

Prerequisites

- Apache web server version 2.0 or above with the `mod_perl` module installed and configured.
- `SOAP::Lite` perl module (v0.69 or greater recommended).

Installation and Configuration

The following instructions are for Unix systems. If you're running Apache on Windows, see the notes below.

Installing the **`SOAP::Lite`** Perl Module

`SOAP::Lite` is a Perl library for managing SOAP calls. It is used by the `CrowdAuth` module to talk to the Crowd server.

The easiest way to install `SOAP::Lite` is via CPAN, by running the following command.

```
perl -MCPAN -e 'install SOAP::Lite'
```

Alternatively, you can download and install the package manually.

Installing the **Apache::CrowdAuth** Perl Module

Download the Apache-CrowdAuth-0.03.tar.gz file and extract and install it as follows:

```
tar xvzf Apache-CrowdAuth-0.03.tar.gz
cd Apache-CrowdAuth-0.03
perl Makefile.PL
make
make install
```

Configuring Apache

Ensure that **mod_perl** is enabled.

Your Apache config file should contain a line like the following:

```
LoadModule perl_module modules/mod_perl.so
```

Many common distributions of Apache come with `mod_perl` preconfigured.

Configure Authentication

To tell Apache to use Crowd to authenticate requests for a particular location, edit the Apache config file to add the following commands to a `<Location>` or `<Directory>` section.

```
Alias /crowd/ "/var/crowd/"
<Directory "/var/crowd/">
.
.
.
  AuthName crowd
  AuthType Basic

  PerlAuthenHandler Apache::CrowdAuth
  PerlSetVar CrowdAppName appname
  PerlSetVar CrowdAppPassword apppassword
  PerlSetVar CrowdSOAPURL http://localhost:8080/crowd/services/SecurityServer

  require valid-user
.
.
.
</Directory>
```

Command	Explanation
`AuthName crowd`	Defines the realm of the authentication. This information is typically provided to the user in the dialog box popped up by their browser
`AuthType Basic`	Tells apache to use basic authentication
`PerlAuthenHandler Apache::CrowdAuth`	Tells Apache to delegate authentication to the CrowdAuth module

`PerlSetVar CrowdAppName`	Set the Application Apache should authenticate as
`PerlSetVar CrowdAppPassword`	Set the password for the Application
`PerlSetVar CrowdSOAPURL`	The URL of the Crowd SOAP service
`require valid-user`	Tells Apache that clients must provide a valid username/password to access the location

Subversion Integration

If you are using Apache to manage access to a subversion repository (instructions), you can use the same configuration method to delegate user authentication to Crowd.

Example:

```
<Location /svn>

 # Uncomment this to enable the repository,
 DAV svn

 # Set this to the path to your repository
 SVNPath /var/lib/svn

 AuthName crowd
 AuthType Basic

 PerlAuthenHandler Apache::CrowdAuth
 PerlSetVar CrowdAppName subversion
 PerlSetVar CrowdAppPassword svn
 PerlSetVar CrowdSOAPURL http://localhost:8080/crowd/services/SecurityServer

 require valid-user

 # The following three lines allow anonymous read, but make
 # committers authenticate themselves.
 <LimitExcept GET PROPFIND OPTIONS REPORT>
 Require valid-user
 </LimitExcept>

</Location>
```

Note that Apache will have to be restarted before any changes to its config files will take effect.

Troubleshooting

- The `CrowdAuth` module logs detailed output if the Apache LogLevel parameter is set to `info` or `debug`. This can be useful in diagnosing problems (WARNING: passwords are logged in plaintext to the Apache log file when `LogLevel` is set to `debug`).

Apache Log Error Messages

`CrowdAppName or CrowdAppPassword is not defined`	One or both of the `CrowdAppName` or `CrowdAppPassword` parameters is missing from the Apache config file
`Failed to authenticate application`	The attempt to authenticate the application with

	crowd failed. Check the values of the `CrowdAppName` or `CrowdAppPassword` parameters
`Failed to authenticate principal`	Failed to authenticate a username/password pair provided by the client. This may just mean that the username or password supplied is incorrect. Note that `CrowdAuth` won't log successfull authentications unless the `LogLevel` is `info` or above.
`User token not found in SOAP response for user <user>`	Internal SOAP protocol error
`error 500...at CrowdAuth.pm..`	Indicates that Apache can't connect to the Crowd SOAP service
`error 404...at CrowdAuth.pm...`	Indicates that the URL used to connect to the Crowd SOAP service is incorrect. Check the value of the `CrowdSOAPURL` parameter
`failed to resolve handler `Apache::CrowdAuth': Can't locate Apache/CrowdAuth.pm ...`	The `CrowdAuth.pm` file isn't located on the Perl include path (or it is permissioned incorrectly)
`failed to resolve handler `Apache::CrowdAuth': Can't locate SOAP/Lite.pm...`	The `SOAP:Lite` module hasn't been installed

Installing Perl, **`mod_perl`** and Perl Modules on Windows

Setting up `CrowdAuth` on an Apache instance running on Windows requires that some things be done differently.

(The following instructions assume you are using ActivePerl as your Perl environment).

- If you don't already have a Perl interpreter installed, you'll need one. The following instructions assume an install of ActiveState's ActivePerl.
- Windows installations of Apache are less likely to come with `mod_perl` pre-installed. A Win32 version of `mod_perl` in PPM format is available here.
- The `.tar.gz` format used to distribute CrowdAuth (and other modules) is supported by most modern Windows archiving utilities (WinZip, for example).
- The `make` utility used to build the Perl modules is not part of a Windows. `nmake`, Microsoft's equivalent, is available (as a self-extracting archive) here.

Installing **`SOAP::Lite`** on Windows

Use the `cpan` shell

```
C:\ cpan
cpan> install SOAP::Lite
```

Installing **`Apache::CrowdAuth`** on Windows

```
Extract Apache-CrowdAuth-0.03.tar.gz using Winzip or equivalent...
cd Apache-CrowdAuth-0.03
perl Makefile.PL
nmake
nmake install
```

When editing the httpd.conf file and adding the mod_perl.so module to Apache, you may need to add the following line above the LoadModule line

```
LoadFile "C:/Perl/bin/perl58.dll"
LoadModule perl_module modules/mod_perl.so
```

This LoadFile line points to the perl58.dll in your Perl install directory.

Related Topics

- 3.1 Using the Application Browser
- 3.2 Adding an Application
  - 3.2.1 Integrating Crowd with Apache
  - 3.2.2 Integrating Crowd with Atlassian Bamboo
  - 3.2.3 Integrating Crowd with Atlassian Confluence
  - 3.2.4 Integrating Crowd with Atlassian JIRA
  - 3.2.5 Integrating Crowd with Cenqua FishEye
  - 3.2.6 Integrating Crowd with Jive Forums
    - 3.2.6.1 Jive SSO
  - 3.2.7 Integrating Crowd with a Custom Application
- 3.3 Mapping a Directory to an Application
  - 3.3.1 Specifying the Directory Order for an Application
- 3.4 Specifying which Groups can access an Application
- 3.5 Specifying an Application's Address or Hostname
- 3.6 Managing an Application's Session
- 3.7 Deleting or Deactivating an Application

Crowd Documentation

3.2.2 Integrating Crowd with Atlassian Bamboo

This page last changed on Mar 29, 2007 by rosie@atlassian.com.

Atlassian's Bamboo integration server can quickly be configured to use the atlassian-user libraries to link in single or multiple directory servers through Crowd.

To configure the atlassian-user framework, perform the following:

1. Copy the Crowd integration libraries and configuration files as described in the 3.2.7 Integrating Crowd with a Custom Application documentation.
2. Edit the `\bamboo\webapp\WEB-INF\classes\atlassian-user.xml` file to add the following repository:

```
<repository key="crowd" class="com.atlassian.crowd.integration.atlassianuser.CrowdRepository">
        <classes>
                <processor>com.atlassian.crowd.integration.atlassianuser.CrowdRepositoryProcessor</processor
                <userManager>com.atlassian.crowd.integration.atlassianuser.CrowdUserManager</userManager>
                <groupManager>com.atlassian.crowd.integration.atlassianuser.CrowdGroupManager</groupManager>
                <authenticator>com.atlassian.crowd.integration.atlassianuser.CrowdAuthenticator</authenticat
                <propertySetFactory>com.atlassian.crowd.integration.atlassianuser.CrowdPropertySetFactory</p
                <entityQueryParser>com.atlassian.crowd.integration.atlassianuser.CrowdEntityQueryParser</ent
        </classes>
</repository>
```

   You will need to comment out the Hibernate repository key

```
<!-- <hibernate name="Hibernate Repository" key="hibernateRepository" description="Hibernate
Repository"/> -->
```

3. This step is only necessary if you wish to enable single sign-on:

> Enabling Single Sign-On
> Single sign-on (SSO) is optional when integrating Bamboo and other Atlassian products. To use centralised authentication do not configure Seraph based authentication.

   Edit the `\bamboo\webapp\WEB-INF\classes\seraph-config.xml`, changing the `authenticator` node to read:

```
<authenticator class="com.atlassian.crowd.integration.seraph.BambooAuthenticator"/>
```

Bamboo's authentication and access request calls will now be performed using the atlassian-user Crowd plugin.

When utilising the atlassian-user and Crowd framework together with Bamboo, it is highly recommended that caching be enabled. Multiple redundant calls to the atlassian-user framework are made on any given request. These results can be stored locally between calls by enabling caching in the Crowd 'Options' menu. In doing so, Bamboo will obtain all necessary information for the period specified by the cache in minutes. If a security change or addition occurs in Crowd, these changes will not be visible in Confluence until the item cache expires.

Additional configuration steps:

- Create the 'bamboo' application via the Crowd Administration Console — for details on adding an application, see 3.2 Adding an Application. Make sure that you use the same password as configured in the `crowd.properties` file.
    - You will need to make sure you add the IP address of the client address, in this case Bamboo's IP address, to the list of authorised addresses — see 3.5 Specifying an Application's Address or Hostname.
- Create a group `bamboo-admin`, through the Crowd console or directly in your directory server.
    - You will need to assign the `bamboo-admin` group to the newly configured 'bamboo' application (see 3.4 Specifying which Groups can access an Application) or authentication attempts will fail.

Related Topics

- 3.1 Using the Application Browser
- 3.2 Adding an Application
    - 3.2.1 Integrating Crowd with Apache
    - 3.2.2 Integrating Crowd with Atlassian Bamboo
    - 3.2.3 Integrating Crowd with Atlassian Confluence
    - 3.2.4 Integrating Crowd with Atlassian JIRA
    - 3.2.5 Integrating Crowd with Cenqua FishEye
    - 3.2.6 Integrating Crowd with Jive Forums
        - 3.2.6.1 Jive SSO
    - 3.2.7 Integrating Crowd with a Custom Application
- 3.3 Mapping a Directory to an Application
    - 3.3.1 Specifying the Directory Order for an Application
- 3.4 Specifying which Groups can access an Application
- 3.5 Specifying an Application's Address or Hostname
- 3.6 Managing an Application's Session
- 3.7 Deleting or Deactivating an Application

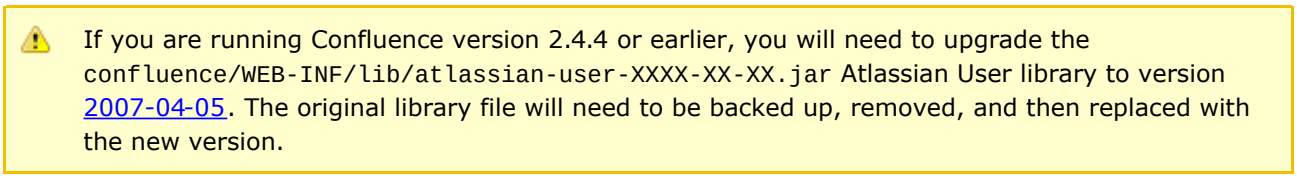
Crowd Documentation

3.2.3 Integrating Crowd with Atlassian Confluence

This page last changed on Apr 19, 2007 by rosie@atlassian.com.

Atlassian's popular Confluence wiki can quickly be configured to use the `atlassian-user` libraries to link in single or multiple directory servers through Crowd.

Currently Crowd supports centralised authentication and single sign-on for Confluence versions 2.3 and later.



Prerequisites

1. Download and install Crowd. Refer to the Crowd installation guide for detailed information on how to do this. We will refer to the Crowd root folder as `CROWD`.
2. Download and install Confluence (version 2.3 or later). Refer to the Confluence installation guide for detailed information on how to do this. We will refer to the Confluence root folder as `CONFLUENCE`. For the purposes of this document, we will assume that the standalone (ie. the easier) installation method of Confluence has been used. If you need to install Confluence as an EAR/WAR, simply explode the EAR/WAR and make the necessary changes as described below, and repackage the EAR/WAR.
3. After Confluence is set up, make sure Confluence is not running when you begin the integration process described below.

Step 1. Configuring Crowd to talk to Confluence

1.1 Prepare Crowd's Directories/Groups/Users for Confluence

The Confluence application will need to authenticate users against a directory configured in Crowd. You will need to set up a directory in Crowd for Confluence. For more information on how to do this, see 2.2 Adding a Directory. We will assume that the directory is called Confluence Directory for the rest of this document. It is possible to assign more than one directory for an application, but for the purposes of this example, we will use Confluence Directory to house Confluence users.

Confluence also requires particular groups to exist in the directory in order to authenticate users. You will need to create two groups in the Confluence Directory:

1. `confluence-users`
2. `confluence-administrators`

See the documentation on Creating Groups for more information on how to define these groups.

You also need to ensure that the Confluence Directory contains at least one user who is a member of

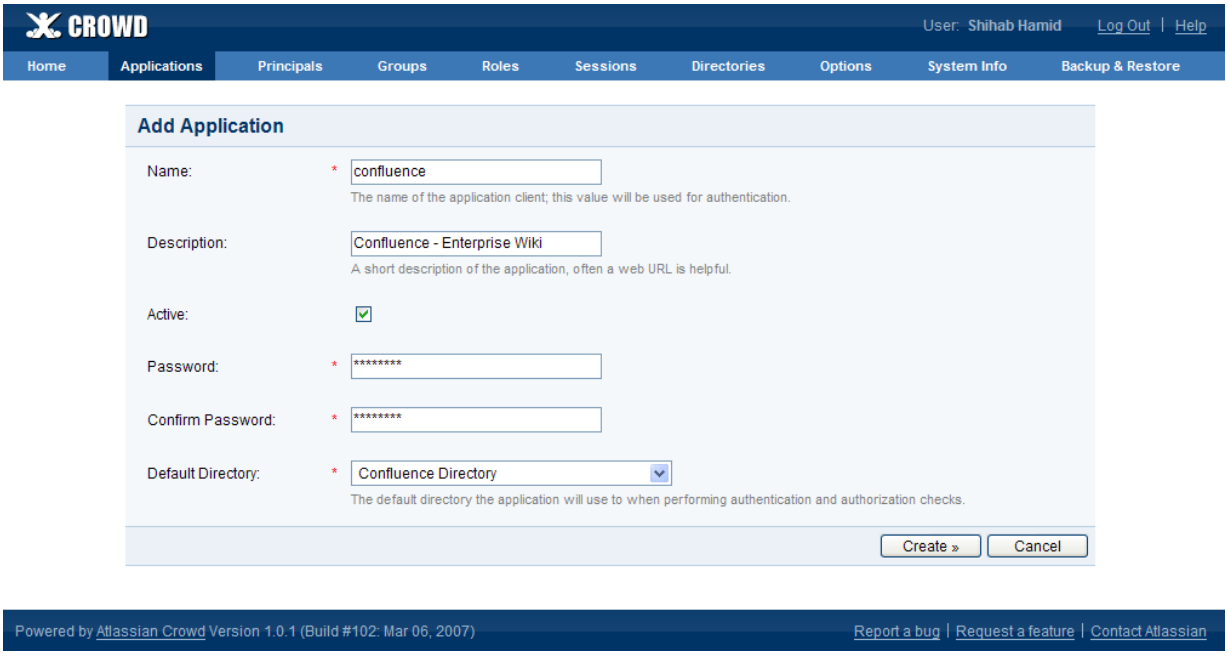
both groups. You can either:

- If you have an existing Confluence deployment and would like to import existing users (principals) and groups into Crowd, use the Confluence Importer tool by navigating to Principals > Import Users > Confluence. Select the Confluence Directory as the directory into which Confluence users will be imported. For details please see 2.4.1 Importing Users from Atlassian Confluence. If you are going to import users into Crowd, you need to do this now before you proceed any further.
  OR:
- If you don't wish to import your Confluence users, make sure you use Crowd to create at least one principal in the Confluence Directory and assign them to both the `confluence-users` and `confluence-administrators` group. If you don't wish to import your JIRA users, use the Crowd Administration Console to create the three groups, then create at least one principal in the JIRA Directory and add them to the three JIRA-specific groups (above). The Crowd documentation has more information on creating groups, creating principals and assigning principals to groups.

1.2 Define the Confluence Application in Crowd

Crowd needs to be aware that the Confluence application will be making authentication requests to Crowd. We need to add the Confluence application to Crowd and map it to the Confluence Directory:

1. Log in to the Crowd Administration Console and navigate to Applications > Add Application.
2. Fill out the form to add the Confluence application:



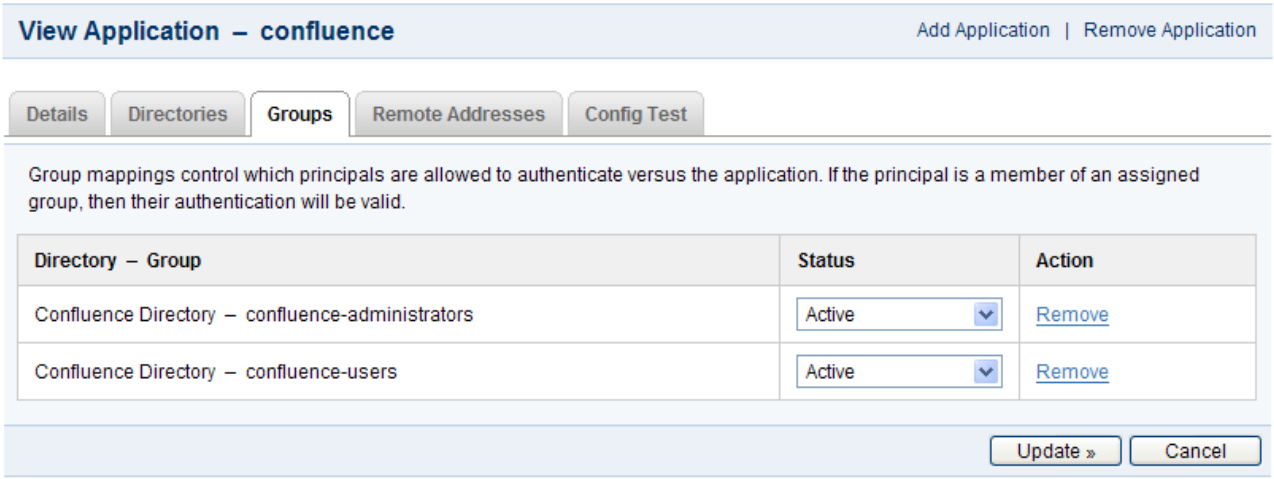
Attribute	Description
Name	The username which the application will use when it authenticates against the Crowd framework as a client. This value must be unique, i.e. it cannot be used by more than one application client.
Description	A short description of the application. Note: a web URL is often helpful.
Active	Only deselect this if you wish to prevent all users (from all directories) from accessing this

	application.
Password	The password which the application will use when it authenticates against the Crowd framework as a client.
Default Directory	A directory that contains relevant users. Note: additional directories can be added later.

The Name and Password values must match the application.name and application.password that you set in the `CONFLUENCE/confluence/WEB-INF/classes/crowd.properties` (see Step 2 below)

1.3 Specify which users can log in to Confluence

Now that Crowd is aware of the Confluence application, Crowd needs to know which users can authenticate (log in) to Confluence via Crowd. You can either allow entire directories to authenticate, or just particular groups within the directories. In our example, we will allow the `confluence-users` and `confluence-administrators` groups within the Confluence Directory to authenticate:



For details please see 3.4 Specifying which Groups can access an Application.

1.4 Specify the address from which Confluence can log in to Crowd

Please see 3.5 Specifying an Application's Address or Hostname. Please note:

- If Confluence is on a different host to Crowd
  If you are running the Confluence on a different host to Crowd, you will need to modify the permissible hosts via the Remote Addresses tab. This lists the hosts/IP addresses that are allowed to authenticate to Crowd. If Confluence is remote to Crowd, add the IP address of your Confluence server and ensure the "Status" field is set to "true". Remove the entry for `localhost`.

- If Confluence is on the same host as Crowd
  By default, when you add an application, `localhost` is a permissible foreign host. However, you will also need to manually add the IP address `127.0.0.1`, as incoming requests to Crowd from Confluence (both on the same, local, host) may be from the host `127.0.0.1` and not `localhost`. Crowd does not do a DNS lookup of the hostname; rather, it compares the values as is. Ensure the

"Status" field is set to "true".

Step 2. Configuring Confluence to talk to Crowd

2.1 Install the Crowd Client Libraries into Confluence

Confluence needs Crowd's client libraries in order to be able to delegate user authentication to the Crowd application. As stated earlier, we are going to be modifying the Confluence application by editing the standalone application, which is an exploded WAR stored in `CONFLUENCE/confluence`.

1. Copy the Crowd client libraries and configuration files to Confluence (this is described in the Client Configuration documentation). This is summarised below:

Copy From	Copy To
CROWD/client/*.jar	CONFLUENCE/confluence/WEB-INF/lib
CROWD/client/conf/crowd.properties	CONFLUENCE/confluence/WEB-INF/classes

   There is no need to copy across anything from `CROWD/client/lib`. All the required libraries from there already exist in Confluence versions 2.3 and later.

2. Edit `CONFLUENCE/confluence/WEB-INF/classes/crowd.properties`. Change the following properties:

Key	Value
application.name	confluence
application.password	set a password
crowd.server.url	http://localhost:8095/crowd/services/

   If your Crowd server's port is configured differently from the default (i.e. 8095), set it accordingly.
   The application.name and application.password must match the Name and Password that you specified when defining the application in Crowd (see Step 1 above).

2.2 Configure Confluence to use Crowd's Authenticator

Now that the Crowd client libraries exist, we need to configure Confluence to use them.

1. Edit the `CONFLUENCE/confluence/WEB-INF/classes/atlassian-user.xml` file to add the following repository:

```
<repository key="crowd" class="com.atlassian.crowd.integration.atlassianuser.CrowdRepository">
 <classes>
   <processor>com.atlassian.crowd.integration.atlassianuser.CrowdRepositoryProcessor</processor>
   <userManager>com.atlassian.crowd.integration.atlassianuser.CrowdUserManager</userManager>
   <groupManager>com.atlassian.crowd.integration.atlassianuser.CrowdGroupManager</groupManager>
   <authenticator>com.atlassian.crowd.integration.atlassianuser.CrowdAuthenticator</authenticator>
   <propertySetFactory>com.atlassian.crowd.integration.atlassianuser.CrowdPropertySetFactory</propertySetFa
   <entityQueryParser>com.atlassian.crowd.integration.atlassianuser.CrowdEntityQueryParser</entityQueryPars
 </classes>
</repository>
```

You will need to comment out the other repositories, eg:

```
<!-- <osuser key="osuserRepository" name="OSUser Repository"/> -->
<!-- <hibernate name="Hibernate Repository" key="hibernateRepository"  description="Hibernate
Repository" /> -->
```

This will tell Confluence to use Crowd's user repository for user management.

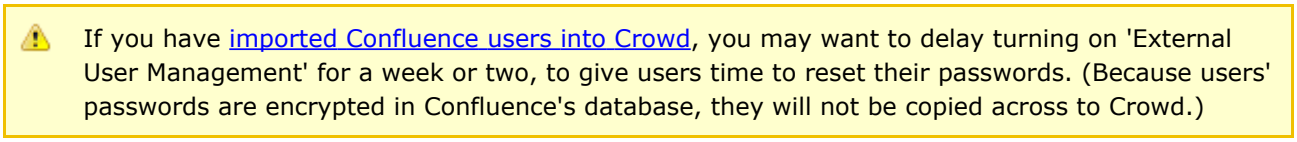
2. At this stage, Confluence is set up for centralised authentication. If you wish to enable single sign-on (SSO) to Confluence, edit `CONFLUENCE/confluence/webapp/WEB-INF/classes/seraph-config.xml`. Change the `authenticator` node to read:

```
<authenticator class="com.atlassian.crowd.integration.seraph.ConfluenceAuthenticator"/>
```

Confluence's authentication and access request calls will now be performed using Seraph.

2.3 Enable Confluence's 'External User Management'

Once the setup is complete, you may optionally wish to enable a Confluence feature known as 'External User Management', to prevent Confluence administrators from creating/modifying principals. For more information please see the Confluence documentation regarding External User Management.



See Crowd in Action

- You should now be able to login using principals belonging to the `confluence-users` group. Try adding a principal to the group using Crowd — you should be able to login to Confluence using this newly created principal. That's centralised authentication in action!
- If you have enabled SSO, you can try adding the Confluence Directory and `confluence-administrators` group to the crowd application (see 3.3 Mapping a Directory to an Application and 3.4 Specifying which Groups can access an Application). This will allow Confluence administrators to log in to the Crowd Administration Console. Try logging in to Crowd as a Confluence administrator, and then point your browser at Confluence. You should be logged in as the same principal in Confluence. That's single sign-on in action!

Related Topics

- 3.1 Using the Application Browser
- 3.2 Adding an Application
  - 3.2.1 Integrating Crowd with Apache
  - 3.2.2 Integrating Crowd with Atlassian Bamboo
  - 3.2.3 Integrating Crowd with Atlassian Confluence

    - 3.2.4 Integrating Crowd with Atlassian JIRA
    - 3.2.5 Integrating Crowd with Cenqua FishEye
    - 3.2.6 Integrating Crowd with Jive Forums
        - 3.2.6.1 Jive SSO
    - 3.2.7 Integrating Crowd with a Custom Application
- 3.3 Mapping a Directory to an Application
    - 3.3.1 Specifying the Directory Order for an Application
- 3.4 Specifying which Groups can access an Application
- 3.5 Specifying an Application's Address or Hostname
- 3.6 Managing an Application's Session
- 3.7 Deleting or Deactivating an Application

Crowd Documentation

3.2.4 Integrating Crowd with Atlassian JIRA

This page last changed on Apr 18, 2007 by justen.stepka@atlassian.com.

Atlassian's popular JIRA issue management system takes advantage of the OSUser framework and can quickly be configured to use OSUser to link in single or multiple directory servers through Crowd. Crowd provides integration libraries for the OpenSymphony OSUser module, which has a simple-to-use API for user-management that allows pluggable implementations. More about the OSUser API can be reviewed at http://www.opensymphony.com/osuser/.

Currently Crowd supports centralised authentication and single sign-on for JIRA versions 3.8 and later.

Prerequisites

1. Download and install Crowd. Refer to the Crowd installation guide for detailed information on how to do this. We will refer to the Crowd root folder as `CROWD`.
2. Download and install JIRA (version 3.7.4 or later). Refer to the JIRA installation guide for detailed information on how to do this. We will refer to the JIRA root folder as `JIRA`. For the purposes of this document, we will assume that the 'standalone' (i.e. the easier and recommended) installation method of JIRA has been used. If you need to install JIRA as an EAR/WAR, simply explode the EAR/WAR and make the necessary changes as described below, and repackage the EAR/WAR.
3. Make sure JIRA is not running when you begin the integration process described below.

Step 1. Configuring Crowd to talk to JIRA

1.1 Prepare Crowd's Directories/Groups/Users for JIRA

The JIRA application will need to locate users from a directory configured in Crowd. You will need to set up a directory in Crowd for JIRA. For information on how to do this, see 2.2 Adding a Directory. We will assume that the directory is called JIRA Directory for the rest of this document. It is possible to assign more than one directory for an application, but for the purposes of this example, we will use JIRA Directory to house JIRA users.

JIRA also requires particular groups to exist in the directory in order to authenticate users. You need to ensure that these three groups exist in the JIRA Directory:

1. `jira-users`
2. `jira-developers`
3. `jira-administrators`

You also need to ensure that the JIRA Directory contains at least one user who is a member of all three groups. You can either:

- If you have an existing JIRA deployment and would like to import existing groups and users (principals) into Crowd, use the JIRA Importer tool by navigating to Principals > Import Users > JIRA. Select the JIRA Directory as the directory into which JIRA users will be imported. For details please see 2.4.2 Importing Users from Atlassian JIRA. If you are going to import users into

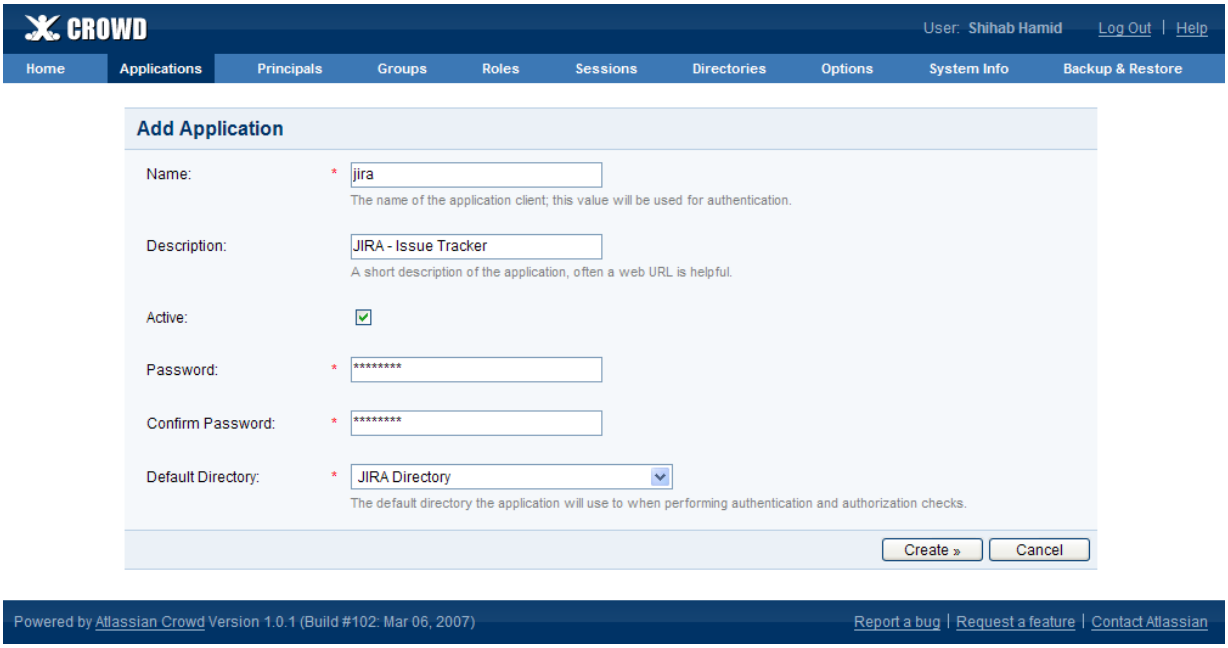
Crowd, you need to do this now before you proceed any further.
OR:

- If you don't wish to import your JIRA users, use the Crowd Administration Console to create the three groups, then create at least one principal in the JIRA Directory and add them to the three JIRA-specific groups (above). The Crowd documentation has more information on creating groups, creating principals and assigning principals to groups.

1.2 Define the JIRA Application in Crowd

Crowd needs to be aware that the JIRA application will be making authentication requests to Crowd. We need to add the JIRA application to Crowd and map it to the JIRA Directory.

1. Log in to the Crowd Administration Console and navigate to Applications > Add Application.
2. Fill out the form to add the JIRA application:

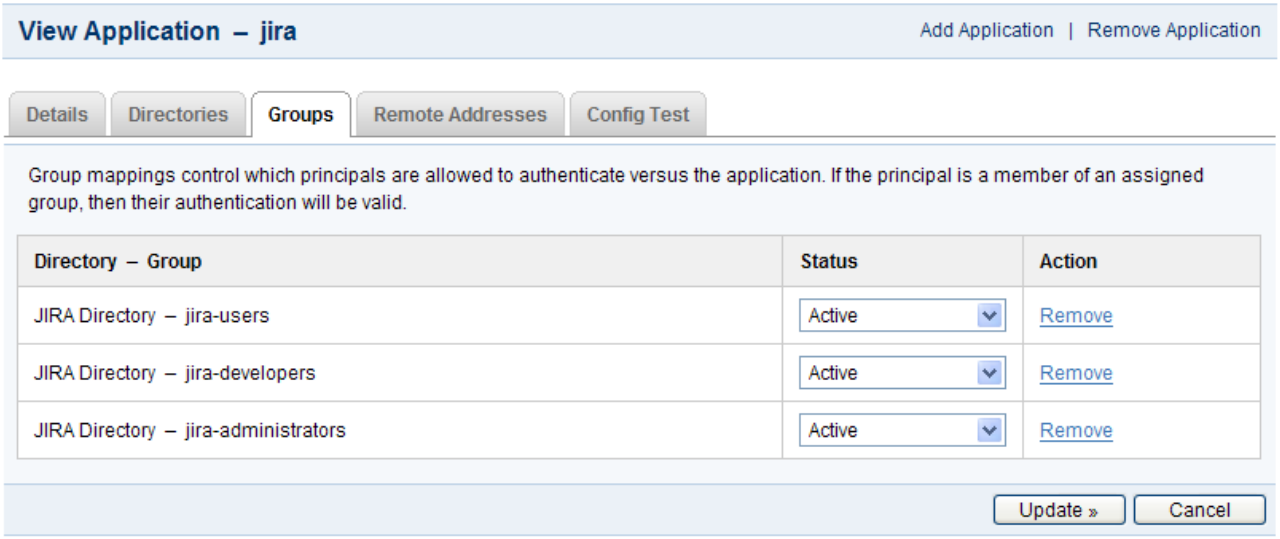


Attribute	Description
Name	The username which the application will use when it authenticates against the Crowd framework as a client. This value must be unique, i.e. it cannot be used by more than one application client.
Description	A short description of the application. Note: a web URL is often helpful.
Active	Only deselect this if you wish to prevent all users (from all directories) from accessing this application.
Password	The password which the application will use when it authenticates against the Crowd framework as a client.
Default Directory	A directory that contains relevant users. Note: additional directories can be added later.

The Name and Password values must match the application.name and application.password that you set in the `JIRA/atlassian-jira/WEB-INF/classes/crowd.properties` (see Step 2 below).

1.3 Specify which users can log in to JIRA

Now that Crowd is aware of the JIRA application, Crowd needs to know which directories or users can authenticate (log in) via Crowd. You can either allow entire directories to authenticate, or just particular groups within the directories. In our example, we will allow the `jira-users`, `jira-developers` and `jira-administrators` groups within the JIRA Directory to authenticate:



- Only principals who are members of the `jira-users` group will be able to authenticate against JIRA.
- Only principals who are members of the `jira-administrators` group will be able to use the JIRA administration console.

For details please see 3.4 Specifying which Groups can access an Application.

1.4 Specify the address from which JIRA can log in to Crowd

Please see 3.5 Specifying an Application's Address or Hostname. Please note:

- If JIRA is on a different host to Crowd
  If you are running the JIRA on a different host to Crowd, you will need to modify the permissible hosts via the Remote Addresses tab. This lists the hosts/IP addresses that are allowed to authenticate to Crowd. If JIRA is remote to Crowd, add the IP address of your JIRA server and ensure the "Status" field is set to "true". Remove the entry for `localhost`.

- If JIRA is on the same host as Crowd
  By default, when you add an application, `localhost` is a permissible foreign host. However, you will also need to manually add the IP address `127.0.0.1`, as incoming requests to Crowd from JIRA (both on the same, local, host) may be from the host `127.0.0.1` and not `localhost`. Crowd does not do a DNS lookup of the hostname, rather, it compares the values as is. Ensure the "Status" field is set to "true".

Step 2. Configuring JIRA to talk to Crowd

2.1 Install the Crowd Client Libraries into JIRA

JIRA needs Crowd's client libraries in order to be able to delegate user authentication to the Crowd application. As stated earlier, we are going to be modifying the JIRA application by editing the standalone application, which is an exploded WAR stored in `JIRA/atlassian-jira`.

1. Copy the Crowd client libraries and configuration files to JIRA (this is described in the Client Configuration documentation). This is summarised below:

Copy From	Copy To
CROWD/client/*.jar	JIRA/atlassian-jira/WEB-INF/lib
CROWD/client/conf/crowd.properties	JIRA/atlassian-jira/WEB-INF/classes

   There is no need to copy across anything from `CROWD/client/lib`. All the required libraries from there already exist in JIRA versions 3.7.4 and later.

2. Edit `JIRA/atlassian-jira/WEB-INF/classes/crowd.properties`. Change the following properties:

Key	Value
application.name	jira
application.password	set a password
crowd.server.url	http://localhost:8095/crowd/services/

   If your Crowd server's port is configured differently from the default (i.e. 8095), set it accordingly.
   The application.name and application.password must match the Name and Password that you specified when you defined the application in Crowd (see Step 1 above).

2.2 Configure JIRA to use Crowd's Authenticator

Now that the Crowd client libraries exist, we need to configure JIRA to use them.

1. Edit the JIRA config file `JIRA/atlassian-jira/WEB-INF/classes/osuser.xml`. Comment out any existing authentication providers and uncomment/insert the Crowd providers:

```
<!-- This is where JIRA's credentials checking can be configured.  For instance, see
http://www.atlassian.com/software/jira/docs/latest/ldap.html -->
<opensymphony-user>
  <authenticator class="com.opensymphony.user.authenticator.SmartAuthenticator" />

<!-- You will need to uncomment the Crowd providers below to enable Crowd integration -->
  <provider class="com.atlassian.crowd.integration.osuser.CrowdCredentialsProvider"/>
  <provider class="com.atlassian.crowd.integration.osuser.CrowdAccessProvider"/>
  <provider class="com.atlassian.crowd.integration.osuser.DelegatingProfileProvider">
    <property
name="provider-1">com.atlassian.crowd.integration.osuser.CrowdProfileProvider</property>
    <property
name="provider-2">com.atlassian.jira.user.ExternalEntityJiraProfileProvider</property>
    <property name="provider-2-exclusive-access">true</property>
```

```
 </provider>

<!-- CROWD:START  - The providers below here will need to be commented out for Crowd
integration -->
<!--
 <provider class="com.atlassian.core.ofbiz.osuser.CoreOFBizCredentialsProvider">
   <property name="exclusive-access">true</property>
 </provider>

 <provider class="com.opensymphony.user.provider.ofbiz.OFBizProfileProvider">
   <property name="exclusive-access">true</property>
 </provider>

 <provider class="com.opensymphony.user.provider.ofbiz.OFBizAccessProvider">
   <property name="exclusive-access">true</property>
 </provider>
-->
<!-- CROWD:END -->

</opensymphony-user>
```

2. View `JIRA/atlassian-jira/WEB-INF/classes/propertyset.xml`. If an entry doesn't exists for the CrowdPropertySet, to add the following <propertyset> at the end of the file as the last <propertyset>:

```
<propertyset name="crowd" class="com.atlassian.crowd.integration.osuser.CrowdPropertySet"/>
```

3. At this stage, JIRA is set up for centralised authentication. If you wish to enable single sign-on (SSO) to JIRA, edit `JIRA/atlassian-jira/WEB-INF/classes/seraph-config.xml`. Change the `authenticator` node to read:

```
<authenticator class="com.atlassian.crowd.integration.seraph.JIRAAuthenticator"/>
```

   JIRA's authentication and access request calls will now be performed using Seraph. Now when authentication or access request calls are performed versus the OSUser framework, the JIRA stack will call the Crowd providers and propertyset implementations.

2.3 Enable JIRA's 'External User Management'

Once the setup is complete, go to the JIRA Administration Console. In the `General Configuration` section, turn on 'External User Management' and 'External Password Management' (see the JIRA Administrator's Guide for details). This means that the following functions can no longer be performed from within the JIRA administration interface: adding users, adding groups, editing users, editing groups.

See Crowd in Action

- You should now be able to login using principals belonging to the `jira-users` group. Try adding a principal to the group using Crowd — you should be able to login to JIRA using this newly created principal. That's centralised authentication in action!
- If you have enabled SSO, you can try adding the JIRA Directory and `jira-administrators` group to the crowd application (see 3.3 Mapping a Directory to an Application and 3.4 Specifying which Groups can access an Application). This will allow JIRA administrators to log in to the Crowd Administration Console. Try logging in to Crowd as a JIRA administrator, and then point your browser at JIRA. You should be logged in as the same principal in JIRA. That's single sign-on in action!

Known Limitations

- JIRA currently does not have logic to handle the removal of a user when external user management is enabled. If the user is removed in Crowd, JIRA will throw a `DataAccessException`. The current workaround for this is to deactivate the principal's account (by removing them from the `jira-users` group). This issue can be tracked here: http://jira.atlassian.com/browse/CWD-202

Related Topics

- 3.1 Using the Application Browser
- 3.2 Adding an Application
    - 3.2.1 Integrating Crowd with Apache
    - 3.2.2 Integrating Crowd with Atlassian Bamboo
    - 3.2.3 Integrating Crowd with Atlassian Confluence
    - 3.2.4 Integrating Crowd with Atlassian JIRA
    - 3.2.5 Integrating Crowd with Cenqua FishEye
    - 3.2.6 Integrating Crowd with Jive Forums
        - 3.2.6.1 Jive SSO
    - 3.2.7 Integrating Crowd with a Custom Application
- 3.3 Mapping a Directory to an Application
    - 3.3.1 Specifying the Directory Order for an Application
- 3.4 Specifying which Groups can access an Application
- 3.5 Specifying an Application's Address or Hostname
- 3.6 Managing an Application's Session
- 3.7 Deleting or Deactivating an Application

Crowd Documentation

3.2.5 Integrating Crowd with Cenqua FishEye

This page last changed on Mar 29, 2007 by rosie@atlassian.com.

FishEye allows you to use Crowd to provide external authentication and authorisation.

Step 1. Configuring Crowd to talk to FishEye

Please follow the instructions in 3.2 Adding an Application.

Step 2. Configuring FishEye to talk to Crowd

> Before you begin
> For any usernames that are already configured through the Fisheye Administration console, you will need to change the account type from 'built-in' to 'custom'. This is required for the new permissioning through Crowd to work properly.

For details please see the Fisheye documentation:
http://www.cenqua.com/fisheye/doc/latest/admin/customauth.html

2.1 Install the Crowd Client Libraries into FishEye

Copy the Crowd integration libraries and configuration files as described in the 3.2.7 Integrating Crowd with a Custom Application documentation.

2.2 Configure FishEye to use Crowd's Authenticator

To configure Fisheye you will need to specify the following Custom Authenticator in the `Users/Security` section of the administration console.

```
com.atlassian.crowd.integration.fisheye.FisheyeAuthenticator
```

It is also important to note that Fisheye requires you to pass in the configuration attributes for Crowd, by specifying your configuration data through the properties editor:

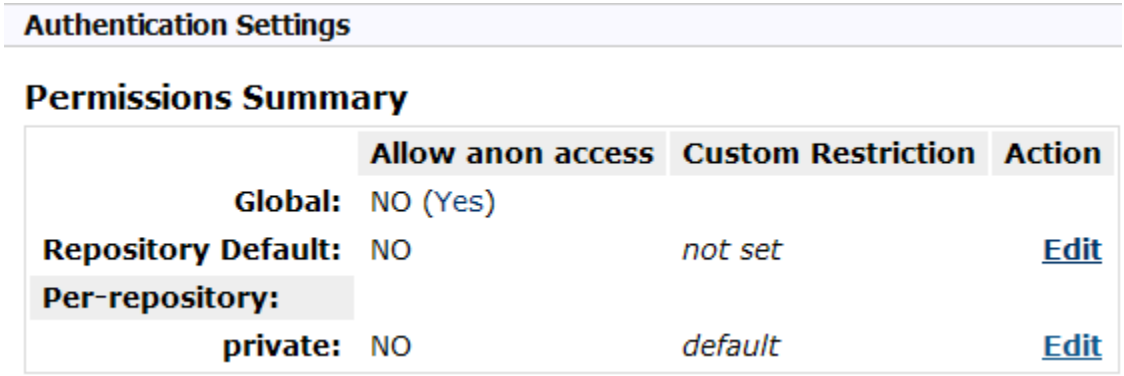
```
application.name                        fisheye
application.password                    password
application.login.url                   http://localhost:8080/
crowd.server.url                        http://localhost:8095/crowd/services/
session.isauthenticated                 session.isauthenticated
session.tokenkey                        session.tokenkey
session.validationinterval              0
session.lastvalidation                  session.lastvalidation
```

2.3 Configure groups for FishEye source-repositories (if required)

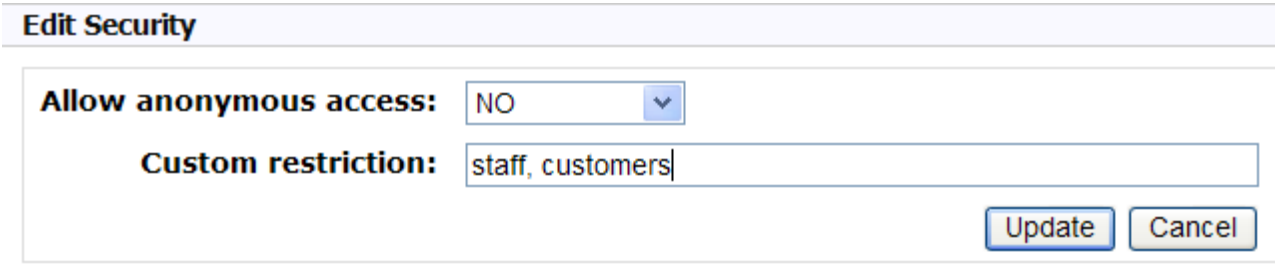
If you are using any FishEye groups to control access to particular source-repositories, you will need to create the groups in Crowd and then configure FishEye as follows:

1. In the FishEye Administration menu, select 'Global Settings', then 'Users/Security'.
2. This will display the 'Authentication Settings' screen. In the 'Permissions Summary' section, edit the 'Per-repository' field and enter the group names (separated by commas) in the 'Custom restriction' field.

Screenshot 1: 'Authentication Settings'



Screenshot 2: 'Custom Restriction'



Related Topics

- 3.1 Using the Application Browser
- 3.2 Adding an Application
  - 3.2.1 Integrating Crowd with Apache
  - 3.2.2 Integrating Crowd with Atlassian Bamboo
  - 3.2.3 Integrating Crowd with Atlassian Confluence
  - 3.2.4 Integrating Crowd with Atlassian JIRA
  - 3.2.5 Integrating Crowd with Cenqua FishEye
  - 3.2.6 Integrating Crowd with Jive Forums
    - 3.2.6.1 Jive SSO
  - 3.2.7 Integrating Crowd with a Custom Application
- 3.3 Mapping a Directory to an Application
  - 3.3.1 Specifying the Directory Order for an Application
- 3.4 Specifying which Groups can access an Application
- 3.5 Specifying an Application's Address or Hostname
- 3.6 Managing an Application's Session
- 3.7 Deleting or Deactivating an Application

Crowd Documentation

3.2.6 Integrating Crowd with Jive Forums

This page last changed on May 16, 2007 by justin.

Jive Forums offers you the ability to specify an implementation to provide authentication and authorisation external to the application. This document outlines how to integrate Crowd's authenticator with Jive Forums.

Currently Crowd provides centralised authentication and single sign-on (SSO) for Jive Forums version 5.0.x. For information regarding compatibility with version 5.5, please see CWD-245.

Prerequisites

1. Download and configure Crowd. Refer to the Crowd installation guide for detailed information on how to do this. We will refer to the Crowd root folder as `CROWD`.
2. Install/configure Jive Forums. Refer to the relevant Jive Forums documentation for information regarding this installation process. The documentation is usually supplied with the software distribution. Do not attempt to use Crowd as the authentication system during the installation process (use the default authentication system for the installation process).

Step 1. Tell Crowd about Jive Forums

1.1 Prepare Crowd's Directory/Users for Jive Forums

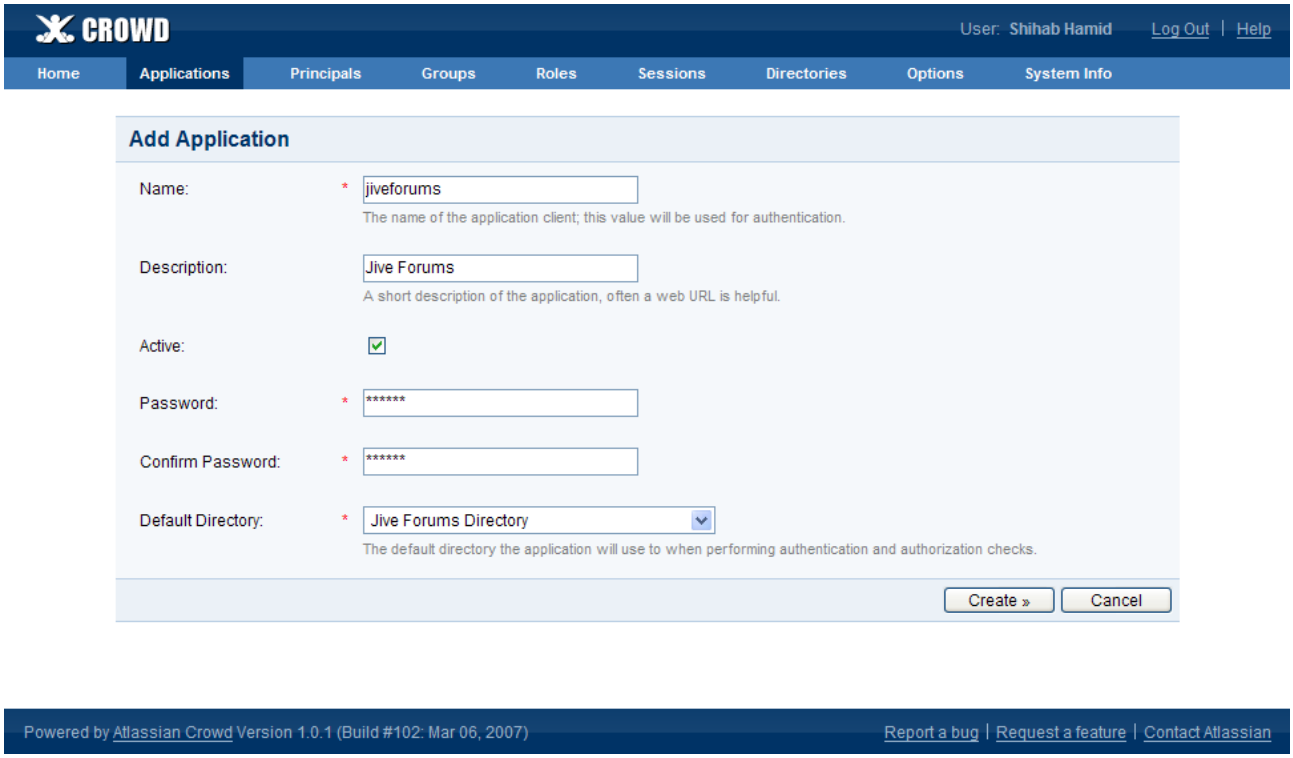
The Jive Forums application will need to locate users from a directory configured in Crowd. You will need to set up a directory in Crowd for Jive. For more information on how to do this, see 2.2 Adding a Directory. We will assume that the directory is called Jive Forum Directory for the rest of this document. It is possible to assign more than one directory for an application, but for the purposes of this example, we will use Jive Forum Directory to house Jive Forum users.

If you have an existing Jive Forums deployment and would like to import existing users (principals) into Crowd, use the Jive Importer tool by navigating Principals > Import Users > JIVE. Select the Jive Forum Directory as the directory into which Jive Forum users will be imported. For details please see 2.4.3 Importing Users from Jive Forums. If you are going to import users into Crowd, you need to do this now before you proceed any further.

1.2 Define the Jive Forums Application in Crowd

Crowd needs to be aware that the Confluence application will be making authentication requests to Crowd. We need to add the Jive Forums application to Crowd and map it to the Jive Forums Directory:

1. Log in to the Crowd Administration Console and navigate to Applications > Add Application.
2. Fill out the form to add the Jive Forums application:

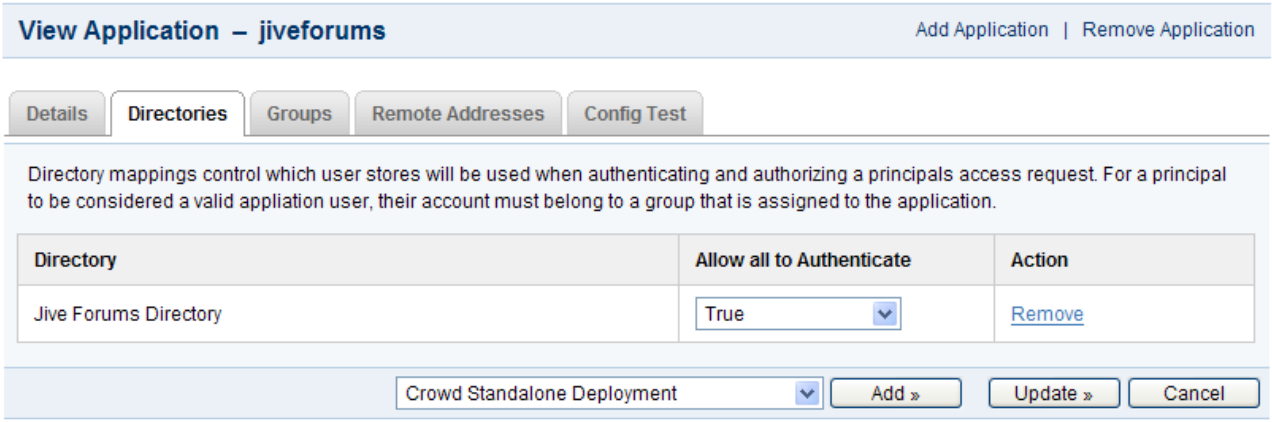


Attribute	Description
Name	The username which the application will use when it authenticates against the Crowd framework as a client. This value must be unique, i.e. it cannot be used by more than one application client.
Description	A short description of the application. Note: a web URL is often helpful.
Active	Only deselect this if you wish to prevent all users (from all directories) from accessing this application.
Password	The password which the application will use when it authenticates against the Crowd framework as a client.
Default Directory	A directory that contains relevant users. Note: additional directories can be added later.

The Name and Password values must match those set in the `JIVEFORUMS/WEB-INF/classes/crowd.properties`(see Step 2 below).

1.3 Specify which users can log in to Jive Forums

Now that Crowd is aware of the Jive Forums application, Crowd needs to know which directories or users can authenticate (log in) via Crowd. You can either configure entire directories to authenticate or allow particular groups. In our example, we can simply allow the entire directory to authenticate:



Alternatively, we can use the Groups tab to restrict the application to only authenticate particular groups of users. For details please see 3.4 Specifying which Groups can access an Application.

1.4 Specify the address from which Jive Forums can log in to Crowd

Please see 3.5 Specifying an Application's Address or Hostname. Please note:

- Jive Forums is on a different host to Crowd
  If you are running Jive Forums on a different host to Crowd, you will need to modify the permissible hosts via the Remote Addresses tab. This lists the hosts/IP addresses that are allowed to authenticate to Crowd. If Jive Forums is remote to Crowd, add the IP address of your Jive Forums server and ensure the "Status" field is set to "true". Remove the entry for `localhost`.

- Jive Forums is on the same host as Crowd
  By default, when you add an application, `localhost` is a permissible foreign host. However, you will also need to manually add the IP address `127.0.0.1`, as incoming requests to Crowd from Jive (both on the same, local, host) may be from the host `127.0.0.1` and not `localhost`. Crowd does not do a DNS lookup of the hostname, rather, it compares the values as is. Ensure the "Status" field is set to "true".

Step 2. Tell Jive Forums about Crowd

2.1 Install the Crowd Client Libraries into the Jive Forums WebApp

Jive Forums may be deployed on an application server as a single WAR file or a an exploded WAR folder. For the rest of the installation process, we will assume that Jive Forums has been set up as an exploded war file. If you need Jive Forums to be installed as a single WAR file, simply expand the WAR to a directory, make the changes as described below, and zip up the directory to form the WAR file. We will refer to the root folder of the Jive Forums web-app as `JIVEFORUMS`.

1. Copy the Crowd integration libraries and configuration files (this is described in the Client Configuration documentation). This is summarised below:

Copy From	Copy To

CROWD/client/crowd-core-*.jar	JIVEFORUMS/WEB-INF/lib
CROWD/client/lib/log4j-1.2.8.jar	JIVEFORUMS/WEB-INF/lib
CROWD/client/lib/ehcache-1.2.3.jar	JIVEFORUMS/WEB-INF/lib
CROWD/client/conf/crowd.properties	JIVEFORUMS/WEB-INF/classes
CROWD/client/conf/crowd-ehcache.xml	JIVEFORUMS/WEB-INF/classes

1. Examine the `JIVEFORUMS/WEB-INF/lib` folder and delete any duplicate JARs. Duplicate JARs represent common libraries used by both the Crowd client and Jive Forums.
2. Edit `JIVEFORUMS/WEB-INF/classes/crowd.properties`. Change the following properties:

Key	Value
application.name	jiveforums
application.password	set a password

 The name and password values must match those set when defining the application in Crowd (see Step 1 above).

2.2 Configure Jive Forums to use Crowd's Authenticator

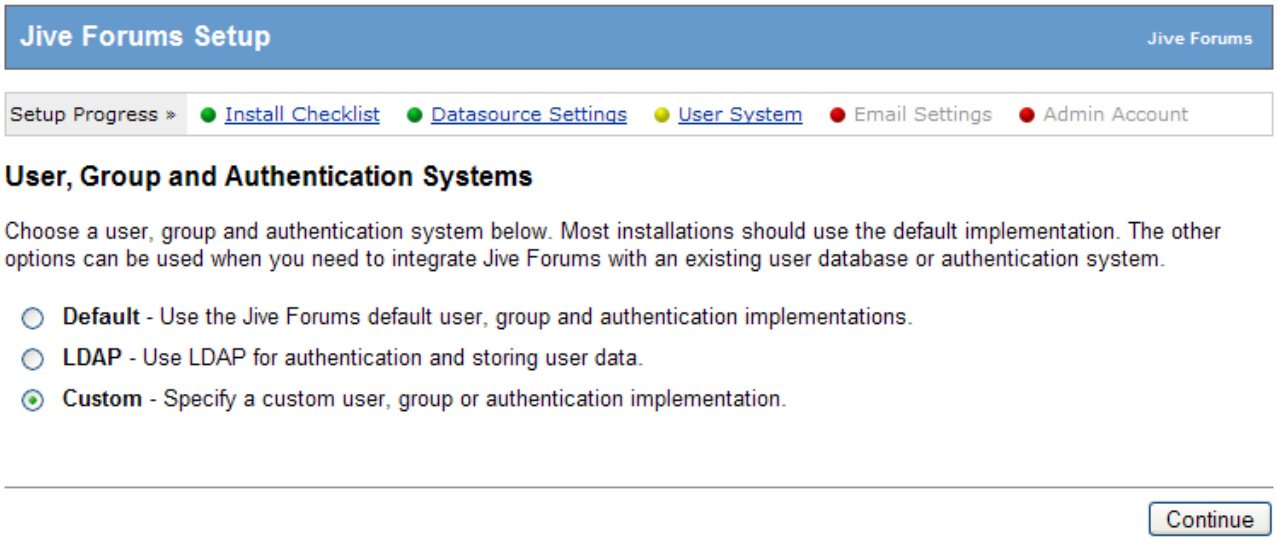
Crowd is now set up to provide authentication services to Jive. Now Jive needs to be set up to use Crowd's authenticator. There are a few ways of doing this; the most user-friendly method is outlined below:

1. In your `jiveHome` directory, edit a file named `jive_startup.xml`. Modify the `<setup>` node to be `false`:

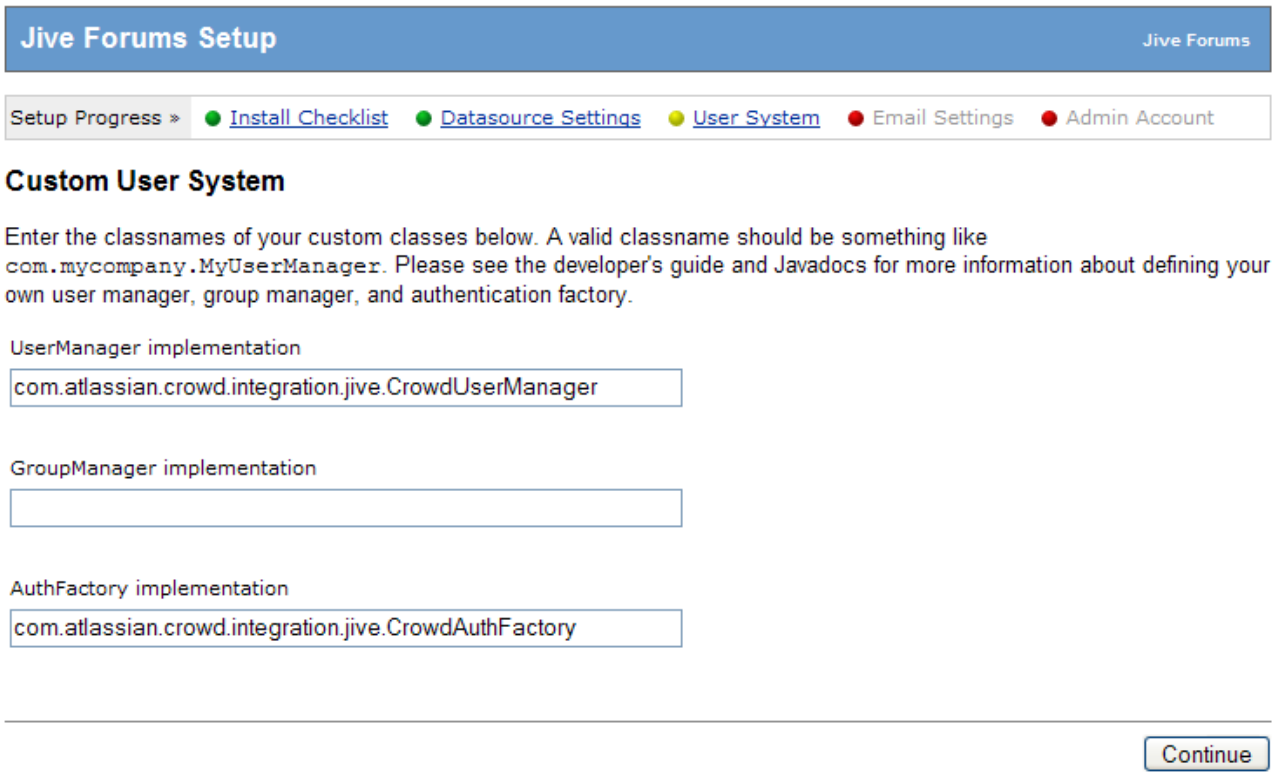
```
<jive>
 <!-- When setup is false, you can access the setup tool. -->
 <setup>false</setup>
 ...
 <!-- Allow SSO login for admins -->
 <admin>
   <tryAlternativeLogin>true</tryAlternativeLogin>
 </admin>
</jive>
```

   As the XML comment states, this lets us re-run Jive's setup.

1. Restart Jive Forums so that it picks up the changes.
2. View the Jive Forums site with a web browser (usually under the `/jiveforums` context-root. Jive will run the "Jive Forums Setup".
3. In the Install Checklist screen, click continue to navigate through the setup process.
4. In the Datasource Settings screen, re-enter your database configuration details and click continue.
5. In the User System screen, select Custom authentication system and click Continue:



1. You should be at the Custom User System screen. Enter the following details which specify Crowd as the custom authenticator:



UserManager implementation:

```
com.atlassian.crowd.integration.jive.CrowdUserManager
```

GroupManager implementation:

Do not specifiy an implementation.

AuthFactory implementation:

```
com.atlassian.crowd.integration.jive.CrowdAuthFactory
```

Click continue.

If you have any errors at this stage, it is very likely that there is a classpath issue (eg. the Crowd client libraries aren't being properly loaded by Jive). Please read the documentation regarding Crowd Client Libraries for help identifying the problem.

1. In the Email Settings screen, re-enter your email configuration details and click continue.
2. In the Admin Account Setup screen, do not enter any details. Click Skip this step.

Warning
The default administrator for Jive Forums is the user `admin`. This user will need to exist in your mapped directory (i.e. the Jive Forums Directory). Without this user, you will not be able to access the administration console of Jive Forums.

1. Bounce the server and test that Crowd is authenticating users for Jive. You can do this by creating users (principals) via the Crowd Administration Console and verifying that they are able to log in to Jive Forums.

Jive Forums Documentation
For further information regarding Jive Forums Authentication Integration, check out the Jive Forums Documentation at http://www.jivesoftware.com/builds/docs/latest/documentation/developer-guide.html#userintegration

Check out the Jive SSO page for more details on Jive SSO Integration and corresponding use cases.

Related Topics

- 3.1 Using the Application Browser
- 3.2 Adding an Application
  - 3.2.1 Integrating Crowd with Apache
  - 3.2.2 Integrating Crowd with Atlassian Bamboo
  - 3.2.3 Integrating Crowd with Atlassian Confluence
  - 3.2.4 Integrating Crowd with Atlassian JIRA
  - 3.2.5 Integrating Crowd with Cenqua FishEye
  - 3.2.6 Integrating Crowd with Jive Forums
    - 3.2.6.1 Jive SSO
  - 3.2.7 Integrating Crowd with a Custom Application
- 3.3 Mapping a Directory to an Application
  - 3.3.1 Specifying the Directory Order for an Application
- 3.4 Specifying which Groups can access an Application
- 3.5 Specifying an Application's Address or Hostname
- 3.6 Managing an Application's Session
- 3.7 Deleting or Deactivating an Application

Crowd Documentation

3.2.6.1 Jive SSO

This page last changed on Mar 29, 2007 by rosie@atlassian.com.

This page details the nuts and bolts of Jive SSO. If you are having issues with Jive SSO, this page should be able to give you a better idea of what's going on behind the scenes and help you diagnose any common problems.

For Crowd-Jive integration, the incoming request must:

1. be authenticated with Crowd (have a Crowd SSO token in session or as a cookie)
2. be authenticated with Jive (have a CrowdAuthToken stored in HttpSession for Jive)

To authenticate with Crowd: simply log in to Crowd via any Crowd-SSO enabled application. This includes Jive's login page.

To authenticate with Jive: you need to be authenticated with Crowd as a principal "allowed to be authenticated" by Jive. This means, the principal must belong to a group or directory which Jive is authorised to authenticate. This user also needs to NOT be on any user/IP ban lists within the Jive application. The Crowd integration will honour the ban list. See note below.

Enumeration of Use Cases

User views Jive Forums and:

1. request is not authenticated with Crowd -> appears as guest user in Jive.
2. request is authenticated with Crowd, but principal is not in directory/group allowed to authenticate with Jive -> appears as guest user in Jive.
3. request is authenticated with Crowd, principal allowed to authenticate with Jive, principal not on any ban list -> appears as logged-in user in Jive.
4. authenticated Jive clicks logout from Jive -> user is logged out of Jive and Crowd.
5. authenticated Jive user logs out of Crowd using another SSO app -> user eventually times out of Jive.
6. request is authenticated with Crowd, principal banned from logging into Crowd -> user appears as guest in Jive.
7. admin authenticated with Crowd and attempts to access Jive admin console -> admin appears logged in to Jive admin console.
8. authenticated Jive admin attempts to logout from Jive's admin console -> admin is still logged in (support issue filed with Jive Forums)
9. authenticated Jive admin attempts to logout from Jive Forums -> admin is logged out of Jive and Crowd.
10. request is authenticated with Crowd but user is banned from Jive Forums -> user is still authenticated with Crowd, but not allowed to login Jive Forums

Special Cases

- It is known that the "remember me" functionality of Jive will cease to function. This has been intentionally disabled. Jive's "remember me" functionality will need to be replaced by a more general "remember me" from within Crowd. Once this is implemented in Crowd, the Jive integration libraries

can utilise Crowd's "remember me", so that "remember me" is centralised.

- It is recommended that admins do not use ban lists, rather, manage access control based on Crowd's groups. So it's best to disable Ban Users from within Ban Settings inside the Jive admin console. There is nothing wrong with using ban lists, as they will be honoured by the Crowd-Jive integration libraries, they will make it hard for a banned user to switch to a non-banned user (as the only way a banned Jive user, authenticated with Crowd for Jive, will be able to switch to a different principal that Jive will pick up, is when the Jive's Crowd authentication cache clears, so that Jive recognises a new principal is signing in). This is because there is no way to logout a banned user from Jive (as they will always appear to be "guest"). So basically, if you have users with multiple identities, if one is banned and attempts to login, the user will have to wait until the client cache is cleared before he/she can login with a different identity. Note: it's easy for non-banned users to switch identities as the client authentication cache is cleared when they click "logout" from within Jive.

Related Topics

- 3.1 Using the Application Browser
- 3.2 Adding an Application
  - 3.2.1 Integrating Crowd with Apache
  - 3.2.2 Integrating Crowd with Atlassian Bamboo
  - 3.2.3 Integrating Crowd with Atlassian Confluence
  - 3.2.4 Integrating Crowd with Atlassian JIRA
  - 3.2.5 Integrating Crowd with Cenqua FishEye
  - 3.2.6 Integrating Crowd with Jive Forums
    - 3.2.6.1 Jive SSO
  - 3.2.7 Integrating Crowd with a Custom Application
- 3.3 Mapping a Directory to an Application
  - 3.3.1 Specifying the Directory Order for an Application
- 3.4 Specifying which Groups can access an Application
- 3.5 Specifying an Application's Address or Hostname
- 3.6 Managing an Application's Session
- 3.7 Deleting or Deactivating an Application

Crowd Documentation

3.2.7 Integrating Crowd with a Custom Application

This page last changed on May 14, 2007 by rosie@atlassian.com.

Crowd ships with out-of-the-box support for a number of applications. You can also integrate Crowd with other applications as follows:

Step 1. Configuring Crowd to talk to your Application

Please see 3.2 Adding an Application.

Step 2. Configuring your Application to talk to Crowd

2.1 Developing a Crowd Client

If your application is not listed in 1.1.1 Supported Applications and Directories then you will need to create your own Crowd Client for your application, using the Crowd SOAP API.
For assistance, please see Creating a Crowd Client for your Custom Application.

2.2 Configuring your Application

The integration libraries and configuration files are included in the Crowd download, in the `client` folder. You will find the Crowd integration library, and the client libraries on which the framework depends, in the `lib` folder. An example client properties file `crowd.properties` is located in the `conf` folder.

To configure your application, perform the following:

1. Copy the Crowd Client and supporting libraries to your application classpath, typically `WEB-INF/lib`.
   - These files will be in the `client` folder similar to `crowd-core-0.4.1.jar` and all supporting jars in the `client/lib` folder.
2. Copy the client properties file `crowd.properties` to your application's deployment directory, typically `WEB-INF/classes`.
3. Edit the `crowd.properties` file to reflect the values of your deployment parameters. The `crowd.properties` attributes are as follows:

Attribute	Description
application.name	The name that the application will use when authenticating with the Crowd server. This needs to match the name you specified in 3.2 Adding an Application.
application.password	The password that the application will use when authenticating with the Crowd server. This needs to match the password you specified in 3.2 Adding an Application.

application.login.url	The URL to which to redirect the principal should their authentication token expire or be invalid due to security restrictions.
crowd.server.url	The URL to use when connecting with the integration libraries to communicate with the Crowd server.
session.isauthenticated	The session key to use when storing a `Boolean` value indicating whether the principal is authenticated or not.
session.tokenkey	The session key to use when storing a `String` value of the principal's authentication token.
session.validationinterval	The session key to use when storing an `Integer` value of the number of minutes between authentication validation. If this value is set to `0`, each HTTP request will be authenticated.
session.lastvalidation	The session key to use when storing a `Date` value of the principal's last authentication.

Related Topics

- 3.1 Using the Application Browser
- 3.2 Adding an Application
  - 3.2.1 Integrating Crowd with Apache
  - 3.2.2 Integrating Crowd with Atlassian Bamboo
  - 3.2.3 Integrating Crowd with Atlassian Confluence
  - 3.2.4 Integrating Crowd with Atlassian JIRA
  - 3.2.5 Integrating Crowd with Cenqua FishEye
  - 3.2.6 Integrating Crowd with Jive Forums
    - 3.2.6.1 Jive SSO
  - 3.2.7 Integrating Crowd with a Custom Application
- 3.3 Mapping a Directory to an Application
  - 3.3.1 Specifying the Directory Order for an Application
- 3.4 Specifying which Groups can access an Application
- 3.5 Specifying an Application's Address or Hostname
- 3.6 Managing an Application's Session
- 3.7 Deleting or Deactivating an Application

Crowd Documentation

3.3 Mapping a Directory to an Application

This page last changed on Apr 23, 2007 by rosie@atlassian.com.

Mapping a directory to an application defines the user-base for an application. Sometimes known as 'application provisioning', directory mappings determine which user stores will be used when authenticating and authorising a user's access request. (Note: users are known in Crowd as principals).

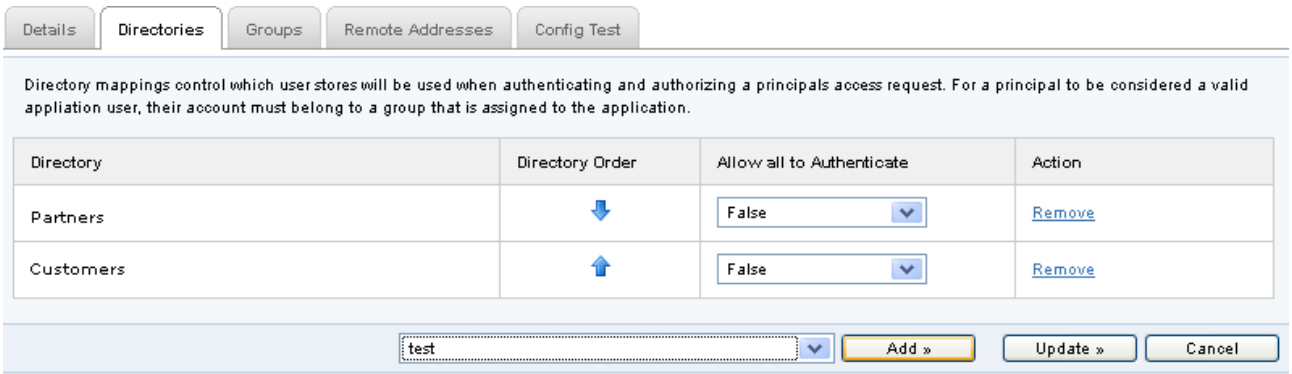
When you defined an application, you chose a default directory for that application to use. Crowd also enables you to map multiple directories to each application. This allows each of your applications to view multiple user directories as a single repository.

To map a directory to an application,

1. Login to the Crowd Administration Console.
2. Click the 'Applications link in the top navigation bar.
3. This will display the Application Browser. Click the 'View' link that corresponds to the application you wish to map.
4. This will display the 'View Application' screen. Click the 'Directories' tab.
5. This will display a list of directories that are currently mapped to the application. Select the new directory from the drop-down list and click the 'Add' button.
6. The new directory will be added to the bottom of the list of mapped directories. You can use the blue up-arrow or down-arrow to move a directory higher or lower in the order:
   
   Why directory order is important
7. You now need to choose which users within the directory may authenticate against the application. You have two choices:

- To allow all users within the directory to authenticate against the application, change 'Allow all to Authenticate' to 'True', then click the 'Update' button.
  OR:
- To allow only specific groups of users within the directory to authenticate against the application, see 3.4 Specifying which Groups can access an Application.

Screenshot: 'Application---Map Directories'



Related Topics

- 3.1 Using the Application Browser
- 3.2 Adding an Application
    - 3.2.1 Integrating Crowd with Apache
    - 3.2.2 Integrating Crowd with Atlassian Bamboo
    - 3.2.3 Integrating Crowd with Atlassian Confluence
    - 3.2.4 Integrating Crowd with Atlassian JIRA
    - 3.2.5 Integrating Crowd with Cenqua FishEye
    - 3.2.6 Integrating Crowd with Jive Forums
        - 3.2.6.1 Jive SSO
    - 3.2.7 Integrating Crowd with a Custom Application
- 3.3 Mapping a Directory to an Application
    - 3.3.1 Specifying the Directory Order for an Application
- 3.4 Specifying which Groups can access an Application
- 3.5 Specifying an Application's Address or Hostname
- 3.6 Managing an Application's Session
- 3.7 Deleting or Deactivating an Application

Crowd Documentation

3.3.1 Specifying the Directory Order for an Application

This page last changed on Apr 23, 2007 by rosie@atlassian.com.

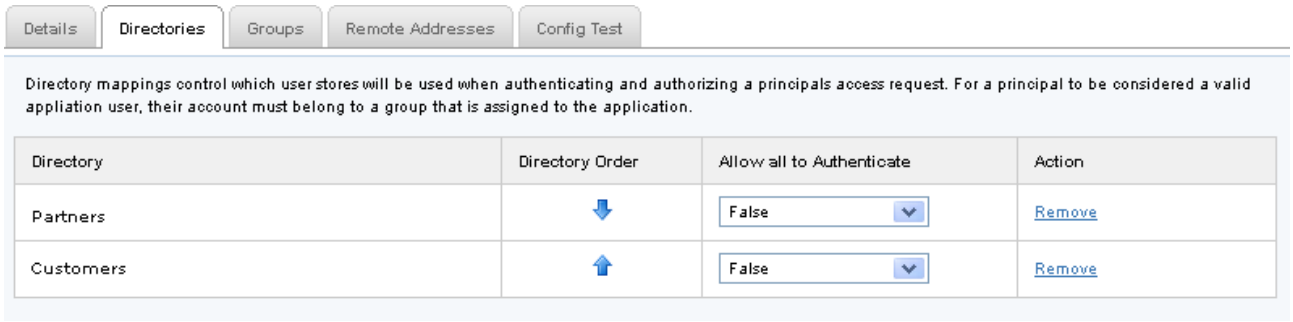
When you map multiple directories to an application, you also need to define the directory order. This is important in case the same user exists in multiple directories. When a user attempts to access an application, Crowd will search the directories in the order you specified, and will use the credentials (password, etc) of the first occurrence of the user to validate the login attempt (see diagram below).

To specify the directory order,

1. Login to the Crowd Administration Console.
2. Click the 'Applications' link in the top navigation bar.
3. This will display the Application Browser. Click the 'View' link that corresponds to the application you wish to map.
4. This will display the 'View Application' screen. Click the 'Directories' tab.
5. This will display a list of directories that are currently mapped to the application. Use the blue up-arrow or down-arrow to move a directory higher or lower in the order:

(Note: in Crowd, users are known as principals.)

Screenshot: 'Application---Mapped Directories'



How it works
Let's assume that JIRA has been set up as a Crowd application, and has been mapped to two directories, 'Partners' and 'Customers', in that order (as shown in the above screenshot).
Here is what happens when a user attempts to login to JIRA:

Related Topics

- 3.1 Using the Application Browser
- 3.2 Adding an Application
  - 3.2.1 Integrating Crowd with Apache
  - 3.2.2 Integrating Crowd with Atlassian Bamboo
  - 3.2.3 Integrating Crowd with Atlassian Confluence
  - 3.2.4 Integrating Crowd with Atlassian JIRA
  - 3.2.5 Integrating Crowd with Cenqua FishEye
  - 3.2.6 Integrating Crowd with Jive Forums

        - 3.2.6.1 Jive SSO
    - 3.2.7 Integrating Crowd with a Custom Application
- 3.3 Mapping a Directory to an Application
    - 3.3.1 Specifying the Directory Order for an Application
- 3.4 Specifying which Groups can access an Application
- 3.5 Specifying an Application's Address or Hostname
- 3.6 Managing an Application's Session
- 3.7 Deleting or Deactivating an Application

Crowd Documentation

3.4 Specifying which Groups can access an Application

This page last changed on Apr 02, 2007 by justin.

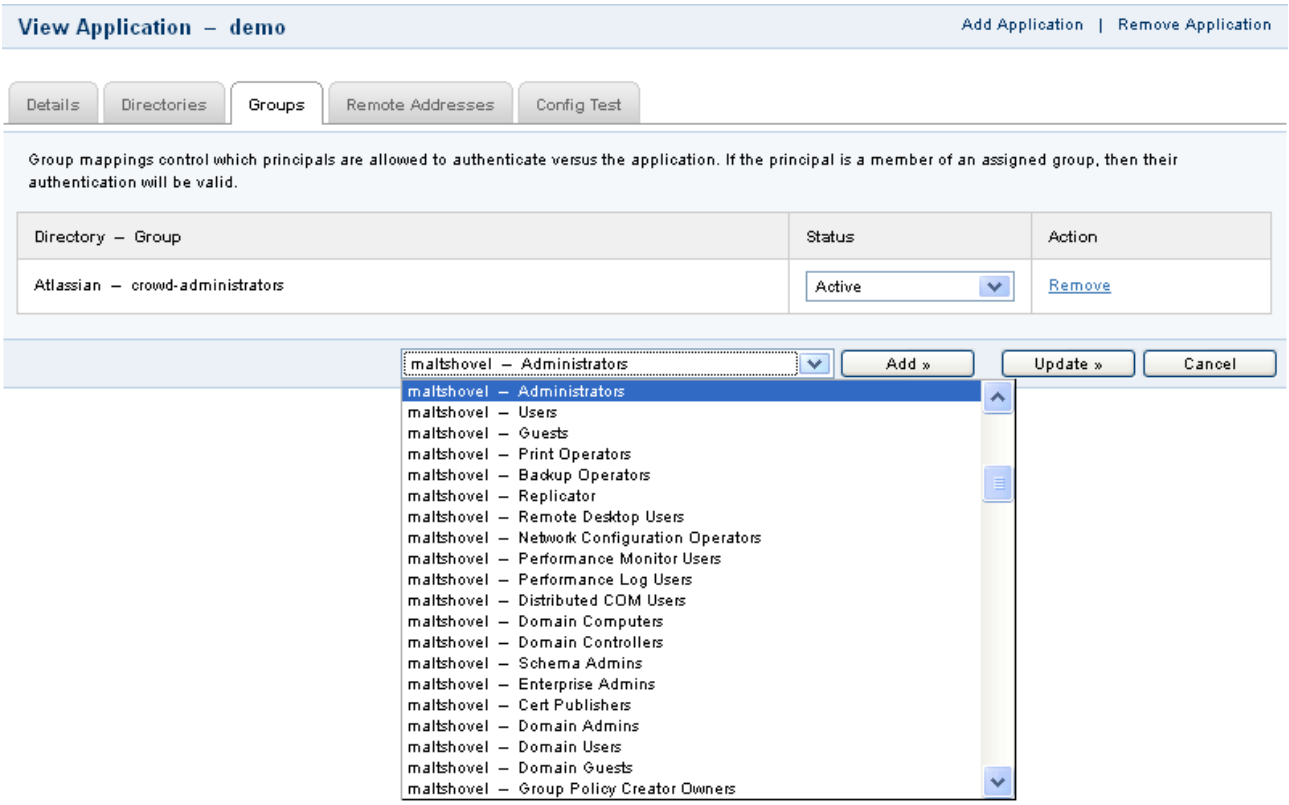
You can specify which principals (i.e. users) are allowed to authenticate against each application. For each mapped directory, you can either allow all users within the directory to authenticate with the application, or just particular groups within the directory.

For example, the default group `crowd-administrators`, which is automatically created in the default directory that you specified during setup, is allowed to access the Crowd Administration Console. This means that principals who belong to the group `crowd-administrators` are allowed to login to the Crowd Administration Console (assuming they supply a valid password).

To allow a group to access an application,

1. Login to the Crowd Administration Console.
2. Click the 'Applications' link in the top navigation bar.
3. This will display the Application Browser. Click the 'View' link that corresponds to the application you wish to map.
4. This will display the 'View Application' screen. Click the 'Groups' tab.
5. This will display a list of groups that currently have access to the application. Click the drop-down arrow next to the 'Add' button.
6. This will display a selection-list of all the groups that exist within each directory. Select the new group from the drop-down list and click the 'Add' button.

Alternatively, you can allow all users from a particular directory to authenticate against the application. See 3.3 Mapping a Directory to an Application.



See Also

4. Managing Principals, Groups and Roles

Related Topics

- 3.1 Using the Application Browser
- 3.2 Adding an Application
  - 3.2.1 Integrating Crowd with Apache
  - 3.2.2 Integrating Crowd with Atlassian Bamboo
  - 3.2.3 Integrating Crowd with Atlassian Confluence
  - 3.2.4 Integrating Crowd with Atlassian JIRA
  - 3.2.5 Integrating Crowd with Cenqua FishEye
  - 3.2.6 Integrating Crowd with Jive Forums
    - 3.2.6.1 Jive SSO
  - 3.2.7 Integrating Crowd with a Custom Application
- 3.3 Mapping a Directory to an Application
  - 3.3.1 Specifying the Directory Order for an Application
- 3.4 Specifying which Groups can access an Application
- 3.5 Specifying an Application's Address or Hostname
- 3.6 Managing an Application's Session
- 3.7 Deleting or Deactivating an Application

Crowd Documentation

3.5 Specifying an Application's Address or Hostname

This page last changed on Mar 29, 2007 by rosie@atlassian.com.

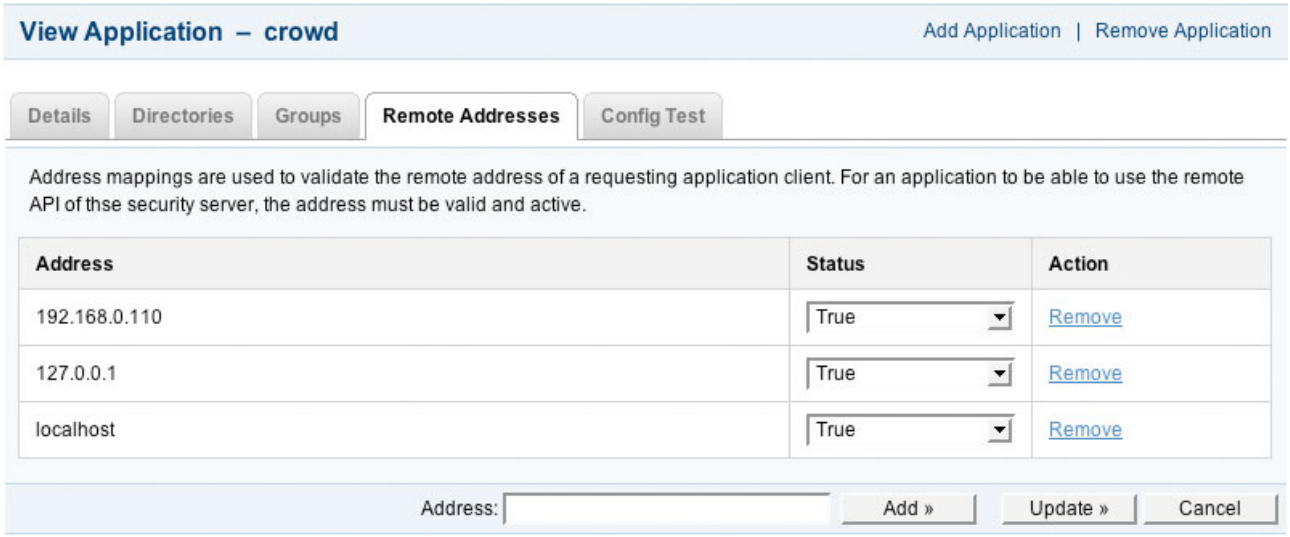
To ensure that your Crowd server can only be used by legitimate applications, Crowd will only allow applications to login from known addresses. This means that you need to specify the IP address(es) and/or hostname(s) of each application.

When you add a new application, it is restricted by default to `localhost` (`127.0.0.1`). If your application is on a different host, you will need to add the applicable host name or IP address, as described below.

To specify an application's IP address or hostname,

1. Login to the Crowd Administration Console.
2. Click the 'Applications link in the top navigation bar.
3. This will display the Application Browser. Click the 'View' link that corresponds to the application you wish to map.
4. This will display the 'View Application' screen. Click the 'Remote Addresses' tab.
5. This will display a list of IP addresses and hostnames that are currently mapped to the application. Type the new IP address or hostname into the 'Address' field and click the 'Add' button.
6. The new address will be added to the bottom of the list.
7. To verify that the address is valid, click the 'Config Test' tab, enter the application's 'Username' and 'Password', then click the 'Update' button. This will verify whether the application can successfully login to the Crowd framework.

Screenshot: 'Application---Addresses'



> **Common Misconfiguration**
> For an application to be able to use Crowd, the application's address must be valid and active.

Related Topics

- 3.1 Using the Application Browser
- 3.2 Adding an Application
    - 3.2.1 Integrating Crowd with Apache
    - 3.2.2 Integrating Crowd with Atlassian Bamboo
    - 3.2.3 Integrating Crowd with Atlassian Confluence
    - 3.2.4 Integrating Crowd with Atlassian JIRA
    - 3.2.5 Integrating Crowd with Cenqua FishEye
    - 3.2.6 Integrating Crowd with Jive Forums
        - 3.2.6.1 Jive SSO
    - 3.2.7 Integrating Crowd with a Custom Application
- 3.3 Mapping a Directory to an Application
    - 3.3.1 Specifying the Directory Order for an Application
- 3.4 Specifying which Groups can access an Application
- 3.5 Specifying an Application's Address or Hostname
- 3.6 Managing an Application's Session
- 3.7 Deleting or Deactivating an Application

Crowd Documentation

3.6 Managing an Application's Session

This page last changed on Apr 02, 2007 by rosie@atlassian.com.

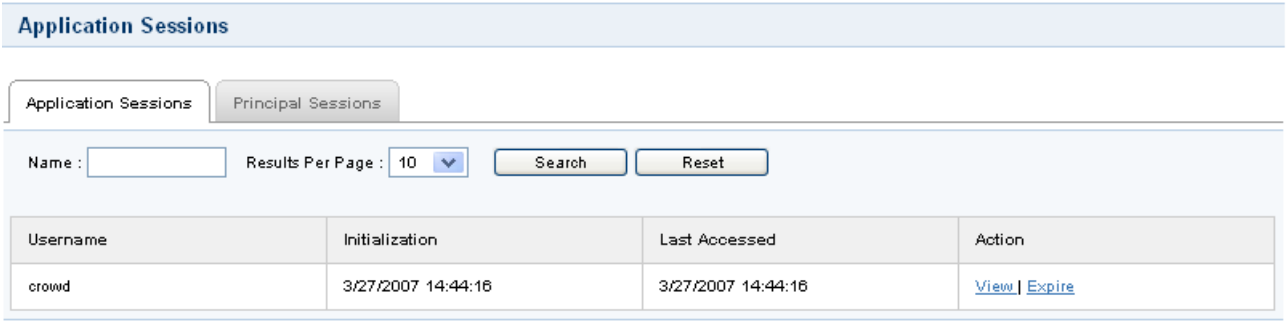
Crowd allows you to see a list of which applications are currently logged in to the Crowd framework. This is effectively a list of which applications currently have principals (users) logged in to them, since an application will only ever log in to the Crowd framework when it needs to authenticate a principal.

You can also force any session to expire, that is, you can log the application out of Crowd.

To see which applications are currently logged in to Crowd,

1. Login to the Crowd Administration Console.
2. Click the 'Sessions' link in the top navigation bar.
3. This will display the 'Session Browser'. Click the 'Application Sessions' tab.
4. This will display a list of all applications which are currently logged in to the Crowd framework. E.g. the screenshot below shows that the crowd application (i.e. the Crowd Administration Console) is currently logged in to the Crowd framework
   You can refine your search by specifying an application's 'Name' (note that this is case-sensitive).

Screenshot: 'Sessions---Applications'



To force an application to log out of Crowd,

1. Login to the Crowd Administration Console.
2. Click the 'Sessions' link in the top navigation bar.
3. Click the 'Application Sessions' tab.
4. This will display a list of all applications which are currently logged in to the Crowd framework. Click the application's 'Expire' link.

If you want to permanently prevent an application from logging in to Crowd, please see 3.7 Deleting or Deactivating an Application.

See Also...

4.4 Managing a Principal's Session
5.2.5 Session Timeout

Related Topics

- 3.1 Using the Application Browser
- 3.2 Adding an Application
    - 3.2.1 Integrating Crowd with Apache
    - 3.2.2 Integrating Crowd with Atlassian Bamboo
    - 3.2.3 Integrating Crowd with Atlassian Confluence
    - 3.2.4 Integrating Crowd with Atlassian JIRA
    - 3.2.5 Integrating Crowd with Cenqua FishEye
    - 3.2.6 Integrating Crowd with Jive Forums
        - 3.2.6.1 Jive SSO
    - 3.2.7 Integrating Crowd with a Custom Application
- 3.3 Mapping a Directory to an Application
    - 3.3.1 Specifying the Directory Order for an Application
- 3.4 Specifying which Groups can access an Application
- 3.5 Specifying an Application's Address or Hostname
- 3.6 Managing an Application's Session
- 3.7 Deleting or Deactivating an Application

Crowd Documentation

3.7 Deleting or Deactivating an Application

This page last changed on Apr 02, 2007 by rosie@atlassian.com.

Deactivating an application prevents principals (users) from logging in to the application. You might do this if you are making changes to an application and need to temporarily keep users out of it.

Deleting an application removes the application's details and its directory mappings. You would typically only do this if the application is no longer required.

To deactivate an application,

1. Login to the Crowd Administration Console.
2. Click the 'Applications link in the top navigation bar.
3. This will display the Application Browser. Click the 'View' link that corresponds to the application you wish to deactivate.
4. This will display the 'Application Details' screen. Deselect the 'Active' check-box, then click the 'Update' button. No users will now be able to log in to the application.

To reactivate the application, follow the same steps but select the 'Active' check-box.

To delete an application,

1. Login to the Crowd Administration Console.
2. Click the 'Applications link in the top navigation bar.
3. This will display the Application Browser. Click the 'View' link that corresponds to the application you wish to deactivate.
4. This will display the 'Application Details' screen. Click the 'Remove Application' link at the top-right of the screen.

The application will be removed from Crowd and will no longer appear in the Application Browser.

The 'crowd' application (i.e. the Crowd Administration Console) cannot be deleted or deactivated.

Related Topics

- 3.1 Using the Application Browser
- 3.2 Adding an Application
  - 3.2.1 Integrating Crowd with Apache
  - 3.2.2 Integrating Crowd with Atlassian Bamboo
  - 3.2.3 Integrating Crowd with Atlassian Confluence
  - 3.2.4 Integrating Crowd with Atlassian JIRA
  - 3.2.5 Integrating Crowd with Cenqua FishEye
  - 3.2.6 Integrating Crowd with Jive Forums
    - 3.2.6.1 Jive SSO
  - 3.2.7 Integrating Crowd with a Custom Application

- 3.3 Mapping a Directory to an Application
    - 3.3.1 Specifying the Directory Order for an Application
- 3.4 Specifying which Groups can access an Application
- 3.5 Specifying an Application's Address or Hostname
- 3.6 Managing an Application's Session
- 3.7 Deleting or Deactivating an Application

Crowd Documentation

4. Managing Principals, Groups and Roles

This page last changed on Mar 30, 2007 by rosie@atlassian.com.

In Crowd, users are referred to as principal entity objects (or just principals). Groups and roles are known as permission container objects.

Groups are particularly important in Crowd, as they are often used to control access to applications.

Roles are used less frequently, depending on the requirements of individual applications.

This section describes how to add/edit principals, groups and roles via the Crowd Administration Console. Note that the ability to do this depends on the permissions of the directory which contains the principals/groups/roles, and also depends on the directory being mapped to the Crowd Administration Console (i.e. the 'crowd' application).

- 4.1 Using the Principal Browser
- 4.2 Adding a Principal
- 4.3 Deleting or Deactivating a Principal
- 4.4 Managing a Principal's Session
- 4.5 Editing a Principal's Details and Password
- 4.6 Specifying a Principal's Attributes
- 4.7 Editing a Principal's Group and Role Membership
- 4.8 Using the Group Browser and Role Browser
- 4.9 Adding a Group or Role

4.1 Using the Principal Browser

This page last changed on Mar 30, 2007 by rosie@atlassian.com.

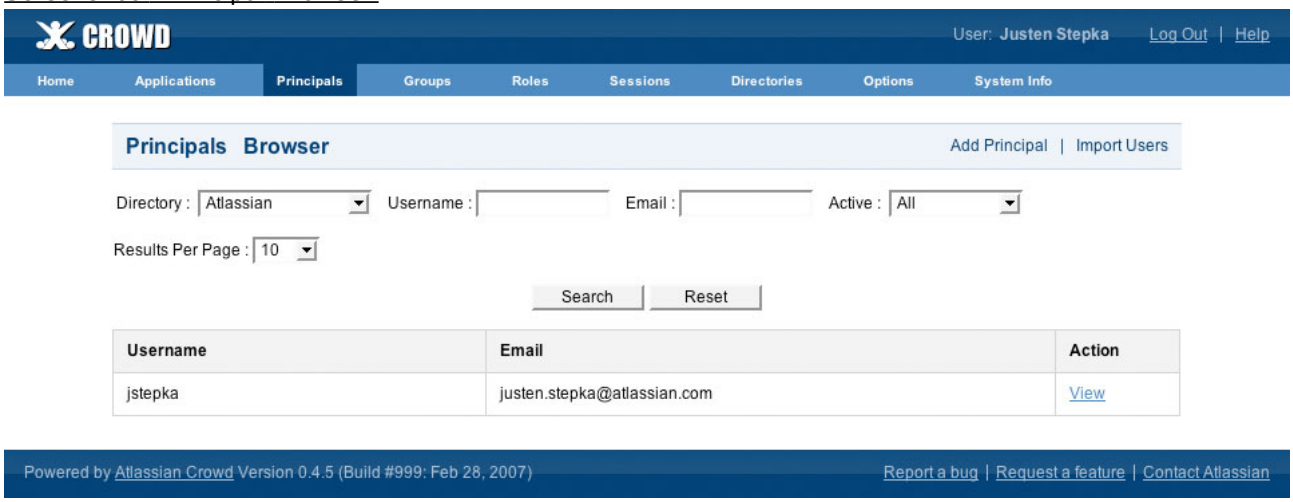
In Crowd, users are referred to as principal entity objects (or just principals).

The Principal Browser allows you to search, view, add and edit principals within a specified directory.

To use the Principal Browser,

1. Login to the Crowd Administration Console.
2. Click the 'Principals' link in the top navigation bar.
3. This will display the Principal Browser. Select the directory in which you are interested, then click the 'Search' button to list all the principals that exist in that directory.
   You can refine your search by specifying a 'Username' and/or 'Email' (note that these are case-sensitive), or 'Active'/'Inactive' principals. (An 'Inactive' principal is typically someone who has left your organisation.)
4. To view/edit a principal's details, click the 'View' link.

Screenshot: 'Principal Browser'



Related Topics

- 4.1 Using the Principal Browser
- 4.2 Adding a Principal
- 4.3 Deleting or Deactivating a Principal
- 4.4 Managing a Principal's Session
- 4.5 Editing a Principal's Details and Password
- 4.6 Specifying a Principal's Attributes
- 4.7 Editing a Principal's Group and Role Membership
- 4.8 Using the Group Browser and Role Browser
- 4.9 Adding a Group or Role

Crowd Documentation

4.2 Adding a Principal

This page last changed on Apr 02, 2007 by rosie@atlassian.com.

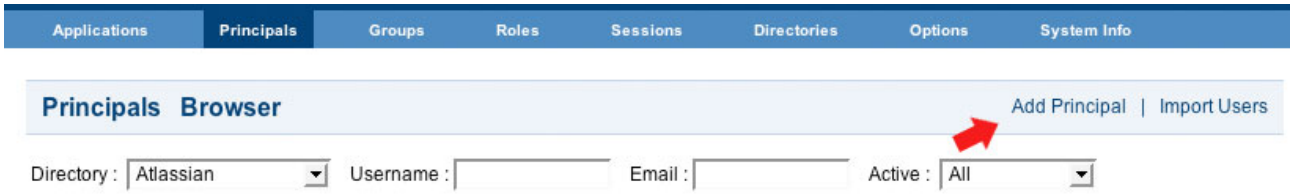
In Crowd, users are referred to as principal entity objects (or just principals).

To add a principal,

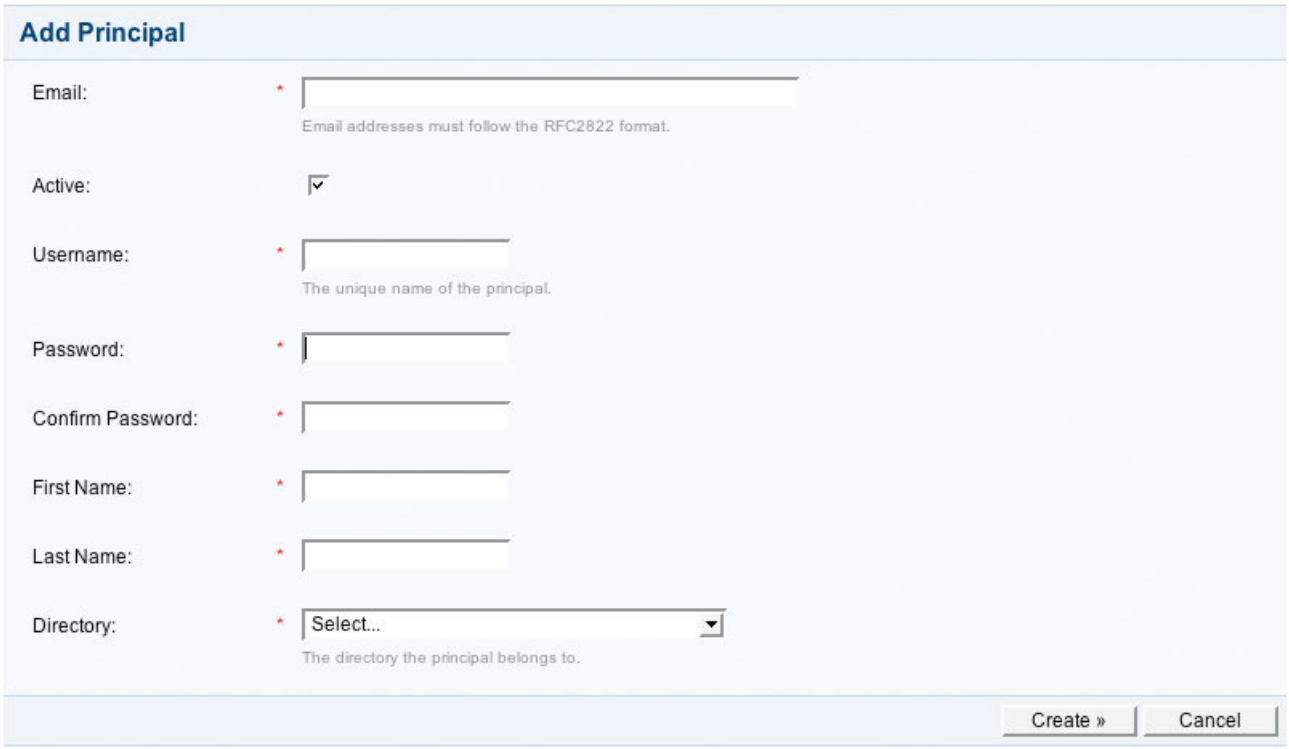
1. Login to the Crowd Administration Console.
2. Click the 'Principals' link in the top navigation bar.
3. This will display the Principal Browser. Click the 'Add Principal' link.
4. Complete the fields as described in the table below, then click the 'Create' button.
5. After creating the principal, you will be able to specify the principal's attributes and group/role membership.

Field	Description
Email	The email address of the principal. Email addresses must follow the RFC2822 format.
Active	Only deselect this if you wish to deny access to the principal.
Username	The login name of the principal. Within a given directory, the Username must be unique. Note that the Username cannot be changed once the principal is created.
Password	The password of the principal. If you have configured an Email Server and a Notification Template, Crowd will send the user an email notification about their new password.
First Name	The first name of the principal.
Last Name	The last name of the principal.
Directory	The directory to which the principal will be added. Note that the principal cannot be moved to a different directory once the principal is created.

Screenshot 1: 'Principal Browser'



Screenshot 2: 'Add Principal'



Related Topics

- 4.1 Using the Principal Browser
- 4.2 Adding a Principal
- 4.3 Deleting or Deactivating a Principal
- 4.4 Managing a Principal's Session
- 4.5 Editing a Principal's Details and Password
- 4.6 Specifying a Principal's Attributes
- 4.7 Editing a Principal's Group and Role Membership
- 4.8 Using the Group Browser and Role Browser
- 4.9 Adding a Group or Role

Crowd Documentation

4.3 Deleting or Deactivating a Principal

This page last changed on Mar 30, 2007 by rosie@atlassian.com.

Deactivating a principal (i.e. a user) prevents them from logging in to any applications that use the Crowd framework. You would typically do this when a user leaves your organisation.

Deleting a principal removes them completely from the relevant directory. It is generally recommended that you deactivate a principal rather than delete them, in case some applications contain historical data (e.g. documents that the user has created).

To deactivate a principal,

1. Login to the Crowd Administration Console.
2. Click the 'Principals' link in the top navigation bar.
3. This will display the Principal Browser. Select the relevant directory, locate the principal you wish to deactivate, and click the 'View' link that corresponds to the principal.
4. This will display the 'Principal Details' screen. Deselect the 'Active' check-box, then click the 'Update' button. The principal will now be unable to log in to any applications which use the Crowd framework.

To delete a principal,

1. Login to the Crowd Administration Console.
2. Click the 'Principals' link in the top navigation bar.
3. This will display the Principal Browser. Click the 'View' link that corresponds to the principal you wish to delete.
4. This will display the 'Principal Details' screen. Click the 'Remove Principal' link at the top-right of the screen.

The principal will be removed from the relevant directory and will no longer appear in the Principal Browser..

Related Topics

- 4.1 Using the Principal Browser
- 4.2 Adding a Principal
- 4.3 Deleting or Deactivating a Principal
- 4.4 Managing a Principal's Session
- 4.5 Editing a Principal's Details and Password
- 4.6 Specifying a Principal's Attributes
- 4.7 Editing a Principal's Group and Role Membership
- 4.8 Using the Group Browser and Role Browser
- 4.9 Adding a Group or Role

Crowd Documentation

4.4 Managing a Principal's Session

This page last changed on Apr 02, 2007 by rosie@atlassian.com.

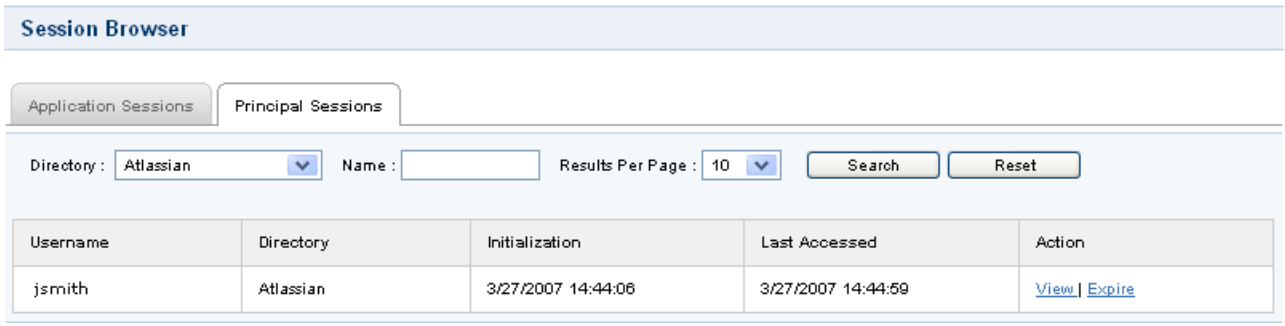
For any given directory, Crowd allows you to see which principals (users) are currently logged in to one or more applications that use the Crowd framework.

You can also force any session to expire, that is, you can log the principal out of Crowd.

To see which principals are currently logged in to Crowd,

1. Login to the Crowd Administration Console.
2. Click the 'Sessions' link in the top navigation bar.
3. This will display the 'Session Browser'. Click the 'Principal Sessions' tab.
4. Select the directory containing the principals in which you are interested, and click the 'Search' button.
5. This will display a list of all principals, within your chosen directory, who are currently logged in to the Crowd framework.
   You can refine your search by specifying a principal's 'Name' (note that this is case-sensitive).

Screenshot: 'Session Browser — Principals'



To log a principal out of Crowd,

1. Login to the Crowd Administration Console.
2. Click the 'Sessions' link in the top navigation bar.
3. Click the 'Principal Sessions' tab.
4. This will display a list of all principals which are currently logged in to the Crowd framework. Click the principal's 'Expire' link.

If you want to permanently prevent a principal from logging in to Crowd, please see 4.3 Deleting or Deactivating a Principal.

See Also...

3.6 Managing an Application's Session
5.2.5 Session Timeout

Related Topics

- 4.1 Using the Principal Browser
- 4.2 Adding a Principal
- 4.3 Deleting or Deactivating a Principal
- 4.4 Managing a Principal's Session
- 4.5 Editing a Principal's Details and Password
- 4.6 Specifying a Principal's Attributes
- 4.7 Editing a Principal's Group and Role Membership
- 4.8 Using the Group Browser and Role Browser
- 4.9 Adding a Group or Role

Crowd Documentation

4.5 Editing a Principal's Details and Password

This page last changed on Apr 02, 2007 by rosie@atlassian.com.

In Crowd, users are referred to as principal entity objects (or just principals).

To edit a principal's details,

1. Login to the Crowd Administration Console.
2. Click the 'Principals' link in the top navigation bar.
3. This will display the Principal Browser. Select the relevant directory, locate the principal in which you are interested, then click the 'View' link corresponding to the principal.
4. This will display the 'Principal Details' screen.
5. Edit the details as required, then click the 'Update' button.

To change a principal's password,

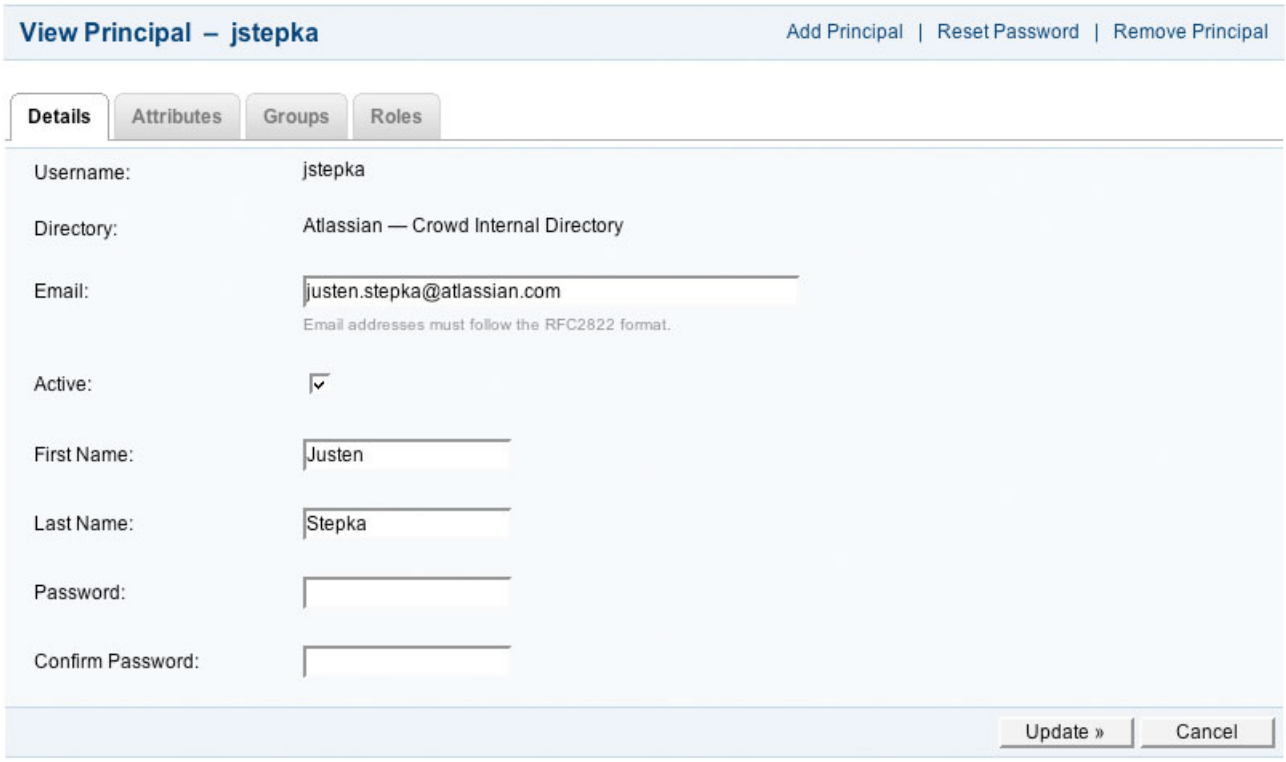
1. Login to the Crowd Administration Console.
2. Click the 'Principals' link in the top navigation bar.
3. This will display the Principal Browser. Select the relevant directory, locate the principal in which you are interested, then click the 'View' link corresponding to the principal.
4. This will display the 'Principal Details' screen. You can either:

- Enter the new password as required, then click the 'Update' button;
  OR

- Click the 'Reset Password' link. This will generate a new password (i.e. one which you do not know) and email it to the user.

If you have configured an Email Server and a Notification Template, Crowd will send the user an email notification about their new password.

Screenshot: 'Principal Details'



Related Topics

- 4.1 Using the Principal Browser
- 4.2 Adding a Principal
- 4.3 Deleting or Deactivating a Principal
- 4.4 Managing a Principal's Session
- 4.5 Editing a Principal's Details and Password
- 4.6 Specifying a Principal's Attributes
- 4.7 Editing a Principal's Group and Role Membership
- 4.8 Using the Group Browser and Role Browser
- 4.9 Adding a Group or Role

Crowd Documentation

4.6 Specifying a Principal's Attributes

This page last changed on Mar 30, 2007 by rosie@atlassian.com.

In Crowd, users are referred to as principal entity objects (or just principals).

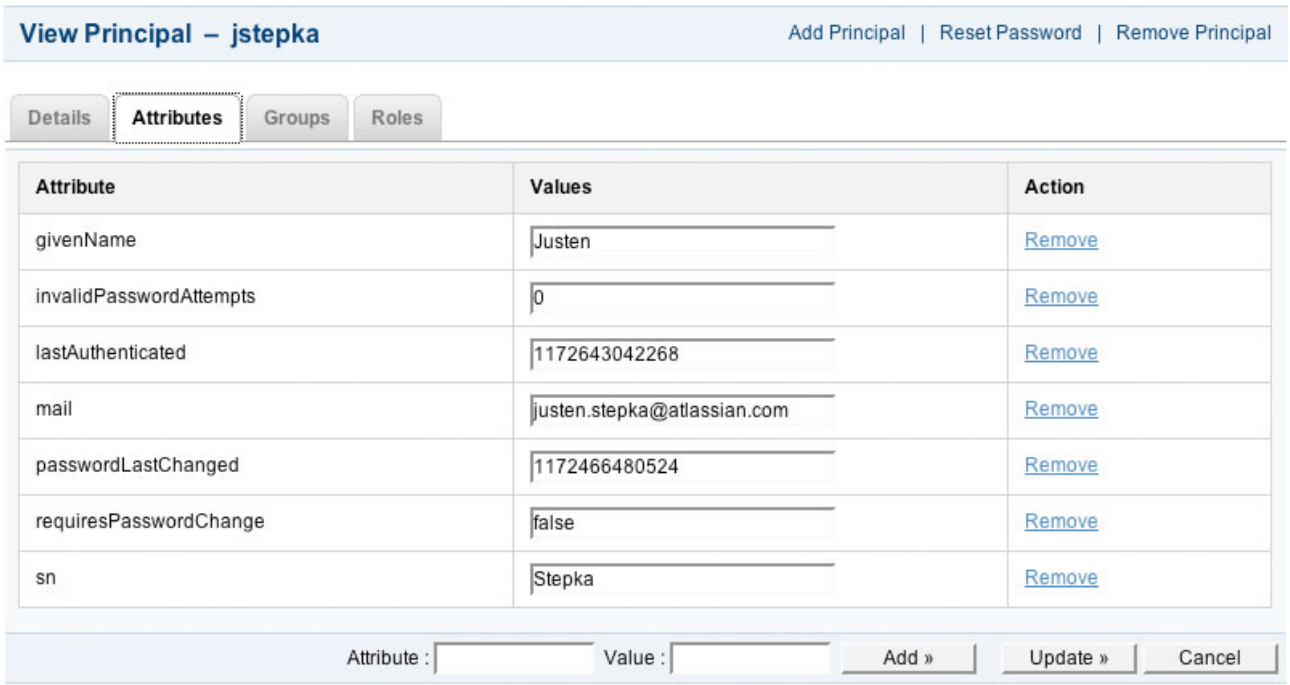
A principal's default attributes are specific to the directory to which the principal belongs. You can add other attributes (e.g. address, phone number, date of birth) manually as required.

To edit a principal's attributes,

1. Login to the Crowd Administration Console.
2. Click the 'Principals' link in the top navigation bar.
3. This will display the Principal Browser. Select the relevant directory, locate the principal in which you are interested, then click the 'View' link corresponding to the principal.
4. This will display the 'Principal Details' screen. Click the 'Attributes' tab.

- To add a new attribute,
  1. Type the name of the new attribute (e.g. `phone`) in the 'Attribute' field at the bottom of the screen.
  2. Type the value of the new attribute (e.g. `0123456789`) in the 'Value' field at the bottom of the screen.
  3. Click the 'Add' button.
- To edit an existing attribute, edit the corresponding field in the 'Values' column, then click the 'Update' button.
- To delete an attribute, click the corresponding 'Remove' link in the 'Action' column.

Note that some attributes may correspond to particular fields on the Principal Details screen. However, attributes are optional, whereas the 'Details' fields are all required.
Screenshot: 'Principal Attributes'



Related Topics

- 4.1 Using the Principal Browser
- 4.2 Adding a Principal
- 4.3 Deleting or Deactivating a Principal
- 4.4 Managing a Principal's Session
- 4.5 Editing a Principal's Details and Password
- 4.6 Specifying a Principal's Attributes
- 4.7 Editing a Principal's Group and Role Membership
- 4.8 Using the Group Browser and Role Browser
- 4.9 Adding a Group or Role

Crowd Documentation

4.7 Editing a Principal's Group and Role Membership

This page last changed on Mar 30, 2007 by rosie@atlassian.com.

Within any given directory, you can choose the groups and roles to which each principal (user) belongs.

Note that a principal's group membership is particularly important, as groups are often used to control access to applications.

To add a principal to a group,

1. Login to the Crowd Administration Console.
2. Click the 'Principals' link in the top navigation bar.
3. This will display the Principal Browser. Select the relevant directory, locate the principal you wish to deactivate, and click the 'View' link that corresponds to the principal.
4. This will display the 'Principal Details' screen. Click the 'Groups' tab.
5. A list of the principal's current groups (if any) will be displayed. Select the relevant group from the drop-down box below the list, then click the 'Add' button.

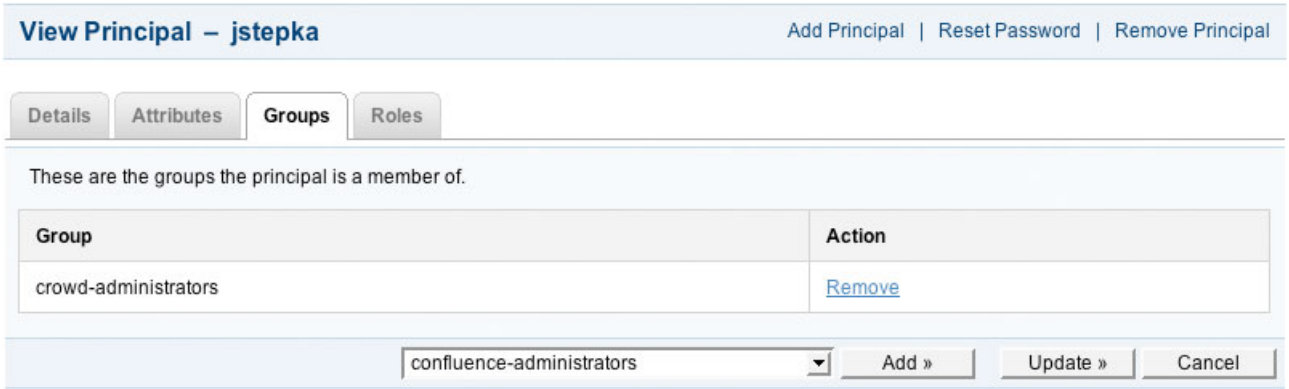
The principal will now be authorised to use any applications that use this group to control access.

To remove a principal to a group,

1. Login to the Crowd Administration Console.
2. Click the 'Principals' link in the top navigation bar.
3. This will display the Principal Browser. Select the relevant directory, locate the principal you wish to deactivate, and click the 'View' link that corresponds to the principal.
4. This will display the 'Principal Details' screen. Click the 'Groups' tab.
5. A list of the principal's current groups (if any) will be displayed. Click the 'Remove' link corresponding to the relevant group.

The principal will now be unable to login to any applications that use this group to control access.

Screenshot: 'Principal — Groups'



The adding or removing of a principal to or from a role is performed via the Role Browser, but is otherwise identical to the process for groups.

Related Topics

- 4.1 Using the Principal Browser
- 4.2 Adding a Principal
- 4.3 Deleting or Deactivating a Principal
- 4.4 Managing a Principal's Session
- 4.5 Editing a Principal's Details and Password
- 4.6 Specifying a Principal's Attributes
- 4.7 Editing a Principal's Group and Role Membership
- 4.8 Using the Group Browser and Role Browser
- 4.9 Adding a Group or Role

Crowd Documentation

4.8 Using the Group Browser and Role Browser

This page last changed on Apr 23, 2007 by rosie@atlassian.com.

About Groups and Roles

Groups and roles contain users (which are known in Crowd as principals).
Groups and roles are known as permission container objects.

Groups are particularly important in Crowd, as they are often used to control access to applications.

Roles are used less frequently, depending on the requirements of individual applications.

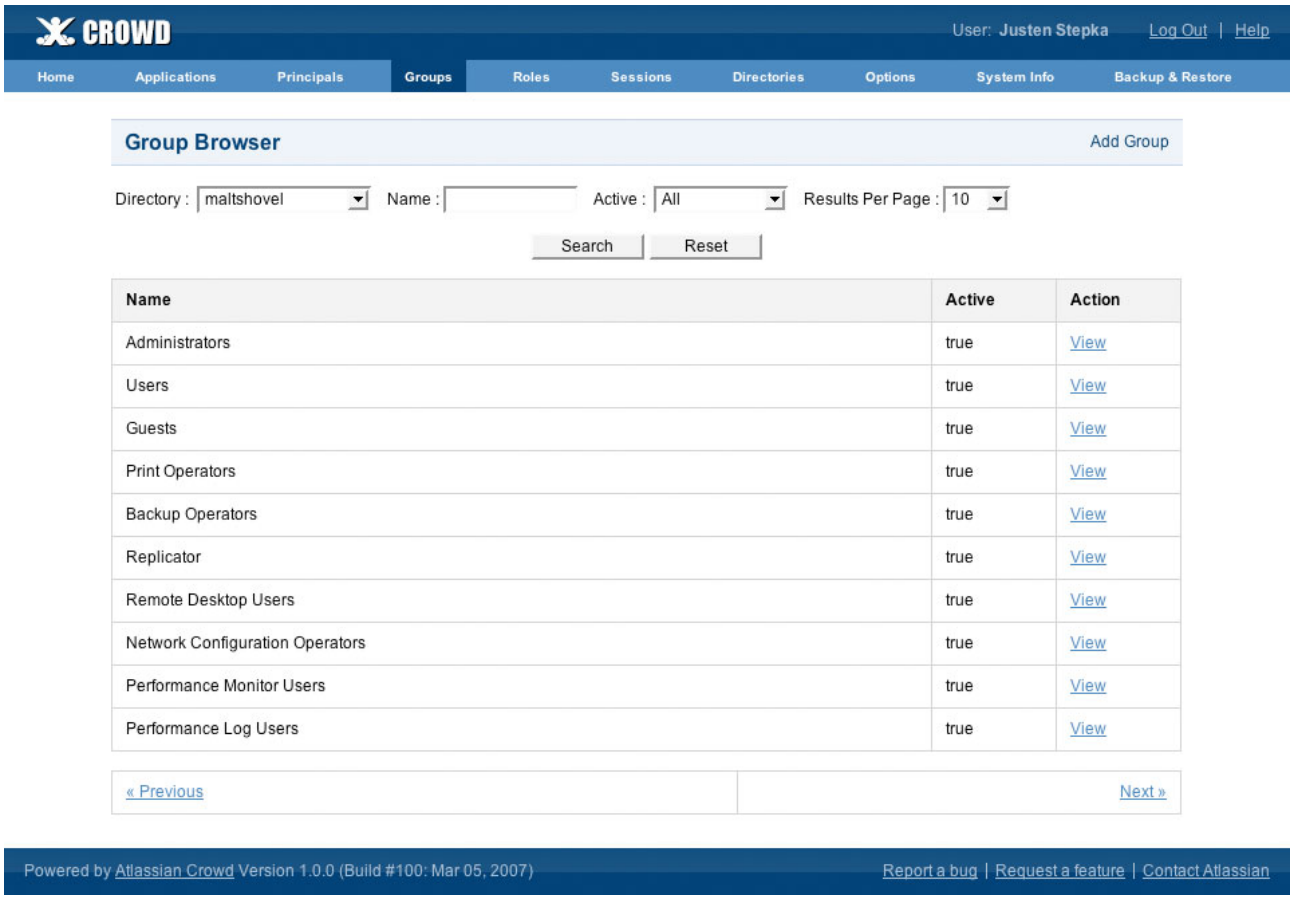
About the Group Browser and the Role Browser

The Group Browser and Role Browser are very similar. They allow you to search, view, add and edit the various groups and roles stored within a specified directory.

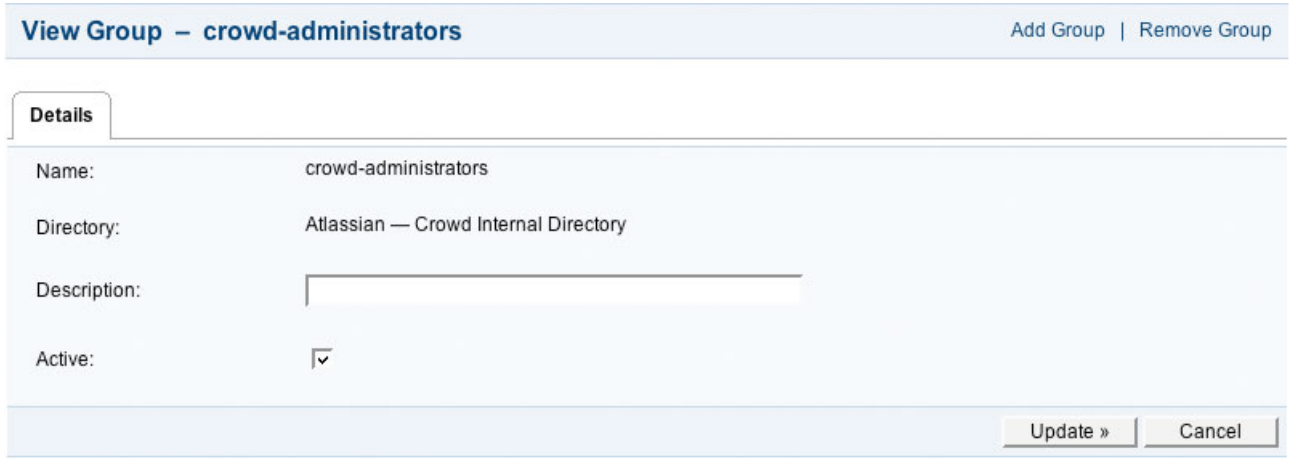
To use the Group Browser,

1. Login to the Crowd Administration Console.
2. Click the 'Groups' link in the top navigation bar.
3. This will display the Group Browser. Select the directory in which you are interested, then click the 'Search' button to list all the groups that exist in that directory.
   You can refine your search by specifying a 'Name' (note that this is case-sensitive), or 'Active'/'Inactive' groups.
4. To view/edit a group's details, click the 'View' link.

Screenshot 1: 'Group Browser'



Screenshot 2: 'View/Update Group Details'



Related Topics

- 4.1 Using the Principal Browser
- 4.2 Adding a Principal
- 4.3 Deleting or Deactivating a Principal
- 4.4 Managing a Principal's Session
- 4.5 Editing a Principal's Details and Password
- 4.6 Specifying a Principal's Attributes
- 4.7 Editing a Principal's Group and Role Membership
- 4.8 Using the Group Browser and Role Browser
- 4.9 Adding a Group or Role

Crowd Documentation

4.9 Adding a Group or Role

This page last changed on Mar 30, 2007 by rosie@atlassian.com.

Groups and roles are known as permission container objects.

Groups are particularly important in Crowd, as they are often used to control access to applications.

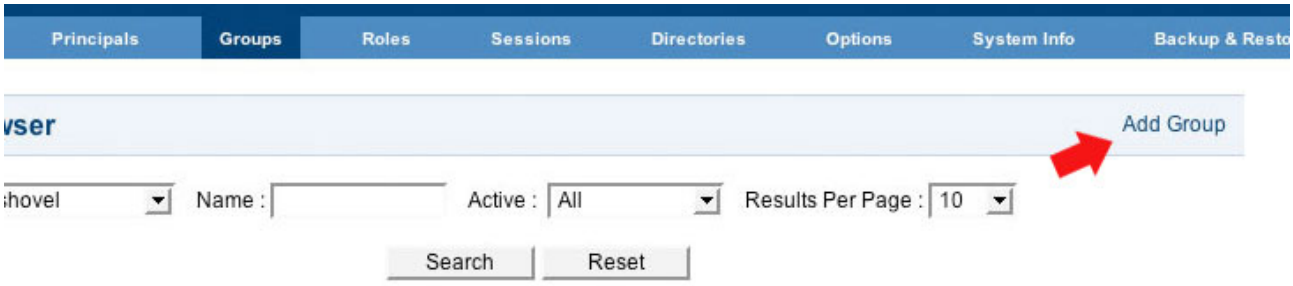
Roles are used less frequently, depending on the requirements of individual applications.

To add a group or role,

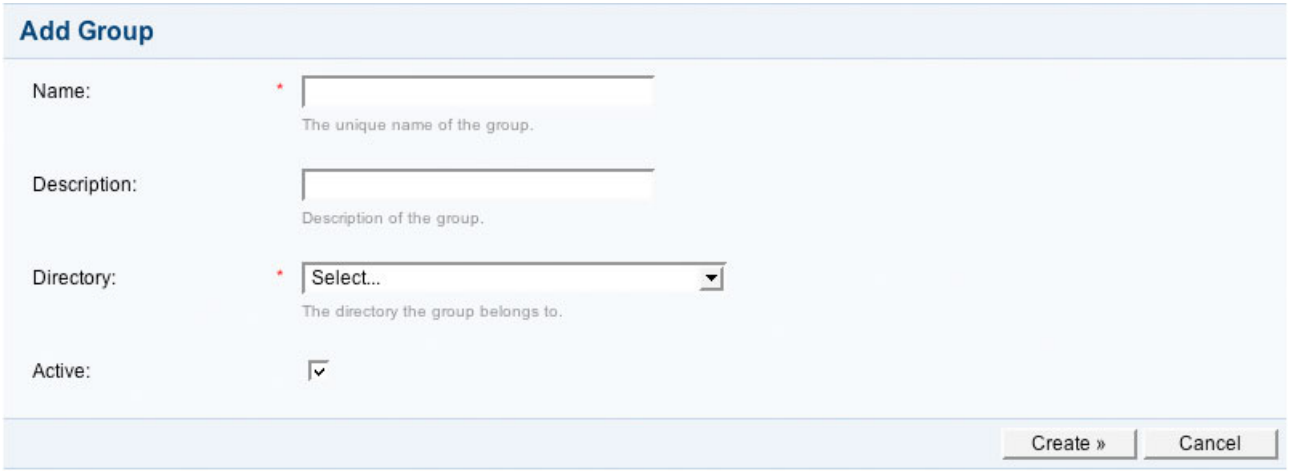
1. Login to the Crowd Administration Console.
2. Click the 'Groups' link (or the 'Roles' link) in the top navigation bar.
3. This will display the Group Browser (or Role Browser). Click the 'Add Group' link (or the 'Add Role' link).
4. Complete the fields as described in the table below, then click the 'Create' button.
   You can now add principals (users) to the new group or role.

Field	Description
Name	The unique name of the group or role. Within a given directory, the Name must be unique. Note that the Name cannot be changed once the group or role is created.
Description	A short description of the group or role.
Directory	The directory to which the group or role will be added. Note that the group or role cannot be moved to a different directory after it is created.
Active	Only deselect this if you wish to deny access to all members of the group or role.

Screenshot 1: 'Group Browser'



Screenshot 2: 'Add Group'



Groups (not roles) can also be added via Crowd's migration tools — see 2.4 Importing Principals and Groups into a Directory.

See Also

3.4 Specifying which Groups can access an Application

Related Topics

- 4.1 Using the Principal Browser
- 4.2 Adding a Principal
- 4.3 Deleting or Deactivating a Principal
- 4.4 Managing a Principal's Session
- 4.5 Editing a Principal's Details and Password
- 4.6 Specifying a Principal's Attributes
- 4.7 Editing a Principal's Group and Role Membership
- 4.8 Using the Group Browser and Role Browser
- 4.9 Adding a Group or Role

Crowd Documentation

5. System Administration

This page last changed on Mar 12, 2007 by rosie@atlassian.com.

- 5.1 Viewing Crowd's System Information
- 5.2 Configuring Server Settings
    - 5.2.1 Licensing
    - 5.2.2 Deployment Title
    - 5.2.3 Domain
    - 5.2.4 Token Seed
    - 5.2.5 Session Timeout
    - 5.2.6 Caching
        - 5.2 Configuring Caching for an Application
- 5.3 Configuring SMTP Email
    - 5.3.1 Creating an Email Notification Template
- 5.4 Backing Up and Restoring Data
- 5.5 Logging

5.1 Viewing Crowd's System Information

This page last changed on Apr 02, 2007 by rosie@atlassian.com.

Crowd provides a useful summary of your server's system information, including:

- time and date information
- Java version
- memory usage
- application server details
- server ID (see 5.2.1 Licensing for more details)

To view your Crowd server's system information,

1. Login to the Crowd Administration Console.
2. Click the 'System Info' link in the top navigation bar.

Screenshot: 'System Information'

System Information

Date:	Monday, 02 Apr 2007
Time:	15:18:40
Timezone:	Eastern Standard Time (New South Wales)
Java Version:	1.5.0_06
Java Vendor:	Sun Microsystems Inc.
JVM Version:	1.5.0_06-b05
JVM Vendor:	Sun Microsystems Inc.
JVM Runtime:	Java HotSpot(TM) Client VM
Username:	administrator
Operating System:	Windows 20035.2
Architecture:	x86

JVM Statistics

Total Memory:	63 MB
Used Memory:	47 MB
Free Memory:	15 MB

Runtime Information

Application Server:	Apache Tomcat/5.5.20
Database Dialect:	org.hibernate.dialect.HSQLDialect
Version:	1.0.3
Build Number:	104
Build Date:	Mar 22, 2007

License Information

License Server ID:	ANB2-YN5T- ANB2-YN5T

Related Topics

- 5.1 Viewing Crowd's System Information
- 5.2 Configuring Server Settings
  - 5.2.1 Licensing

    - 5.2.2 Deployment Title
    - 5.2.3 Domain
    - 5.2.4 Token Seed
    - 5.2.5 Session Timeout
    - 5.2.6 Caching
        - 5.2 Configuring Caching for an Application
- 5.3 Configuring SMTP Email
    - 5.3.1 Creating an Email Notification Template
- 5.4 Backing Up and Restoring Data
- 5.5 Logging

Crowd Documentation

5.2 Configuring Server Settings

This page last changed on Apr 02, 2007 by rosie@atlassian.com.

You can alter the settings which were specified when your Crowd server was installed:

- 5.2.1 Licensing
- 5.2.2 Deployment Title
- 5.2.3 Domain
- 5.2.4 Token Seed
- 5.2.5 Session Timeout
- 5.2.6 Caching

Related Topics

- 5.1 Viewing Crowd's System Information
- 5.2 Configuring Server Settings
    - 5.2.1 Licensing
    - 5.2.2 Deployment Title
    - 5.2.3 Domain
    - 5.2.4 Token Seed
    - 5.2.5 Session Timeout
    - 5.2.6 Caching
        - 5.2 Configuring Caching for an Application
- 5.3 Configuring SMTP Email
    - 5.3.1 Creating an Email Notification Template
- 5.4 Backing Up and Restoring Data
- 5.5 Logging

Crowd Documentation

5.2.1 Licensing

This page last changed on Apr 02, 2007 by rosie@atlassian.com.

Crowd licenses are based on the number of end-users who will login to one or more of the applications that are integrated with Crowd. Evaluation licenses may be obtained from the Atlassian website. When you obtain an evaluation license — or purchase, renew or upgrade your license — you will receive a license key via email. You will need to enter your license key into your Crowd server as follows.

Note:

If the number of users who are allowed to login to the Crowd framework exceeds the user license limit, noone will be able to login to any applications (other than the Crowd Administration Console). If this happens, you can obtain a temporary license from Atlassian.

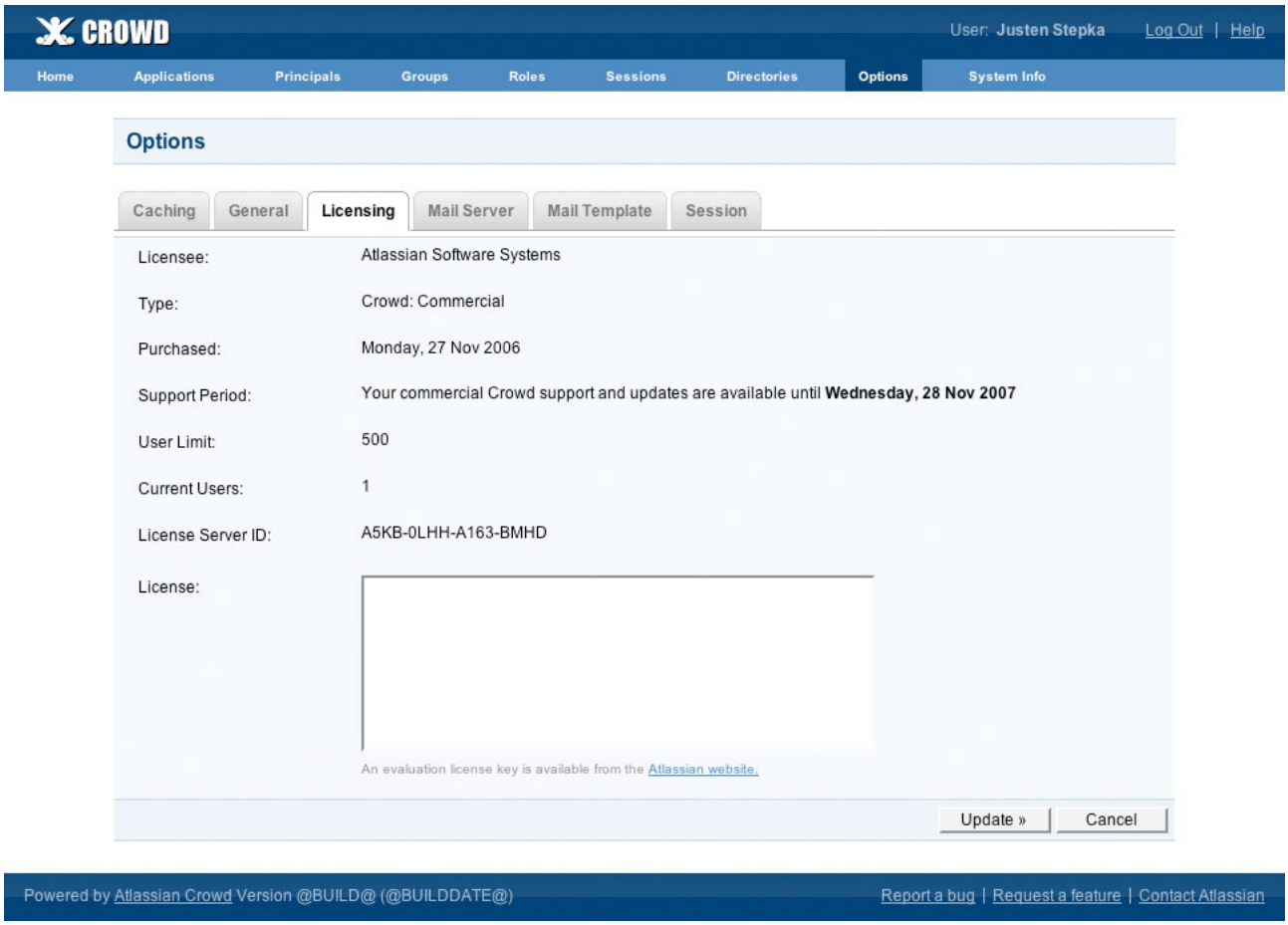
To minimise your licensing cost, if you have more than one directory, ensure that the same user does not exist in multiple directories. It is also recommended that you alllow only particular groups to login to each application, rather than entire directories. Note that a mapped application can 'see' all users in a directory, even if not all of them can login to the application. For example, a Human Resources application might be mapped to your entire Active Directory server, but only the HR group is allowed to login to the application.

To enter your license key,

1. Login to the Crowd Administration Console.
2. Click the 'Options' link in the top navigation bar.
3. Click the 'Licensing' tab.
4. Type (or paste) your license key into the 'License' field.
5. Click the 'Update' button.

Your Server ID is generated automatically, based on your license key.

Screenshot: 'Licensing'



Related Topics

- 5.1 Viewing Crowd's System Information
- 5.2 Configuring Server Settings
  - 5.2.1 Licensing
  - 5.2.2 Deployment Title
  - 5.2.3 Domain
  - 5.2.4 Token Seed
  - 5.2.5 Session Timeout
  - 5.2.6 Caching
    - 5.2 Configuring Caching for an Application
- 5.3 Configuring SMTP Email
  - 5.3.1 Creating an Email Notification Template
- 5.4 Backing Up and Restoring Data
- 5.5 Logging

Crowd Documentation

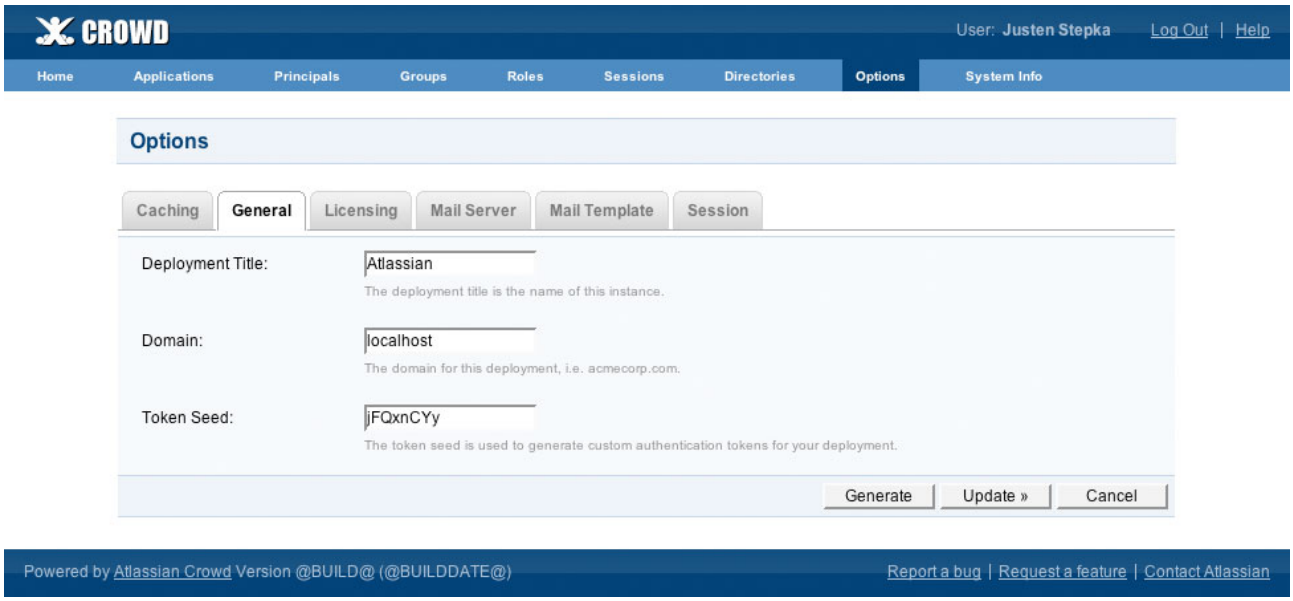
5.2.2 Deployment Title

This page last changed on Apr 02, 2007 by rosie@atlassian.com.

The Deployment Title specifies a unique name for your Crowd instance. The Deployment Title can be used when sending email notifications.
To specify the Deployment Title,

1. Login to the Crowd Administration Console.
2. Click the 'Options' link in the top navigation bar.
3. Click the 'General' tab.
4. Type the new name into the 'Deployment Title' field, then click the 'Update' button.

Screenshot: 'General Options'



Related Topics

- 5.1 Viewing Crowd's System Information
- 5.2 Configuring Server Settings
  - 5.2.1 Licensing
  - 5.2.2 Deployment Title
  - 5.2.3 Domain
  - 5.2.4 Token Seed
  - 5.2.5 Session Timeout
  - 5.2.6 Caching
    - 5.2 Configuring Caching for an Application
- 5.3 Configuring SMTP Email
  - 5.3.1 Creating an Email Notification Template
- 5.4 Backing Up and Restoring Data
- 5.5 Logging

Crowd Documentation

5.2.3 Domain

This page last changed on Apr 02, 2007 by rosie@atlassian.com.

The Domain is used when setting HTTP authentication cookies in a user's browser. If this attribute is not the correct, single sign-on will not work when the user switches between applications.

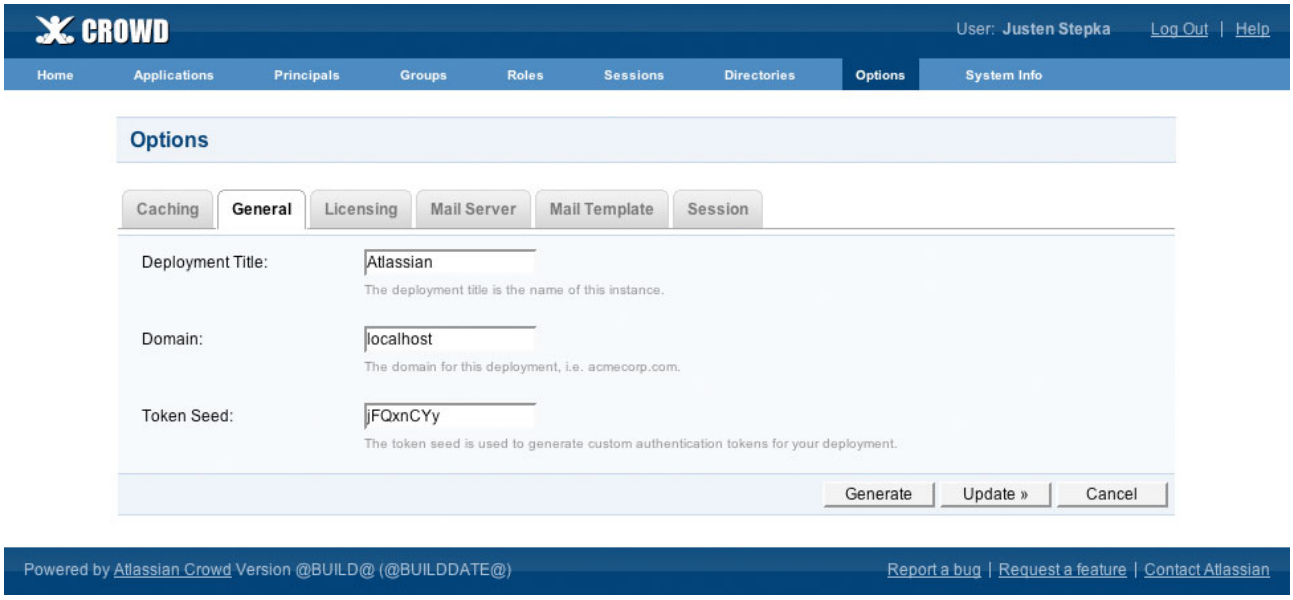
Note:

- When developing on your local machine, the domain should be set to `localhost`.
- If you wish to have single sign-on (SSO) support for *.mydomain.com, you will need to set the Domain to `.mydomain.com`. Please note the '.' before the top-level-domain.

To specify the Domain,

1. Login to the Crowd Administration Console.
2. Click the 'Options' link in the top navigation bar.
3. Click the 'General' tab.
4. Type the new domain into the 'Domain' field, then click the 'Update' button.

Screenshot: 'General Options'



Related Topics

- 5.1 Viewing Crowd's System Information
- 5.2 Configuring Server Settings
  - 5.2.1 Licensing
  - 5.2.2 Deployment Title
  - 5.2.3 Domain

    - 5.2.4 Token Seed
    - 5.2.5 Session Timeout
    - 5.2.6 Caching
        - 5.2 Configuring Caching for an Application
- 5.3 Configuring SMTP Email
    - 5.3.1 Creating an Email Notification Template
- 5.4 Backing Up and Restoring Data
- 5.5 Logging

Crowd Documentation

5.2.4 Token Seed

This page last changed on Apr 02, 2007 by rosie@atlassian.com.

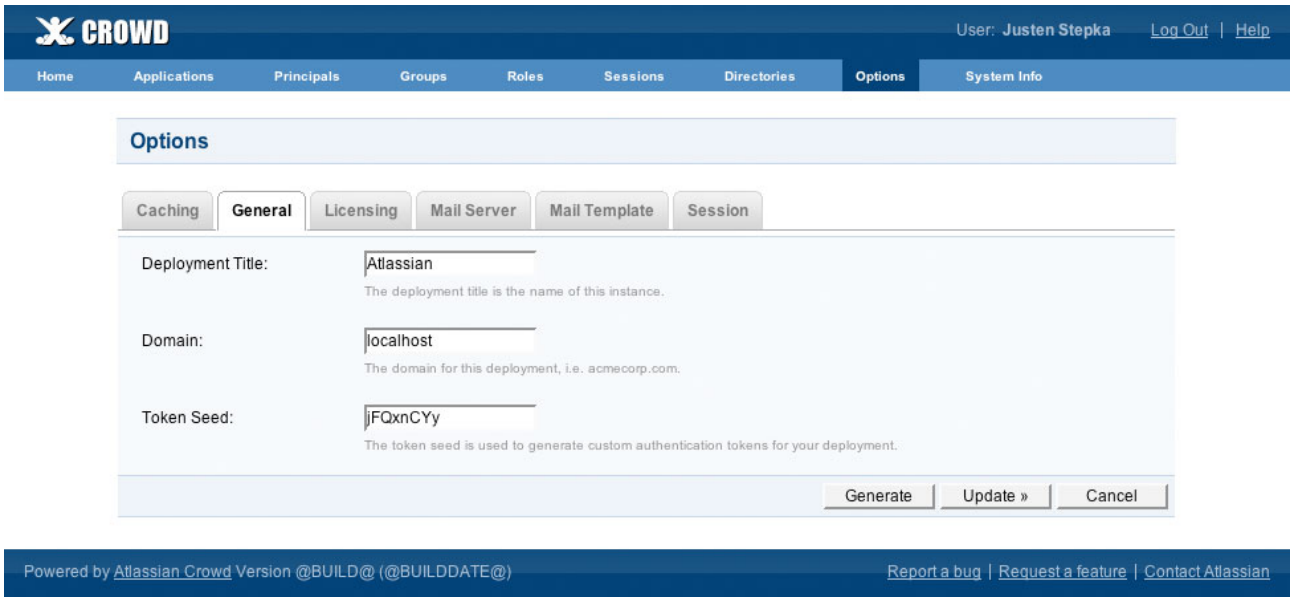
The Token Seed is a unique key for each site deployment of Crowd. This key is used when generating tokens for an authenticated application.

To specify the Token Seed,

1. Login to the Crowd Administration Console.
2. Click the 'Options' link in the top navigation bar.
3. Click the 'General' tab.
4. You can either:

- Type the new key into the 'Token Seed' field, then click the 'Update' button;
  OR
- Click the 'Generate' button to automatically create a random key.

Screenshot: 'General Options'



Related Topics

- 5.1 Viewing Crowd's System Information
- 5.2 Configuring Server Settings
  - 5.2.1 Licensing
  - 5.2.2 Deployment Title
  - 5.2.3 Domain
  - 5.2.4 Token Seed
  - 5.2.5 Session Timeout

    - 5.2.6 Caching
        - 5.2 Configuring Caching for an Application
- 5.3 Configuring SMTP Email
    - 5.3.1 Creating an Email Notification Template
- 5.4 Backing Up and Restoring Data
- 5.5 Logging

Crowd Documentation

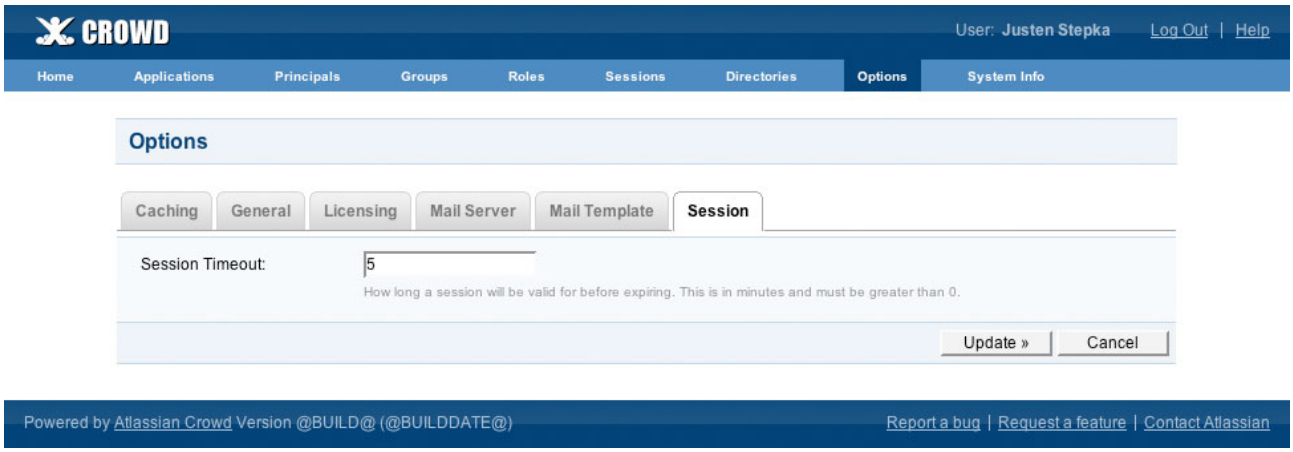
5.2.5 Session Timeout

This page last changed on Apr 02, 2007 by rosie@atlassian.com.

When a successful authentication occurs, for either an application or a principal (i.e. a user), a unique token is assigned. Tokens are valid for the period of time specified as the Session Timeout attribute. The Session Timeout controls how long a session will be considered valid during any period of inactivity. This is in minutes and must be greater than `0`.
To specify the Session Timeout,

1. Login to the Crowd Administration Console.
2. Click the 'Options' link in the top navigation bar.
3. Click the 'Session' tab.
4. Type the new value into the 'Session Timeout' field, then click the 'Update' button.

Screenshot: 'Session Options'



See Also...

3.6 Managing an Application's Session
4.4 Managing a Principal's Session

Related Topics

- 5.1 Viewing Crowd's System Information
- 5.2 Configuring Server Settings
  - 5.2.1 Licensing
  - 5.2.2 Deployment Title
  - 5.2.3 Domain
  - 5.2.4 Token Seed
  - 5.2.5 Session Timeout
  - 5.2.6 Caching
    - 5.2 Configuring Caching for an Application
- 5.3 Configuring SMTP Email
  - 5.3.1 Creating an Email Notification Template

- 5.4 Backing Up and Restoring Data
- 5.5 Logging

Crowd Documentation

5.2.6 Caching

This page last changed on Apr 02, 2007 by rosie@atlassian.com.

Caching is used to store run-time authentication and authorisation rules, which can be expensive to calculate.

It is recommended that caching be turned off during development cycles, and re-enabled for production use.

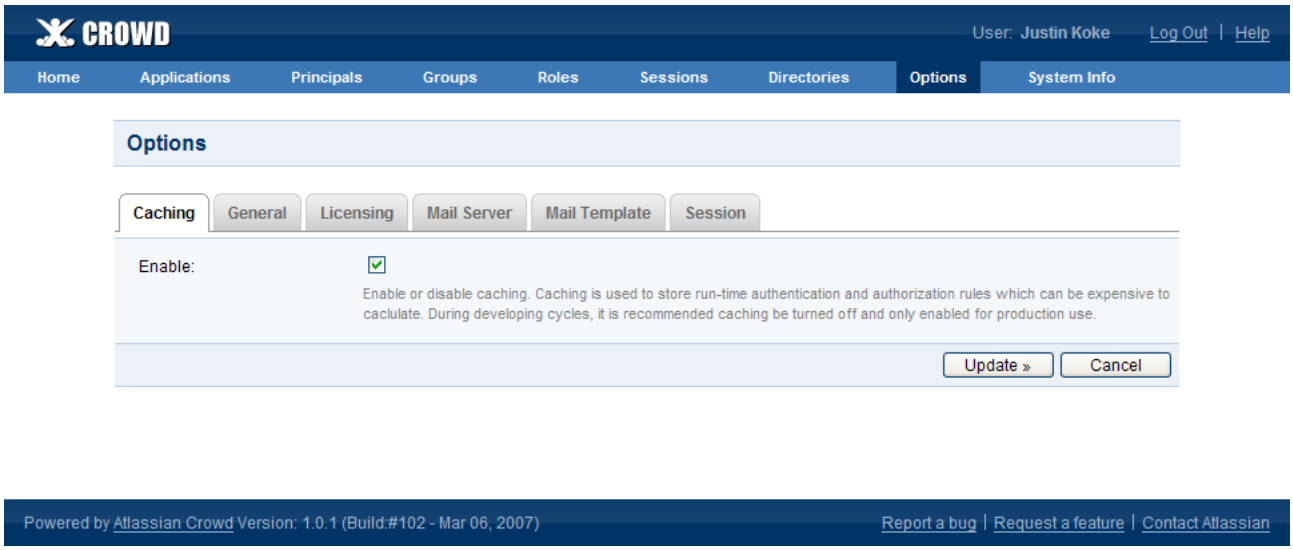
In Crowd, caching occurs in two main areas:

- The Crowd server itself — certain parts of the Crowd Administration Console application are stored in a local cache to improve performance.
- The applications that are connected to Crowd — e.g. JIRA, Confluence and Bamboo. These applications store users, groups and role data in a local cache, which helps improve the performance of Crowd since these applications do not have to repeatedly request information from Crowd. Generally it is not necessary to configure application caching, although this depends on the size of your application deployments. To fine-tune how the caching works for your Crowd application, please see 5.2 Configuring Caching for an Application.

To enable caching on the Crowd server,

1. Login to the Crowd Administration Console.
2. Click the 'Options' link in the top navigation bar.
3. Click the 'Caching' tab.
4. Select the 'Enable' check-box, then click the 'Update' button.

Screenshot: 'Caching'



Related Topics

- 5.1 Viewing Crowd's System Information
- 5.2 Configuring Server Settings
    - 5.2.1 Licensing
    - 5.2.2 Deployment Title
    - 5.2.3 Domain
    - 5.2.4 Token Seed
    - 5.2.5 Session Timeout
    - 5.2.6 Caching
        - 5.2 Configuring Caching for an Application
- 5.3 Configuring SMTP Email
    - 5.3.1 Creating an Email Notification Template
- 5.4 Backing Up and Restoring Data
- 5.5 Logging

Crowd Documentation

5.2 Configuring Caching for an Application

This page last changed on Apr 02, 2007 by rosie@atlassian.com.

In Crowd, caching occurs in two main areas:

- The Crowd server itself — certain parts of the Crowd Administration Console application are stored in a local cache to improve performance.
- The applications that are connected to Crowd — e.g. JIRA, Confluence and Bamboo. These applications store users, groups and role data in a local cache, which helps improve the performance of Crowd since these applications do not have to repeatedly request information from Crowd. Generally it is not necessary to configure application caching, although this depends on the size of your application deployments.

To enable server caching, please see 5.2.6 Caching.

To enable application caching,

- Edit the `crowd-ehcache.xml` file, which is located in the `WEB-INF/classes` directory of your application's Crowd Client. The two main properties are:
  - diskStore: If you have enabled disk persistence (`diskPersistent="true"`) this is the location on the file system where Ehcache will store its caching information. By default it uses `java.io.tmpdir` which is Java's default temporary file location.
  - defaultCache: This property has many configurable options. Please read the documentation provided by Ehcache to fully understand the implications and possibilities with this property. Some basic features are described below.

Below is a small snippet of the `crowd-ehcache.xml` file.

```
<ehcache>

    <diskStore path="java.io.tmpdir"/>

    <defaultCache
        maxElementsInMemory="50000"
        eternal="false"
        overflowToDisk="false"
        timeToIdleSeconds="300"
        timeToLiveSeconds="300"
        diskPersistent="false"
        diskExpiryThreadIntervalSeconds="120"/>

</ehcache>
```

Some basic features of defaultCache:

- eternal: This indicates that all elements in the cache will live for ever and that any time-outs will be ignored. It is strongly recommended that set this to false.
- timeToIdleSeconds: This sets the maximum amount of time between an element being accessed and it expiry. If you set this value to 0, the element will idle indefinitely.
- timeToLiveSeconds: This set the maximum time between creation time of an element and when it will expire. If you set this value to 0 it will live indefinitely.
- maxElementsInMemory: Sets the maximum number of elements that can be stored in the cache's

memory. If this limit is reached, the default caching strategy LRU (Least Recently Used) will be invoked and those elements will be removed.

Related Topics

- 5.1 Viewing Crowd's System Information
- 5.2 Configuring Server Settings
  - 5.2.1 Licensing
  - 5.2.2 Deployment Title
  - 5.2.3 Domain
  - 5.2.4 Token Seed
  - 5.2.5 Session Timeout
  - 5.2.6 Caching
    - 5.2 Configuring Caching for an Application
- 5.3 Configuring SMTP Email
  - 5.3.1 Creating an Email Notification Template
- 5.4 Backing Up and Restoring Data
- 5.5 Logging

Crowd Documentation

5.3 Configuring SMTP Email

This page last changed on Apr 02, 2007 by rosie@atlassian.com.

If SMTP email has been configured, Crowd can send email notifications to users during special events, such as when a user's password is reset or a server event occurs.

To configure SMTP email,

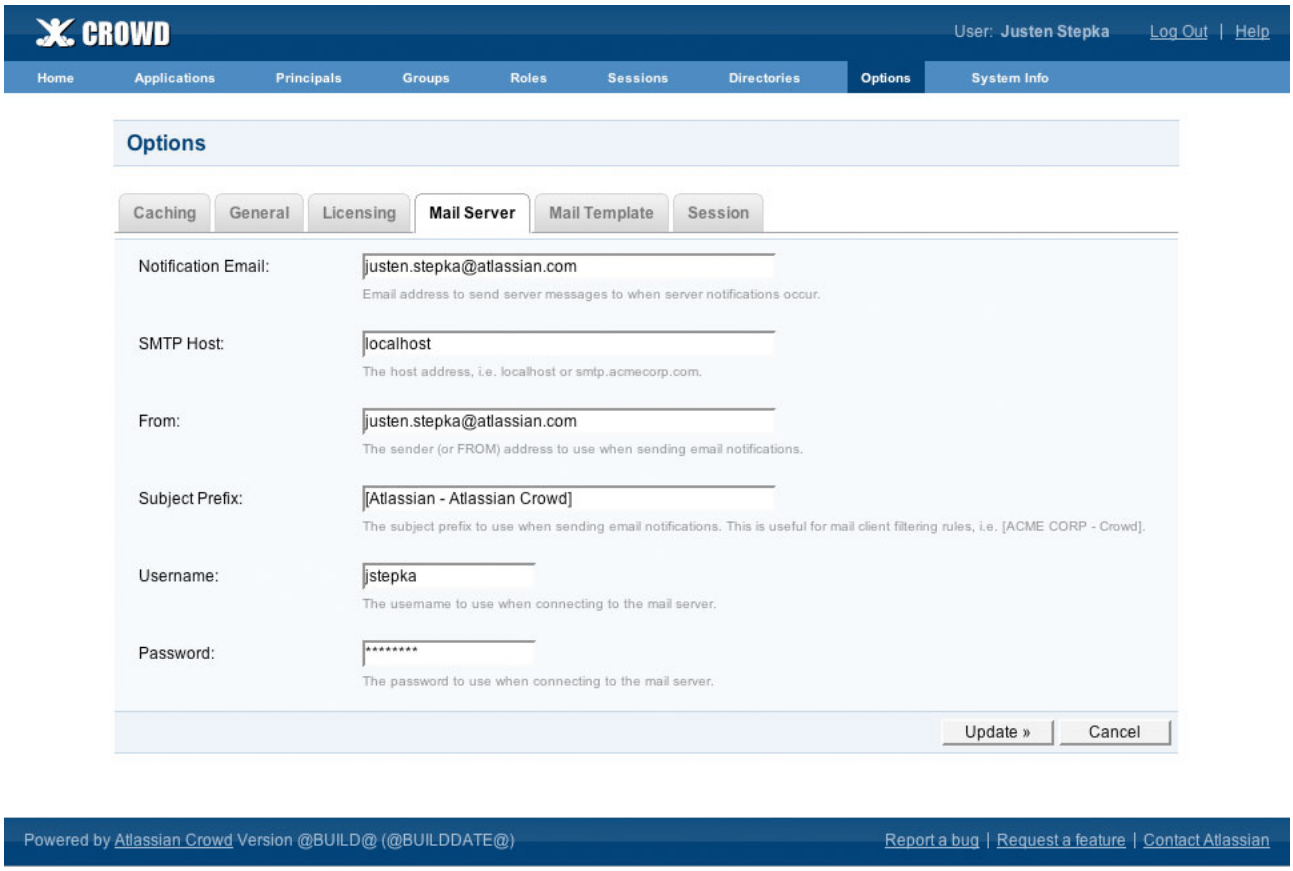
1. Login to the Crowd Administration Console.
2. Click the 'Options' link in the top navigation bar.
3. Click the 'Mail Server' tab.
4. Enter the details of your mail server, and the username and password (if required) that Crowd will use to log in to your mail server:

- Notification Email — The email address which will receive notifications about server events.
- SMTP Host — The hostname of the SMTP mail server, e.g. 'localhost' or 'smtp.acme.com'
- From — The email address from which password notifications will be sent to users.
- Subject Prefix — The prefix which will appear at the start of the email subject, for all emails generated by Crowd. This can be useful for email client programs that offer filtering rules.
- Username — The username that your Crowd server will use when it logs into your mail server.
- Password — The password that your Crowd server will use when it logs into your mail server.

Then Click the 'Update' button.

To customise the password notification message, please see 5.3.1 Creating an Email Notification Template

Screenshot: 'Mail Server'



Related Topics

- 5.1 Viewing Crowd's System Information
- 5.2 Configuring Server Settings
  - 5.2.1 Licensing
  - 5.2.2 Deployment Title
  - 5.2.3 Domain
  - 5.2.4 Token Seed
  - 5.2.5 Session Timeout
  - 5.2.6 Caching
    - 5.2 Configuring Caching for an Application
- 5.3 Configuring SMTP Email
  - 5.3.1 Creating an Email Notification Template
- 5.4 Backing Up and Restoring Data
- 5.5 Logging

Crowd Documentation

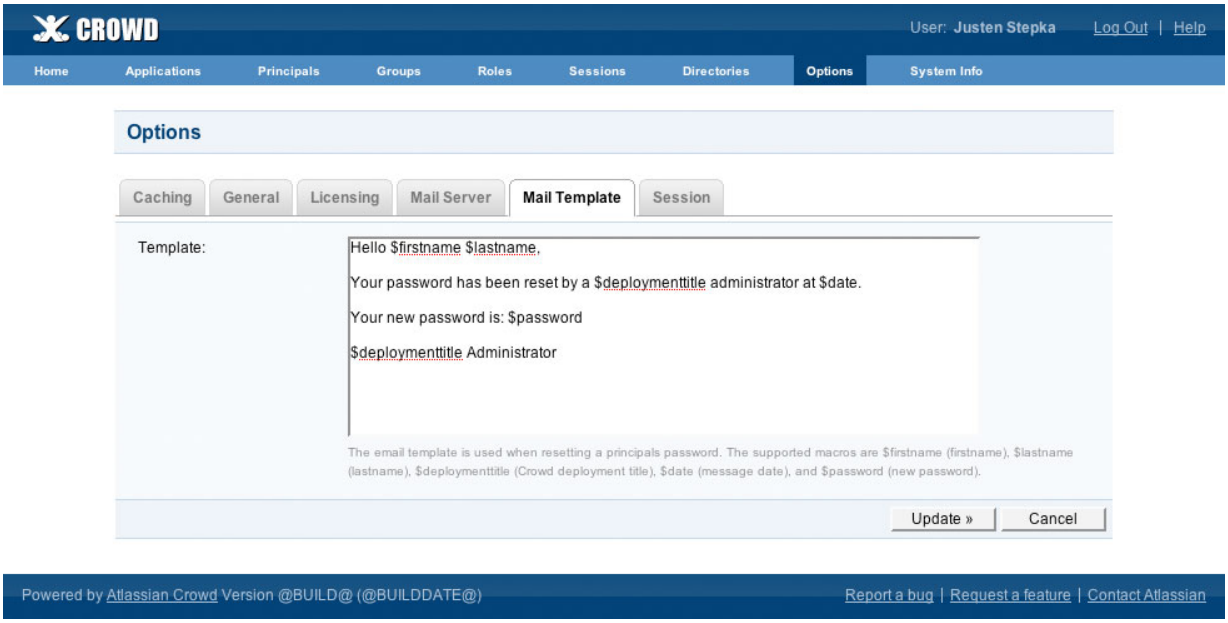
5.3.1 Creating an Email Notification Template

This page last changed on Apr 02, 2007 by rosie@atlassian.com.

Mail Template

The email template is used when sending a notification to a principal (user), e.g. when resetting a principal's password. The following template macros are available:

- $firstname: will be replaced by the principal's first name.
- $lastname: will be replaced by the principal's last name.
- $deploymenttitle: will be replaced by the title of your Crowd deployment, as defined in 5.2.3 Domain.
- $date: will be replaced by the time of the message event.
- $password: will be replaced by the principal's new password.



Related Topics

- 5.1 Viewing Crowd's System Information
- 5.2 Configuring Server Settings
  - 5.2.1 Licensing
  - 5.2.2 Deployment Title
  - 5.2.3 Domain
  - 5.2.4 Token Seed
  - 5.2.5 Session Timeout
  - 5.2.6 Caching
    - 5.2 Configuring Caching for an Application
- 5.3 Configuring SMTP Email
  - 5.3.1 Creating an Email Notification Template
- 5.4 Backing Up and Restoring Data
- 5.5 Logging

Crowd Documentation

5.4 Backing Up and Restoring Data

This page last changed on Apr 02, 2007 by justin.

You can backup your Crowd data to an XML file. This data includes:

- your Crowd server configuration details, including connection details for all your directories and applications.
- any internal directories that exist.

It is recommended that you backup your data regularly, especially after any significant configuration changes. It is also recommended that you perform regular backups of your database.

To backup your Crowd data,

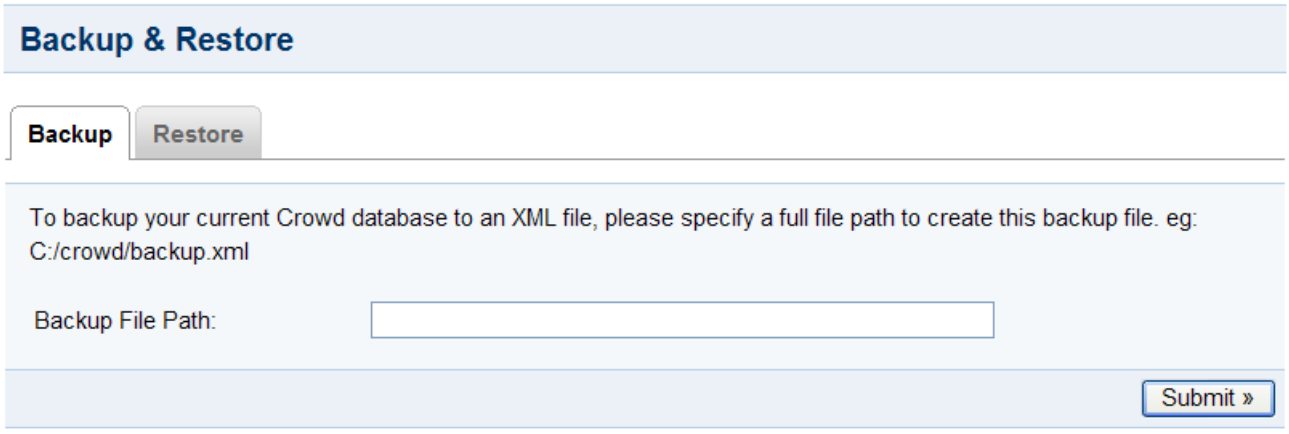
1. Login to the Crowd Administration Console.
2. Click the 'Backup & Restore' link in the top navigation bar.
3. Click the 'Backup' tab.
4. Type an appropriate 'File Backup Path', including the name of the XML file, then click the 'Submit' button.

To restore your Crowd data,

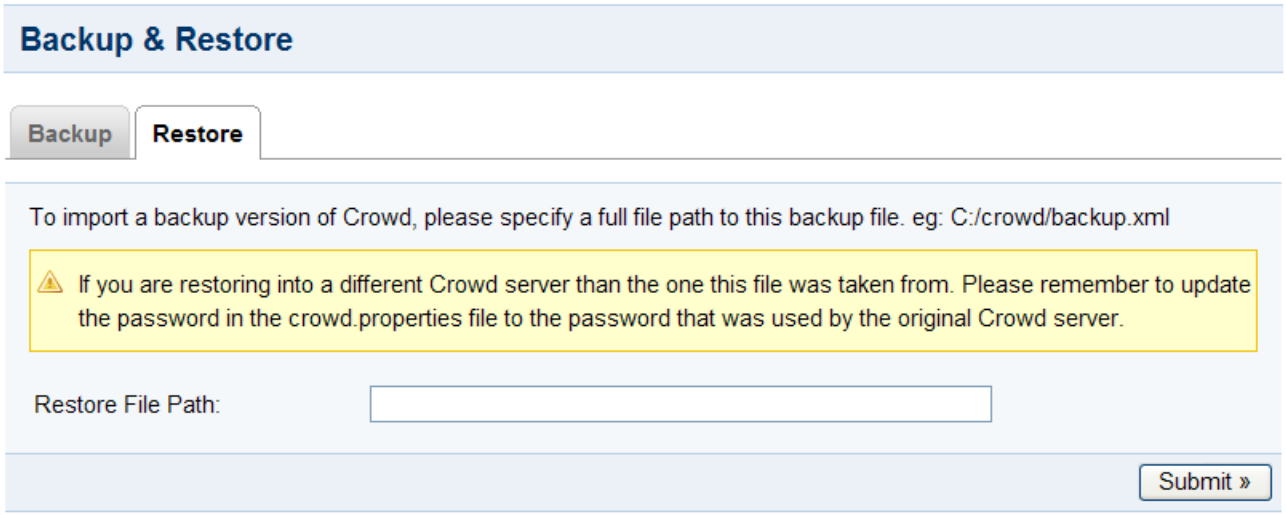
Before you begin: If you created the XML backup file on a different server, edit the `crowd.properties` file and change the password to match the password of the server on which you created the XML backup file.

1. Login to the Crowd Administration Console.
2. Click the 'Backup & Restore' link in the top navigation bar.
3. Click the 'Restore' tab.
4. In the 'Restore File Path' field, type the path to the XML file. Then click the 'Submit' button.

Screenshot 1: 'Backup'



Screenshot 2: 'Restore'



Related Topics

- 5.1 Viewing Crowd's System Information
- 5.2 Configuring Server Settings
  - 5.2.1 Licensing
  - 5.2.2 Deployment Title
  - 5.2.3 Domain
  - 5.2.4 Token Seed
  - 5.2.5 Session Timeout
  - 5.2.6 Caching
    - 5.2 Configuring Caching for an Application
- 5.3 Configuring SMTP Email
  - 5.3.1 Creating an Email Notification Template
- 5.4 Backing Up and Restoring Data
- 5.5 Logging

Crowd Documentation

5.5 Logging

This page last changed on Apr 02, 2007 by rosie@atlassian.com.

Logging

A common task when identifying Crowd problems is to turn up the log level. This section describes how to adjust the various Crowd log settings.

Background

Crowd's logging output is classified by importance, with the levels being:

- DEBUG: low-level details most people never need to know about.
- INFO: Informational messages on what Crowd is doing. Usually not interesting.
- WARN: Warnings that something may have gone wrong, or other messages a sysadmin may wish to know.
- ERROR: Something went wrong in Crowd. The person responsible for configuring Crowd should be notified.

The default level is WARN, meaning warnings and errors are displayed. Sometimes it is useful to adjust this level to see more details.

Change the Logging Level

1. With a text editor, open `crowd-webapp/WEB-INF/classes/log4j.properties`.
2. Adjust the output level to the expected level of importance listed above in the Background section.
3. Save the `log4j.properties` file.
4. Restart Crowd to have the new log settings take affect.

When diagnosing a server problem you need to adjust Crowd's package logging be:

```
log4j.logger.com.atlassian.crowd=DEBUG
```

XFire / Web Services Messages

Crowd has specific loggers that allow you to review the incoming and outgoing messages to the Crowd security framework. This is useful in debugging your applications or to monitor how much traffic is being used by an integrated application.

To turn on the XFire in and out logging handler, add uncomment the following lines in your `log4.properties` file:

```
# Uncomment the line below to have the Crowd server output the incoming SOAP request method and
parameters.
```

```
log4j.logger.com.atlassian.crowd.integration.service.soap.xfire.XFireOutLoggingMethodHandler=DEBUG

# Uncomment the line below to have the Crowd server output the outgoing SOAP request method and
parameters.
log4j.logger.com.atlassian.crowd.integration.service.soap.xfire.XFireInLoggingMethodHandler=DEBUG
```

Related Topics

- 5.1 Viewing Crowd's System Information
- 5.2 Configuring Server Settings
    - 5.2.1 Licensing
    - 5.2.2 Deployment Title
    - 5.2.3 Domain
    - 5.2.4 Token Seed
    - 5.2.5 Session Timeout
    - 5.2.6 Caching
        - 5.2 Configuring Caching for an Application
- 5.3 Configuring SMTP Email
    - 5.3.1 Creating an Email Notification Template
- 5.4 Backing Up and Restoring Data
- 5.5 Logging

Crowd Documentation

Crowd Development Hub

This page last changed on Mar 28, 2007 by rosie@atlassian.com.

- Creating a Crowd Client for your Custom Application
  - Application Integration Overview
    - Sample Application ('demo')
  - Java Integration Libraries
  - SOAP API
    - Axis Client Stub Generation
    - Microsoft .NET Client
- Creating a Custom Directory Connector

Creating a Crowd Client for your Custom Application

This page last changed on Apr 19, 2007 by rosie@atlassian.com.

Crowd allows your applications to authenticate users against Crowd's user directories.

Crowd ships with ready-made connectors ('Crowd Clients') for several popular applications (see 1.1.1 Supported Applications and Directories for the complete list). If you need to connect Crowd to one of these applications, please see 3. Managing Applications. If you need to connect Crowd to an application that is not listed, you can achieve this by creating a Crowd Client for your application, using the SOAP API.

Creating a Crowd Client

Crowd ships with Java Client Libraries which simplify the process of communicating with the SOAP API. If you have a Java application, you can use these libraries. If you are using a language other than Java (e.g. PHP, Ruby, etc), please use the SOAP API directly.

For assistance please see:

- Application Integration Overview
  - Sample Application ('demo')
- Java Integration Libraries
- SOAP API
  - Axis Client Stub Generation
  - Microsoft .NET Client

Next Steps:

After creating your Crowd Client, please see 3.2.7 Integrating Crowd with a Custom Application.

Related Topics

- Creating a Crowd Client for your Custom Application
  - Application Integration Overview
    - Sample Application ('demo')
  - Java Integration Libraries
  - SOAP API
    - Axis Client Stub Generation
    - Microsoft .NET Client
- Creating a Custom Directory Connector

Crowd Documentation

Application Integration Overview

This page last changed on Mar 29, 2007 by rosie@atlassian.com.

The Crowd framework allows an application to perform authentication and authorisation calls against a mapped directory, including:

- Authenticate a principal (i.e. a user).
- Validate and invalidate an existing principal authentication.
- Find a principal by their authentication token.
- Search principals, groups and roles by name or attributes
- Add principals, groups and roles.
- Validate a principal's group and role membership.
- Add and remove principals from groups and roles.
- Update a principal's attribute data.
- Update or reset a principal's authentication credentials.

Crowd's application provisioning allows an application to be mapped to multiple directories. When an application needs to authenticate or authorise a principal, Crowd will call the directory listed first. If the security call can be processed by the directory, the operation will then return the result. If the call cannot be processed, the next directory in the list will then be used when processing the security call until all directories have been exhausted. If the security call cannot be processed, an `Exception` (based on the method) will be thrown.

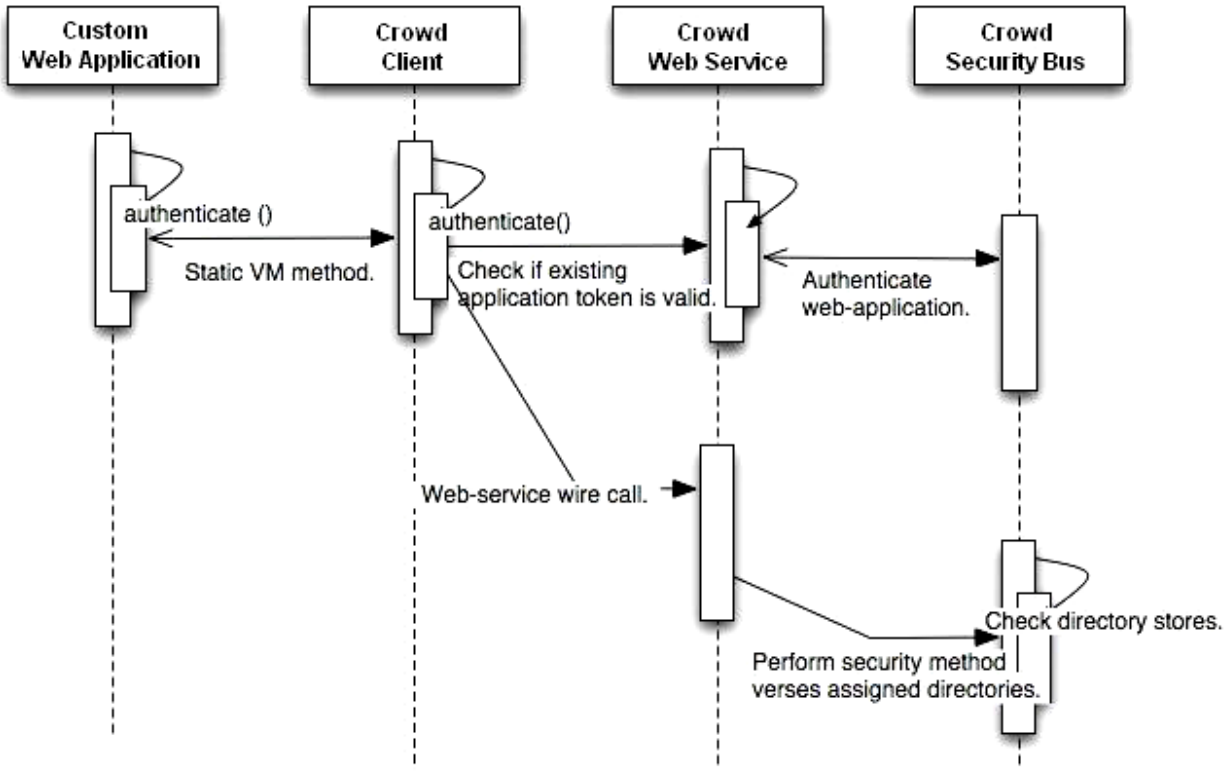
Integration Overview

When an application needs to perform a security request (that is, needs to authenticate or authorise a user) via Crowd's API, the following two steps need to occur:

1. The application authenticates itself with Crowd; the authentication token may be reused by the application during subsequent calls. During this step, Crowd validates the application's credentials and address against known application credentials/addresses.
2. Using the authenticated token from the previous step, the application then performs the security request for a particular user.

Should the application's requesting token become invalid, the client library will attempt to re-authenticate and perform the security request. If the second authentication request fails, an `Exception` will be thrown, specifying that the application's credentials are invalid.

Diagram — Application Authorisation Sequence:



Next Step

- If you are using the Java Integration Libraries, the application authorisation sequence above is fully handled by the supplied Java implementation.
- If you are using the SOAP interface, you will need to explicitly implement each step of the application authorisation sequence. As an example, please see the Microsoft .NET Client.

Related Topics

- Application Integration Overview
  - Sample Application ('demo')
- Java Integration Libraries
- SOAP API
  - Axis Client Stub Generation
  - Microsoft .NET Client

Crowd Documentation

Sample Application ('demo')

This page last changed on Apr 02, 2007 by justin.

To assist you when integrating your web applications, the entire sourcecode to the sample 'demo' application is included in the `src` folder of the Crowd download archive.

The 'demo' application highlights best practices when using the Crowd framework, and can be used as an example when integrating your own web applications.

To access the 'demo' application, go to http://localhost:8095/demo (assuming you installed it during setup).

Related Topics

- Application Integration Overview
  - Sample Application ('demo')
- Java Integration Libraries
- SOAP API
  - Axis Client Stub Generation
  - Microsoft .NET Client

Crowd Documentation

Java Integration Libraries

This page last changed on May 20, 2007 by rosie@atlassian.com.

This page provides sample code for creating a Crowd Client using the supplied Java Integration Libraries.

SecurityServerClient

The `SecurityServerClient` is useful for common create, update and delete operations for principals, groups and roles. To accomplish this, the `SecurityServerClient` maps 1-to-1 with the SOAP API of the Crowd server. The class works by first reading in the `crowd.properties` configuration file from your application's class path, once the `crowd.properties` configuration values have been parsed. The client will then authenticate the application with the Crowd security server for future SOAP requests.

A full list of the available methods for the `SecurityServerClient` is available here:

- http://docs.atlassian.com/crowd/current/com/atlassian/crowd/integration/service/soap/server/SecurityServer.htr

HttpAuthenticator

The `HttpAuthenticator` simplifies the authentication of HTTP based clients. When an authentication or invalidation is performed, the `HttpAuthenticator` manages the setting and resetting of integration variables for the principal's HTTP session. If the application has little need beyond authentication and validation, the `HttpAuthenticator` is a simple and very straightforward integration piece. Shown below is a code example of authenticating and logging off a principal:

Example 1:

```
HttpAuthenticator.authenticate(request, response, username, password);
```

Example 2:

```
HttpAuthenticator.authenticate(request, response);
```

If there were any issues with the authentication or logoff calls, an `Exception` will be thrown to the application.

The `HttpAuthenticator` manages the following:

- Authenticating an HTTP request, and setting the session with the correct attributes for other integration points of the IDX framework.
- Invalidating an HTTP request includes removing session related attributes.
- Obtaining a principal's authenticated token from a session or browser cookie.
- Validating an existing HTTP authentication for single sign-on. If another application in the same domain has already authenticated the principal, the HttpAuthenticator will attempt to validate the existing authentication.

- Building a standard AuthenticationContext for a principal. This can be used to assure the authentication is consistent across all applications when setting validation factors of the application client.

VerifyTokenFilter

The `VerifyTokenFilter` is an HTTP servlet filter that protects secured resources by verifying the session or cookie token is active and the principal has access to the requesting application. The token filter works in conjunction with the HttpAuthenticator validating and setting various session and cookie attributes. Should the principal's token become expired or invalid due to security restrictions, the principal will be redirected to the URL provided by the `crowd.properties`.

Using the token filter is very straight forward, simply edit your `web.xml` deployment descriptor to reflect the filter and desired resource mapping:

```
<filter>
    <filter-name>VerifyTokenFilter</filter-name>
    <filter-class>com.atlassian.crowd.integration.http.VerifyTokenFilter</filter-class>
</filter>

<filter-mapping>
    <filter-name>VerifyTokenFilter</filter-name>
    <url-pattern>/secure/*</url-pattern>
</filter-mapping>
```

In this example, the verify token filter will prevent any pages on the /secure/ path from being accessed unless a valid token is found.

Should the token expire or be found invalid, the original URL will be stored in the principal's session at a String with the key of `VerifyTokenFilter.ORIGINAL_URL`. This is useful because, when the principal later authenticates, the original URL and parameters can then be used as a redirect bringing the principal back to their original `POST`. An example of how this can be accomplished at login is shown below:

```
HttpAuthenticator.authenticate(request, response, username, password);

// Check if principal was requesting a page that was prevented, if so, redirect.
String requestingPage = (String) getSession().getAttribute(VerifyTokenFilter.ORIGINAL_URL);

if (requestingPage != null) {
    // redirect the principal to the requesting page
    response().sendRedirect(requestingPage);

} else {
    // return the to the login page
    return SUCCESS;
}
```

Related Topics

- Application Integration Overview
  - Sample Application ('demo')
- Java Integration Libraries
- SOAP API

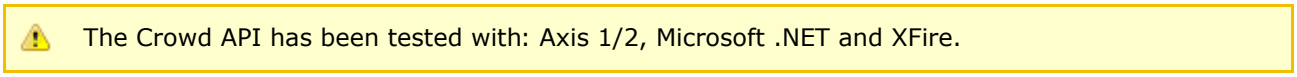
- Axis Client Stub Generation
- Microsoft .NET Client

Crowd Documentation

SOAP API

This page last changed on May 16, 2007 by justen.stepka@atlassian.com.

This page provides sample code for creating a Crowd Client using the SOAP API.



The SOAP WSDL is available on the following URL for the Crowd Standalone version:

- http://localhost:8080/crowd/services/SecurityServer?wsdl

The Java Remote Interface that is used to generate the SOAP service is available here:

- http://docs.atlassian.com/crowd/current/com/atlassian/crowd/integration/service/soap/server/SecurityServer.htr

This JavaDoc file details inputs and outputs for the available Crowd security server SOAP server. You will see that all methods require an `AuthenticatedToken`. A valid token can be obtained by calling the `authenticateApplication` service method.

Like a user token, the application client token is valid only for the same period of time a user token would be. If you receive a SOAP fault for an invalid application client you will need to re-authenticate your application client and re-invoke the SOAP service.

Crowd ships with out of the box Java Integration Libraries that map one-to-one to these web services.

authenticateApplication - Authenticating an Application Client

Here is the server request which passes in the server name and a password credential.

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <soap:Body>
                <authenticateApplication xmlns="urn:SecurityServer">
                        <in0>
                                <credential
xmlns="http://authentication.integration.crowd.atlassian.com">
                                        <credential>password</credential>
                                </credential>

                                <name
xmlns="http://authentication.integration.crowd.atlassian.com">jira</name>

                                <validationFactors
xmlns="http://authentication.integration.crowd.atlassian.com" xsi:nil="true" />
                        </in0>
                </authenticateApplication>
        </soap:Body>
</soap:Envelope>
```

The server will respond with an application token:

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <soap:Body>
                <authenticateApplicationResponse xmlns="urn:SecurityServer">
                        <out>
                                <name
xmlns="http://authentication.integration.crowd.atlassian.com">jira</name>
                                <token
xmlns="http://authentication.integration.crowd.atlassian.com">9vN5haaWY+xGBs3XitgAIg==</token>
                        </out>
                </authenticateApplicationResponse>
        </soap:Body>
</soap:Envelope>
```

authenticatePrincipal - Authenticating an Principal

In this message the principal is authenticated using the previously obtained application token.

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <soap:Body>
                <authenticatePrincipal xmlns="urn:SecurityServer">
                        <in0>
                                <name
xmlns="http://authentication.integration.crowd.atlassian.com">jive</name>
                                <token
xmlns="http://authentication.integration.crowd.atlassian.com">9vN5haaWY+xGBs3XitgAIg==</token>
                        </in0>
                        <in1>
                                <application
xmlns="http://authentication.integration.crowd.atlassian.com">jive</application>
                                <credential
xmlns="http://authentication.integration.crowd.atlassian.com">
                                        <credential>password</credential>
                                </credential>
                                <name
xmlns="http://authentication.integration.crowd.atlassian.com">jstepka</name>
                                <validationFactors
xmlns="http://authentication.integration.crowd.atlassian.com" />
                        </in1>
                </authenticatePrincipal>
        </soap:Body>
</soap:Envelope>
```

The server then responds back with the token for the now authenticated user:

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <soap:Body>
                <authenticatePrincipalResponse xmlns="urn:SecurityServer">
                        <out>o7MSozJJbKQttOLvC4hN2w==</out>
                </authenticatePrincipalResponse>
        </soap:Body>
</soap:Envelope>
```

An invalid authentication attempt will look like the following:

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
```

```
                xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soap:Body>
        <soap:Fault>
            <faultcode>soap:Server</faultcode>
            <faultstring>Fault:
com.atlassian.crowd.integration.exception.InvalidAuthenticationException</faultstring>
            <detail>
                <InvalidAuthenticationException xmlns="urn:SecurityServer"/>
            </detail>
        </soap:Fault>
    </soap:Body>
</soap:Envelope>
```

findPrincipalByToken - Finding a Principal by their Authenticated Token

Now that the principal is authenticated, we may want to find additional details about the principal. With the authenticated principal token, the application can now lookup a user by a token or their name. The example below shows looking up a principal by their authenticated token:

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <soap:Body>
                <findPrincipalByName xmlns="urn:SecurityServer">
                        <in0>
                                <name
xmlns="http://authentication.integration.crowd.atlassian.com">jive</name>
                                <token
xmlns="http://authentication.integration.crowd.atlassian.com">9vN5haaWY+xGBs3XitgAIg==</token>
                        </in0>
                        <in1>jstepka</in1>
                </findPrincipalByName>
        </soap:Body>
</soap:Envelope>
```

The server lookup response for the principal token:

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soap:Body>
        <findPrincipalByNameResponse xmlns="urn:SecurityServer">
            <out>
                <ID xmlns="http://soap.integration.crowd.atlassian.com">-1</ID>
                <active xmlns="http://soap.integration.crowd.atlassian.com">true</active>
                <attributes xmlns="http://soap.integration.crowd.atlassian.com">
                    <SOAPAttribute>
                        <name>sn</name>
                        <values>
                            <ns1:string xmlns:ns1="urn:SecurityServer">Stepka</ns1:string>
                        </values>
                    </SOAPAttribute>
                    <SOAPAttribute>
                        <name>invalidPasswordAttempts</name>
                        <values>
                            <ns1:string xmlns:ns1="urn:SecurityServer">0</ns1:string>
                        </values>
                    </SOAPAttribute>
                    <SOAPAttribute>
                        <name>requiresPasswordChange</name>
                        <values>
                            <ns1:string xmlns:ns1="urn:SecurityServer">false</ns1:string>
                        </values>
                    </SOAPAttribute>
```

```
                    <SOAPAttribute>
                        <name>mail</name>
                        <values>
                            <ns1:string
xmlns:ns1="urn:SecurityServer">justen.stepka@atlassian.com</ns1:string>
                        </values>
                    </SOAPAttribute>
                    <SOAPAttribute>
                        <name>lastAuthenticated</name>
                        <values>
                            <ns1:string
xmlns:ns1="urn:SecurityServer">1169440408520</ns1:string>
                        </values>
                    </SOAPAttribute>
                    <SOAPAttribute>
                        <name>givenName</name>
                        <values>
                            <ns1:string xmlns:ns1="urn:SecurityServer">Justen</ns1:string>
                        </values>
                    </SOAPAttribute>
                    <SOAPAttribute>
                        <name>passwordLastChanged</name>
                        <values>
                            <ns1:string
xmlns:ns1="urn:SecurityServer">1168995491407</ns1:string>
                        </values>
                    </SOAPAttribute>
                </attributes>
                <conception
xmlns="http://soap.integration.crowd.atlassian.com">2007-01-17T11:58:11+11:00</conception>
                <description xmlns="http://soap.integration.crowd.atlassian.com"
xsi:nil="true"/>
                <directoryID
xmlns="http://soap.integration.crowd.atlassian.com">1</directoryID>
                <lastModified
xmlns="http://soap.integration.crowd.atlassian.com">2007-01-17T18:38:51+11:00
                </lastModified>
                <name xmlns="http://soap.integration.crowd.atlassian.com">jstepka</name>
            </out>
        </findPrincipalByNameResponse>
    </soap:Body>
</soap:Envelope>
```

Related Topics

- Application Integration Overview
  - Sample Application ('demo')
- Java Integration Libraries
- SOAP API
  - Axis Client Stub Generation
  - Microsoft .NET Client

Crowd Documentation

Axis Client Stub Generation

This page last changed on Apr 25, 2007 by justen.stepka@atlassian.com.

Generating client stubs with Axis can be accomplished by running this command:

```
java -cp "$AXISCLASSPATH" org.apache.axis.wsdl.WSDL2Java
http://localhost:8095/crowd/services/SecurityServer?wsdl
```

When the necessary objects are created off the Crowd server WSDL, you will end up with a directory structure similar to this:

```
drwxr-xr-x   6 jstepka  jstepka  204 Apr 19 16:56 SecurityServer_pkg
drwxr-xr-x   3 jstepka  jstepka  102 Apr 19 16:55 com
drwxr-xr-x   4 jstepka  jstepka  136 Apr 19 17:05 java
```

When you attempt to compile the generated class files, you will end up with a compilation error similar to the following:

```
java/rmi/RemoteException.java:[10,7] cyclic inheritance involving java.rmi.RemoteException

java/rmi/RemoteException.java:[11,32] modifier private not allowed here

java/rmi/RemoteException.java:[12,29] modifier private not allowed here

java/rmi/RemoteException.java:[64,29] modifier private not allowed here

java/rmi/RemoteException.java:[86,20] modifier private not allowed here

java/rmi/RemoteException.java:[104,56] modifier private static not allowed here

com/atlassian/crowd/integration/exception/InvalidCredentialException.java:[26,30] incompatible
types
```

To resolve these compile errors you will need to delete the generated `java` package.

The security server can then be used as below:

```
// connect to the crowd server, using the supplied service URL, similar to
http://localhost:8095/crowd/services/SecurityServer?wsdl
SecurityServerLocator secServer = new SecurityServerLocator();
secServer.setSecurityServerHttpPortEndpointAddress(serviceURL);

// obtain a reference to the SOAP service, which axis manages.
SecurityServerHttpBindingStub stub = (SecurityServerHttpBindingStub)
secServer.getSecurityServerHttpPort();

// authenticate the integrated crowd application
savedAppToken = stub.authenticateApplication(new ApplicationAuthenticationContext(
        new PasswordCredential(appPassword),
        appName,
        new ValidationFactor[0]));

// do your custom calls here
...
```

Microsoft .NET Client

This page last changed on Mar 28, 2007 by rosie@atlassian.com.

You will need to create a .NET proxy to the SOAP API, as follows:

1. Open a Microsoft Visual Studio .NET Command Prompt.

1. Run the following command to generate a proxy class (change the location of the WSDL according to your installation):

```
wsdl /l:CS /protocol:SOAP http://localhost:8080/crowd/services/SecurityServer?wsdl
```

   (Note: Ignore any schema validation warnings returned here)

2. Compile the generated class with the following references:

```
csc /t:library /r:System.Web.Services.dll /r:System.Xml.dll SecurityServer.cs
```

   This should generate a .NET assembly called `SecurityServer.DLL`.

When creating your .NET client application, remember to add a reference to this proxy. You will also need to add a reference to `System.Web.Services.DLL`.

The sample code calls methods from the proxy to perform authentication in a sample Crowd Application. Change the constants at the top of the code relevant to any Application you have previously set-up in Crowd.

Related Topics

- Application Integration Overview
  - Sample Application ('demo')
- Java Integration Libraries
- SOAP API
  - Axis Client Stub Generation
  - Microsoft .NET Client

Crowd Documentation

Creating a Custom Directory Connector

This page last changed on Mar 29, 2007 by rosie@atlassian.com.

If your directory is not listed in 1.1.1 Supported Applications and Directories then you will need to create your own custom directory connector.
Custom directory connectors allow developers to connect Crowd to custom user-stores, such as existing databases or legacy system.

Full Javadoc for the `RemoteDirectory` interface can be found here: http://docs.atlassian.com/crowd/current/com/atlassian/crowd/integration/directory/RemoteDirectory.html.

Next Steps:

After creating your directory connector, please see 2.2.3 Configuring a Custom Directory Connector.

Related Topics

- Creating a Crowd Client for your Custom Application
  - Application Integration Overview
    - Sample Application ('demo')
  - Java Integration Libraries
  - SOAP API
    - Axis Client Stub Generation
    - Microsoft .NET Client
- Creating a Custom Directory Connector

Crowd Documentation

Crowd Installation & Upgrade Guide

This page last changed on Mar 12, 2007 by rosie@atlassian.com.

- Crowd Release Notes
- Installing Crowd
- Upgrading Crowd

Crowd Release Notes

This page last changed on Apr 10, 2007 by rosie@atlassian.com.

Crowd 1.0.7 has now been released — see the Crowd 1.0.7 Release Notes

Installation

Information for installing Crowd can be found here. If upgrading from a previous version, please follow the Upgrade Guide.

Crowd Release Notes

- Crowd 1.0.7 Release Notes
- Crowd 1.0.6 Release Notes
- Crowd 1.0.5 Release Notes
- Crowd 1.0.4 Release Notes
- Crowd 1.0.3 Release Notes
- Crowd 1.0.2 Release Notes
- Crowd 1.0.1 Release Notes
- Crowd 1.0.0 Release Notes
- Crowd 0.4 Beta Release Notes
- Crowd 0.4.5 Beta Release Notes
- Crowd 0.4.4 Beta Release Notes
- Crowd 0.4.3 Beta Release Notes
- Crowd 0.4.2 Beta Release Notes
- Crowd 0.4.1 Beta Release Notes
- Crowd 0.3 Beta Release Notes
- Crowd 0.3.3 Beta Release Notes
- Crowd 0.3.2 Beta Release Notes
- Crowd 0.2 Beta Release Notes
- newreleaseCrowd

__newreleaseCrowd

This page last changed on May 22, 2007 by rosie@atlassian.com.

Crowd 1.0.7 has now been released — see the Crowd 1.0.7 Release Notes

Crowd 0.2 Beta Release Notes

This page last changed on Mar 11, 2007 by rosie@atlassian.com.

Crowd 1.0.7 has now been released — see the Crowd 1.0.7 Release Notes

Crowd 0.2

- Standalone version - Tomcat 5.5 with HSQL - .zip (59.5Mbs)
- Standalone version - Tomcat 5.5 with HSQL - .tar.gz (59.7Mbs)

Points of Interest

- There is an error when unzipping on the Windows platform, the archive integrity is fine and this will be fixed for the 0.3 release.
- The focus of this distribution is for JIRA and Confluence integration. Performance enhancements will be added for the 0.3 release which will allow large user-databases to be integrated.

Crowd 0.3 Beta Release Notes

This page last changed on Mar 11, 2007 by rosie@atlassian.com.

Crowd 1.0.7 has now been released — see the Crowd 1.0.7 Release Notes

Crowd 0.3

- Standalone version - Tomcat 5.5 with HSQL - .zip (65.3 Mbs)
- Standalone version - Tomcat 5.5 with HSQL - .tar.gz (64.7 Mbs)

Points of Interest

- The focus of this distribution is on performance for a large number of users and groups when integrating JIRA, Confluence and Bamboo integration.

Crowd 0.3.2 Beta Release Notes

This page last changed on Mar 11, 2007 by rosie@atlassian.com.

Crowd 1.0.7 has now been released — see the Crowd 1.0.7 Release Notes

The Crowd development team has released a new version of Crowd - 0.3.2.

This release addresses a Seraph SSO issue when integrating JIRA, Confluence and Bamboo.

http://jira.atlassian.com/secure/IssueNavigator.jspa?reset=true&pid=11291&fixfor=12540

You can now download Crowd from http://www.atlassian.com/Crowd

Cheers,

The Atlassian Crowd Development Team

Crowd 0.3.3 Beta Release Notes

This page last changed on Mar 11, 2007 by rosie@atlassian.com.

Crowd 1.0.7 has now been released — see the Crowd 1.0.7 Release Notes

The Crowd development team has released a new version of Crowd - 0.3.3.

This release addresses the following:

- Upgrade from Webwork 1 to Webwork 2
- Workaround for Active Directory to support CN forwards.

CRITICAL POSTGRES UPGRADE NOTES: http://jira.atlassian.com/browse/CWD-71

We started testing on IE7 and have noticed the CSS bugs and will work to get this addressed for the next build.

http://jira.atlassian.com/secure/IssueNavigator.jspa?reset=true&pid=11291&fixfor=12544

You can now download Crowd from http://www.atlassian.com/Crowd

Cheers,

The Atlassian Crowd Development Team

Crowd 0.4 Beta Release Notes

This page last changed on Mar 11, 2007 by rosie@atlassian.com.

Crowd 1.0.7 has now been released — see the Crowd 1.0.7 Release Notes

The Crowd development team has released a new version of Crowd - 0.4.

This release addresses several critical issues:

- Seraph Logout code fails to logout the user in Confluence, Bamboo and JIRA.
- Unable to search for a Principal by email address.
- Accept header authentication factor unreliable with Mozilla based browsers.
- Default 'localhost' configuration not added valid IP address of 127.0.0.1.

New features include:

- Allow all to authenticate.
- New LDAP connectors build off Spring LDAP Template with better performance enhancements.
- Support for LDAP filters

All Postgres DB will need to have the following command ran:

```
alter table "APPLICATIONDIRECTORIES" add column "ALLOWALLTOAUTHENTICATE" boolean;
```

http://jira.atlassian.com/secure/IssueNavigator.jspa?reset=true&pid=11291&fixfor=12266

You can now download Crowd from http://www.atlassian.com/Crowd

Cheers,

The Atlassian Crowd Development Team

Crowd 0.4.1 Beta Release Notes

This page last changed on Mar 11, 2007 by rosie@atlassian.com.

Crowd 1.0.7 has now been released — see the Crowd 1.0.7 Release Notes

The Crowd development team has released a new version of Crowd - 0.4.1.

This addresses bugs which can be viewed through our JIRA issue tracker:

http://jira.atlassian.com/secure/IssueNavigator.jspa?reset=true&pid=11291&fixfor=12600

You can now download Crowd from http://www.atlassian.com/Crowd

Cheers,

The Atlassian Crowd Development Team

Crowd 0.4.2 Beta Release Notes

This page last changed on Mar 11, 2007 by rosie@atlassian.com.

Crowd 1.0.7 has now been released — see the Crowd 1.0.7 Release Notes

The Crowd development team has released a new version of Crowd - 0.4.2.

This addresses bugs which can be viewed through our JIRA issue tracker:

http://jira.atlassian.com/secure/IssueNavigator.jspa?reset=true&pid=11291&fixfor=12623

You can now download Crowd from http://www.atlassian.com/Crowd

Cheers,

The Atlassian Crowd Development Team

Crowd 0.4.3 Beta Release Notes

This page last changed on Mar 11, 2007 by rosie@atlassian.com.

Crowd 1.0.7 has now been released — see the Crowd 1.0.7 Release Notes

The Crowd development team has released a new version of Crowd - 0.4.3.

This addresses bugs which can be viewed through our JIRA issue tracker:

http://jira.atlassian.com/secure/IssueNavigator.jspa?reset=true&pid=11291&fixfor=12267

- Support for AD when there are more than 999 records in a search result.
- Reduced the number of necessary libs for a client application.
- Improved the 'build.properties' file configuration.

You can now download Crowd from http://www.atlassian.com/Crowd

Cheers,

The Atlassian Crowd Development Team

Crowd 0.4.4 Beta Release Notes

This page last changed on Mar 11, 2007 by rosie@atlassian.com.

Crowd 1.0.7 has now been released — see the Crowd 1.0.7 Release Notes

The Crowd development team has released a new version of Crowd - 0.4.4.

This addresses bugs which can be viewed through our JIRA issue tracker:

http://jira.atlassian.com/secure/IssueNavigator.jspa?reset=true&pid=11291&fixfor=12642

- Caching improvement for Confluence.
- Removed an additional attribute that was causing integration problems with SOAP services when using Active Directory.

You can now download Crowd from http://www.atlassian.com/Crowd

Cheers,

The Atlassian Crowd Development Team

Crowd 0.4.5 Beta Release Notes

This page last changed on Mar 11, 2007 by rosie@atlassian.com.

Crowd 1.0.7 has now been released — see the Crowd 1.0.7 Release Notes

The Crowd development team has released a new version of Crowd - 0.4.5.

This addresses bugs which can be viewed through our JIRA issue tracker:

http://jira.atlassian.com/secure/IssueNavigator.jspa?reset=true&pid=11291&fixfor=12652

- Improved Active Directory LDAP attribute filtering.
- UI improvements with new screen layouts.
- Spring TX management.

You can now download Crowd from http://www.atlassian.com/Crowd

Cheers,

The Atlassian Crowd Development Team

Crowd 1.0.0 Release Notes

This page last changed on Mar 12, 2007 by rosie@atlassian.com.

Crowd 1.0.7 has now been released — see the Crowd 1.0.7 Release Notes

The Crowd development team has released Crowd 1.0.

This addresses bugs which can be viewed through our JIRA issue tracker:

- UI improvements with new screen layouts.
- Import and Export process for XML.
- LDAP Fixes for OpenLDAP and Microsoft Active Directory.
- Improved error reporting.
- Apache / Subversion support.

You can now download Crowd from http://www.atlassian.com/Crowd. If upgrading from a previous version, please follow the Upgrade Guide.

Atlassian JIRA (10 issues)			
Key	Summary	Pr	Status
CWD-173	Implement an import and export function in Crowd		Closed
CWD-188	License update (when invalid) page should detail current license details.		Closed
CWD-184	Make Crowd's internal exception extend NestableException from commons-lang		Closed
CWD-180	Schema violation with LDAP and Groups/Roles		Closed
CWD-178	LDAP flags are incorrect for Active Directory/LDAP (Win2k3 domain)		Closed
CWD-150	Build fails		Closed
CWD-101	Unable to upgrade from 0.2 to 0.3.3		Closed
CWD-97	Apache mod Crowd integration		Closed
CWD-90	sso support for fisheye		Closed
CWD-62	500 page.		Closed

Cheers,

The Atlassian Crowd Development Team

Crowd 1.0.1 Release Notes

This page last changed on Mar 12, 2007 by rosie@atlassian.com.

Crowd 1.0.7 has now been released — see the Crowd 1.0.7 Release Notes

The Crowd development team has released Crowd 1.0.1.

This addresses 3 critical bugs which can be viewed through our JIRA issue tracker:

- Create new group/role broken using OpenLDAP.
- XFireFault exception: "No write method for property".
- Single sign on Seraph authentication fails when the host on a domain is not the same.

You can now download Crowd from http://www.atlassian.com/Crowd

Atlassian JIRA (3 issues)			
Key	Summary	Pr	Status
CWD-190	XFireFault exception: "No write method for property".		 Closed
CWD-189	Create new group/role broken using OpenLDAP		 Closed
CWD-82	Single sign on Seraph authentication fails when the host on a domain is not the same.		 Closed

Cheers,

The Atlassian Crowd Development Team

Crowd 1.0.2 Release Notes

This page last changed on Mar 22, 2007 by rosie@atlassian.com.

Crowd 1.0.7 has now been released — see the Crowd 1.0.7 Release Notes

The Crowd development team has released Crowd 1.0.2.

This addresses bugs and feature improvements which can be viewed through our JIRA issue tracker:

- Included missing libraries for build archive.
- Added logging for input and output operations on SOAP services.
- Improved Jira caching for Crowd data.
- Added support for SSO beyond centralised authentication for Jive Forums.

You can now download Crowd from http://www.atlassian.com/Crowd

Atlassian JIRA (6 issues)			
Key	Summary	Pr	Status
CWD-199	Missing libraries from the Crowd distribution		 Closed
CWD-198	I renamed the docs from "Documentation" to "Crowd Documentation" (sorry). Can you please fix the "Help link?		 Closed
CWD-197	XFire service input and output logging.		 Closed
CWD-196	Improve the ability to configure the internal cache's used by the Crowd client and the Crowd console		 Closed
CWD-195	Implement SSO for Jive Forums		 Closed
CWD-193	Download archive is missing wsdl4j-1.5.2.jar		 Closed

Cheers,

The Atlassian Crowd Development Team

Crowd 1.0.3 Release Notes

This page last changed on Apr 02, 2007 by rosie@atlassian.com.

 Crowd 1.0.7 has now been released — see the Crowd 1.0.7 Release Notes

The Crowd development team has released Crowd 1.0.3.

This build is a mix of new features, bugs fixes and feature improvements:

- Improved SSO integration with Seraph for JIRA, Confluence and Bamboo.
- First builds of Apache Directory Server connector.
- Now supports directory server version that do not have the paged ldap control.
- Documentation updates.

You can now download Crowd from http://www.atlassian.com/Crowd

Atlassian JIRA (8 issues)			
Key	Summary	Pr	Status
CWD-218	When an application is searching for its members from an LDAP repo AND an Internal Directory a HibernateException is thrown around trying to persist elements in a RemoteGroup.members		 Closed
CWD-216	Crowd session token should be unique for each user, directory, machine		 Closed
CWD-214	Login should logout any previous logged in users before a new login		 Closed
CWD-179	Paged results control option for LDAP connectors.		 Closed
CWD-177	Fisheye connector logs unnecessary exception.		 Closed
CWD-175	Computers show up in the Principal list within Crowd from MSAD		 Closed
CWD-169	NullPointerException on add OpenLDAP directory		 Closed
CWD-121	Setting a "Remember Me" flag in Confluence, JIRA or Bamboo does not work, since the Token Reaper 'reaps' all session when the timeout is reached		 Closed

Cheers,

The Atlassian Crowd Development Team

Crowd 1.0.4 Release Notes

This page last changed on Apr 11, 2007 by rosie@atlassian.com.

Crowd 1.0.7 has now been released — see the Crowd 1.0.7 Release Notes

The Crowd development team has released Crowd 1.0.4.

This build focused on bug fixes:

- Import export process was failing with Oracle DB.
- Implemented updating known attribute types on an LDAP object..
- Importing JIRA users is fixed for MySQL on a Unix like filesystem.

You can now download Crowd from http://www.atlassian.com/Crowd

Atlassian JIRA (6 issues)			
Key	Summary	Pr	Status
CWD-221	Add documentation (marketing) section for Apache Directory Server		Closed
CWD-220	Implement RemoteDirectory updatePrincipal(RemotePrincipal) method for LDAP servers using InetOrgPerson as the Principal object.		Resolved
CWD-225	Import and export of Crowd fails when the database is Oracle		Closed
CWD-213	The Sitemesh and Webwork cleanup filters are being wrapped around the XFire requests.		Resolved
CWD-206	JIRA User Import Doesn't Set Groups on Principals		Resolved
CWD-172	Remove this error: SEVERE: No Store configured, persistence disabled		Resolved

Cheers,

The Atlassian Crowd Development Team

Crowd 1.0.5 Release Notes

This page last changed on Apr 19, 2007 by rosie@atlassian.com.

Crowd 1.0.7 has now been released — see the Crowd 1.0.7 Release Notes

The Crowd development team has released Crowd 1.0.5.

If you are running Confluence version 2.4.4 or before, you will need to upgrade the confluence/WEB-INF/lib/atlassian-user-XXXX-XX-XX.jar Atlassian User library to version 2007-04-05. The original library file will need to be backed up, removed, and then replaced with the new version listed above.

This build is mix of bug fixes, documentation improvements, and feature enhancements:

You can now download Crowd from http://www.atlassian.com/Crowd

Atlassian JIRA (15 issues)			
Key	Summary	Pr	Status
CWD-252	Active Directory filter does not exclude accounts which are no sAMAccountName type.		Closed
CWD-244	Set compile flags with maven build scripts to be vs. 1.4		Resolved
CWD-259	Username is not displayed in Confluence (2.4.X) when first logging in.		Closed
CWD-258	Domain for multihost single sign-on is not setting the cookie correctly.		Closed
CWD-257	VerifyTokenFilter missing from the Demo application.		Closed
CWD-256	Importer success screens display success even on an exception.		Resolved
CWD-254	review Installation documentation		Resolved
CWD-248	CLONE -The Sitemesh and Webwork cleanup filters are being wrapped around the XFire requests.		Closed
CWD-243	Document how you can not delete the Crowd console.		Resolved
CWD-242	You can delete the integrated Crowd		Resolved

	application		
CWD-235	System error when no directory is selected when adding a group		Resolved
CWD-234	Add Websphere installation notes for Crowd.		Resolved
CWD-229	Transactions wrapping transactions. The transaction manager is not aware about the wrapping transaction.		Resolved
CWD-222	Crowd is not handling latin1 characters correctly		Resolved
CWD-226	browser window title should say 'View Application'		Resolved

Cheers,

The Atlassian Crowd Development Team

Crowd 1.0.6 Release Notes

This page last changed on Apr 16, 2007 by justen.stepka@atlassian.com.

The Crowd development team has released Crowd 1.0.6.

This build is a quick fix for problems reported with the SSO integration for multi host environments:

You can now download Crowd from http://www.atlassian.com/Crowd

Atlassian JIRA (3 issues)			
Key	Summary	Pr	Status
CWD-265	Confluence displays the users fullname instead of email when integrated with Crowd		Resolved
CWD-263	Fails with exception on Search		Resolved
CWD-262	Improve the management of the Crowd domain during setup and in the Console.		Resolved

Cheers,

The Atlassian Crowd Development Team

Crowd 1.0.7 Release Notes

This page last changed on May 09, 2007 by justin.

The Crowd development team has released Crowd 1.0.7.

This release is a highly recommended upgrade from Crowd 1.0.6 and fixes 2 major issues found in 1.0.6:

Atlassian JIRA (5 issues)			
Key	Summary	Pr	Status
CWD-296	LDAP update password implementation.		Resolved
CWD-316	Active Directory principals can signin with a blank password		Resolved
CWD-181	Continually asked to re-auth with Apache		Resolved
CWD-287	Reset password option for the Console		Resolved
CWD-233	javadoc SecurityServer		Resolved

Cheers,

The Atlassian Crowd Development Team

Installing Crowd

This page last changed on Mar 12, 2007 by rosie@atlassian.com.

Installing Crowd

You can download Crowd here.

- 1. System Requirements
- 2. Installing and Configuring Crowd
  - 2.1 Changing the Port that Crowd uses
- 3. Connecting Crowd to a Database
  - 3.1 HSQL DB
  - 3.2 MS SQL Server
  - 3.3 MySQL
  - 3.4 Oracle
  - 3.5 PostgreSQL
- 4. Running the Setup Wizard

Related Topics

- Crowd Release Notes
- Installing Crowd
- Upgrading Crowd

1. System Requirements

This page last changed on Apr 10, 2007 by justin.

Hardware Requirements

The hardware required to run Crowd depends significantly on the number of applications and users that your installation will have, as well as the maximum number of concurrent requests that the system will experience during peak hours.

During evaluation Crowd will run well on any reasonably fast workstation computer (eg. 1.5+Ghz processor). Memory requirements depend on how many projects and issues you will store, but 256MB is enough for most evaluation purposes.

Most users start by downloading Crowd, and running it on their local computer. It is easy to migrate Crowd to your enterprise infrastructure later.

We would appreciate if you let us know what hardware configuration works for you. Please create a support request in JIRA with your hardware specification and mention the number of users and issues in your Crowd installation.

Software Requirements

1. Sun JDK 1.4 (1.5 or higher is preferred).
2. J2EE 1.4 application server or a Servlet 2.3 web container.
3. JDBC-compliant database that is supported by Hibernate. NOTE: Crowd ships with a built-in HSQL database, which is fine for evaulation purposes. For production environments we recommend configuring Crowd to use an external database.

Supported Databases

The following database servers are supported:

- DB2
- Firebird
- Frontbase
- HypersonicSQL
- Informix
- Ingres
- Interbase
- Pointbase
- PostgreSQL
- Mckoi SQL
- Microsoft SQL Server
- MySQL
- Oracle
- Pointbase
- SAP DB

- Sybase

Supported J2EE Servers

The following J2EE servers are supported:

- Borland
- JBoss
- JOTM
- JOnAS
- JRun
- Orion
- Resin (3.0.x) - tested on 3.0.23
- Tomcat (5.5.x) - tested on 5.5.20
- Weblogic
- WebSphere

Next Step

2. Installing and Configuring Crowd

Related Topics

- 1. System Requirements
- 2. Installing and Configuring Crowd
  - 2.1 Changing the Port that Crowd uses
- 3. Connecting Crowd to a Database
  - 3.1 HSQL DB
  - 3.2 MS SQL Server
  - 3.3 MySQL
  - 3.4 Oracle
  - 3.5 PostgreSQL
- 4. Running the Setup Wizard

2. Installing and Configuring Crowd

This page last changed on Apr 10, 2007 by rosie@atlassian.com.

Installing Crowd

1. Download Crowd.
2. Unzip the download archive (Note: do not specify directory names that contain spaces).
3. Run the start-up script:
    - `start_crowd.bat` for Windows;
      or:
    - `start_crowd.sh` for Unix environment.
4. Point a web browser at http://localhost:8095/ where you will see the Setup Wizard.

Configuring Crowd

You can configure Crowd to suit your environment, as described in:

- 2.1 Changing the Port that Crowd uses
- 3. Connecting Crowd to a Database

Important Files

When configuring Crowd, there are two important files to be aware of:

- `build.properties` — this is a configuration file that stores various deployment properties of Crowd and the demo application.
- `build.xml` — this is an Ant script that loads properties from the build.properties configuration file.

When you change the port that Crowd uses or connect Crowd to an external database, you will need to edit `build.properties` and run `build.bat` (or `build.sh`).

`build.properties`

The default `build.properties` file will look similar to the following:

```
# Modify the attributes of this file to quickly adjust the deployment values of Crowd.

# The Hibernate database dialect to use. See .\etc\hibernate.properties for a list of supported
dialects.
hibernate.dialect=org.hibernate.dialect.HSQLDialect

# The Hibernate transaction factory to use. See .\etc\hibernate.properties for a list of
supported dialects.
hibernate.transaction.factory_class=org.hibernate.transaction.JTATransactionFactory

# Crowd context root
crowd.url=http://localhost:8080/crowd

# Demo context root
crowd.url=http://localhost:8080/demo
```

Parameter	Description
hibernate.dialect	This parameter controls the database dialect the Hibernate persistence system will use when executing commands versus your database server.
hibernate.transaction.factory_class	This parameter controls the transaction factory to use when executing transactions at run-time: Hibernate provides two genertic options, additional application server specific options are available: • `org.hibernate.transaction.JDBCTransactionFactory` delegates to database (JDBC) transactions (default). • `org.hibernate.transaction.JTATransactionFactory` delegates to JTA (if an existing transaction is under way, the work performed is done in that context. Otherwise a new transaction is started).
crowd.url	The path and port for the root of the Crowd Administration Console web-application.
demo.url	The path and port for the door of the Crowd demo web-application

build.xml

If configuring Crowd and/or the demo application to run on a port and context path other than the default, you will need to run the command `build.sh` (or `build.sh`) against the `build.xml` configuration file. This process will then edit all of the necessary Crowd configuration files for your deployment.

The sample output from running `build.xml` will look similar to the following:

```
jstepka-osx:~/atlassian-crowd-1.0.1 $ ./build.sh
Buildfile: build.xml

init:

assistant:
     [echo] Configuring the Crowd Console
     [echo] Copying crowd.properties to: crowd-webapp/WEB-INF/classes
     [copy] Copying 1 file to
```

```
/Users/jstepka/Projects/crowd/releases/atlassian-crowd-1.0.1/crowd-webapp/WEB-INF/classes
     [echo] Configuring the Crowd hibernate configuration
     [echo] Copying hibernate.properties to: crowd-webapp/WEB-INF/classes
     [copy] Copying 1 file to
/Users/jstepka/Projects/crowd/releases/atlassian-crowd-1.0.1/crowd-webapp/WEB-INF/classes
     [echo] Configuring the demo application
     [echo] Renaming and copying demo.properties to:
demo-webapp/WEB-INF/classes/crowd.properties
     [copy] Copying 1 file to
/Users/jstepka/Projects/crowd/releases/atlassian-crowd-1.0.1/demo-webapp/WEB-INF/classes

BUILD SUCCESSFUL
Total time: 0 seconds
```

Related Topics

- 1. System Requirements
- 2. Installing and Configuring Crowd
  - 2.1 Changing the Port that Crowd uses
- 3. Connecting Crowd to a Database
  - 3.1 HSQL DB
  - 3.2 MS SQL Server
  - 3.3 MySQL
  - 3.4 Oracle
  - 3.5 PostgreSQL
- 4. Running the Setup Wizard

2.1 Changing the Port that Crowd uses

This page last changed on Apr 10, 2007 by rosie@atlassian.com.

By default, Crowd is configured to use port 8095. If this port is already in use within your network, you will need to change the port that Crowd uses.

To change the Port that Crowd uses,

1. Edit the `build.properties` file, as described in 2. Installing and Configuring Crowd.
2. Change the `crowd.url` property to the new port on which the Crowd Application Console will be accessed.
3. Change the `demo.url` property to the new port on which the Crowd 'demo' application will be accessed.
4. Run the `build.xml` script, as described in 2. Installing and Configuring Crowd.

Related Topics

- 1. System Requirements
- 2. Installing and Configuring Crowd
  - 2.1 Changing the Port that Crowd uses
- 3. Connecting Crowd to a Database
  - 3.1 HSQL DB
  - 3.2 MS SQL Server
  - 3.3 MySQL
  - 3.4 Oracle
  - 3.5 PostgreSQL
- 4. Running the Setup Wizard

3. Connecting Crowd to a Database

This page last changed on Apr 10, 2007 by rosie@atlassian.com.

By default, Crowd 'Standalone' is shipped preconfigured with HSQL. This is fine for evaluation purposes, but for production installations, you should connect Crowd to an enterprise database. This also lets you take advantage of existing database backup and recovery procedures.

The following instructions will allow you to configure Crowd to an external database:

- 3.1 HSQL DB
- 3.2 MS SQL Server
- 3.3 MySQL
- 3.4 Oracle
- 3.5 PostgreSQL

Database Overview

The Crowd distribution includes the Apache Tomcat application server and an in-memory HSQL database engine. This JNDI reference (`CrowdDS`) can be adjusted to use your custom database and driver by editing the `crowd.xml` deployment description.

You will also need to edit the file `build.properties`, and run the script `build.xml`, as described in 2. Installing and Configuring Crowd. The two relevant properties in the `build.properties` file are:

- `hibernate.dialect`
- `hibernate.transaction.factory_class`

These are described as follows.

`hibernate.dialect`

Below is a list of supported databases and their Hibernate configurations. You will need to edit the `hibernate.dialect` property to correspond to whichever database you are using:

RDBMS	Hibernate SQL Dialect
HypersonicSQL	`org.hibernate.dialect.HSQLDialect`
Microsoft SQL Server	`org.hibernate.dialect.SQLServerDialect`
MySQL	`org.hibernate.dialect.MySQLDialect`
MySQL with InnoDB	`org.hibernate.dialect.MySQLInnoDBDialect`
MySQL with MyISAM	`org.hibernate.dialect.MySQLMyISAMDDialect`
Oracle	`org.hibernate.dialect.OracleDialect`
PostgreSQL	`org.hibernate.dialect.PostgreSQLDialect`

hibernate.transaction.factory_class

You will need to edit the `hibernate.transaction.factory_class` property to correspond to whichever application server you are using:

J2EE Server	Dialect
Borland ES	`org.hibernate.transaction.BESTransactionManagerLookup`
JBoss	`org.hibernate.transaction.JBossTransactionManagerLooku`
JOnAS	`org.hibernate.transaction.JOnASTransactionManagerLooku`
JOTM	`org.hibernate.transaction.JOTMTransactionManagerLookup`
JRun4	`org.hibernate.transaction.JRun4TransactionManagerLooku`
Orion	`org.hibernate.transaction.OrionTransactionManagerLooku`
Resin	`org.hibernate.transaction.ResinTransactionManagerLooku`
Weblogic	`org.hibernate.transaction.WeblogicTransactionManagerLo`
WebSphere	`org.hibernate.transaction.WebSphereTransactionManagerL`

Related Topics

- 1. System Requirements
- 2. Installing and Configuring Crowd
  - 2.1 Changing the Port that Crowd uses
- 3. Connecting Crowd to a Database
  - 3.1 HSQL DB
  - 3.2 MS SQL Server
  - 3.3 MySQL
  - 3.4 Oracle
  - 3.5 PostgreSQL
- 4. Running the Setup Wizard

3.1 HSQL DB

This page last changed on Apr 10, 2007 by rosie@atlassian.com.

The default version of Crowd uses an embedded HSQL DB

Also see http://hsqldb.sourceforge.net/doc/guide/ch01.html#N101C2 .

HSQL DB periodically must update its files to represent changes made in the database. In doing so, it must delete the current `crowddb.data` file on the filesystem (beneath the `/database` folder) and replace it with a new one.

If an administrator issues a shutdown on Crowd in this period, data can be lost, and typically all configuration data for your Crowd server will be lost.

Related Topics

- 1. System Requirements
- 2. Installing and Configuring Crowd
  - 2.1 Changing the Port that Crowd uses
- 3. Connecting Crowd to a Database
  - 3.1 HSQL DB
  - 3.2 MS SQL Server
  - 3.3 MySQL
  - 3.4 Oracle
  - 3.5 PostgreSQL
- 4. Running the Setup Wizard

3.2 MS SQL Server

This page last changed on Apr 10, 2007 by rosie@atlassian.com.

To connect Crowd to MS SQL Server,

1. Configure SQL Server

1. Create a database user which Crowd will connect as (e.g. crowduser).

> In SQL Server, the database user (crowduser above) should not be the database owner, but should be in the `db_owner role`.

2. Create a database for Crowd to store data in (e.g. crowddb).
3. Ensure that the user has permission to connect to the database, and create and populate tables

2. Copy the SQL Server driver to your application server

1. Download the SQL Server JDBC driver from JTDS (recommended, assumed below), or I-net software (commercial).

> Microsoft have their own JDBC driver but we strongly recommend avoiding it after our JIRA customers have reported various connection errors (JRA-5760, [JRA-6872|http://jira.atlassian.com/browse/JRA-6872), workflow problems (JRA-8443) and Chinese character problems (JRA-5054).

2. Add the SQL Server JDBC driver jar (jtds-[version].jar) to the `common/lib` directory.

3. Configure your application server to connect to SQL Server

1. Edit the conf/Catalina/localhost/crowd.xml and customise the username, password, driverClassName and url parameters for the Datasource.

```
<Context path="/crowd" docBase="../../crowd-webapp" debug="0">

   <Resource name="jdbc/CrowdDS" auth="Container" type="javax.sql.DataSource"
             username="[enter db username here]"
             password="[enter db password here]"
             driverClassName="net.sourceforge.jtds.jdbc.Driver"
             url="jdbc:jtds:sqlserver://localhost:1433/crowddb"
             [ delete the minEvictableIdleTimeMillis, timeBetweenEvictionRunsMillis and
maxActive params here ]
          />

   <Manager className="org.apache.catalina.session.PersistentManager" saveOnRestart="false"/>

</Context>
```

2. Delete the minEvictableIdleTimeMillis, timeBetweenEvictionRunsMillis and maxActive attributes (which are only needed for HSQL, and degrade performance otherwise).

4. Configure Crowd to use MS SQL Server

1. Edit the build.properties file located in the root of the standalone release and modify the hibernate.dialect to the following

```
hibernate.dialect=org.hibernate.dialect.SQLServerDialect
```

2. Then run the ./build.sh or build.bat, this will configure crowd to use the MS SQL Server dialect.

If you do not wish to edit this file and run the build script, you can edit the jdbc.properties (which the above script modifies) directly. The jdbc.properties file is located here: `crowd-webapp\WEB-INF\classes\jdbc.properties`; modify the file to the following:

```
# - Crowd Configuration Options

hibernate.connection.datasource=java\:comp/env/jdbc/CrowdDS
hibernate.dialect=org.hibernate.dialect.SQLServerDialect
hibernate.transaction.factory_class=org.hibernate.transaction.JDBCTransactionFactory

...
```

Next Steps

You should now have an application server configured to connect to a database, and Crowd configured to use the correct database. Now start up Crowd and watch the logs for any errors.

Related Topics

- 1. System Requirements
- 2. Installing and Configuring Crowd
  - 2.1 Changing the Port that Crowd uses
- 3. Connecting Crowd to a Database
  - 3.1 HSQL DB
  - 3.2 MS SQL Server
  - 3.3 MySQL
  - 3.4 Oracle
  - 3.5 PostgreSQL
- 4. Running the Setup Wizard

3.3 MySQL

This page last changed on Apr 10, 2007 by rosie@atlassian.com.

To connect Crowd to MySQL,

1. Configure MySQL

1. Create a database user which Crowd will connect as (e.g. crowduser).
2. Create a database for Crowd to store data in (e.g. crowddb).
3. Ensure that the user has permission to connect to the database, and create and populate tables

2. Copy the MySQL driver to your application server

1. Download the latest MySQL Connector/J JDBC driver.
2. Add the MySQL JDBC driver jar (mysql-connector-java-3.x.x-bin.jar) to the `common/lib/` directory. NOTE: Do not place the Debug Driver (mysql-connector-java-3.x.x-bin-g.jar) on the `CLASSPATH` as this can cause issues (JRA-8674).

3. Configure your application server to connect to MySQL

1. Edit the conf/Catalina/localhost/crowd.xml and customise the username, password, driverClassName and url parameters for the Datasource.

```
<Context path="/crowd" docBase="../../crowd-webapp" debug="0">

   <Resource name="jdbc/CrowdDS" auth="Container" type="javax.sql.DataSource"
             username="[enter db username here]"
             password="[enter db password here]"
             driverClassName="com.mysql.jdbc.Driver"
             url="jdbc:mysql://localhost/crowddb?autoReconnect=true&amp;useUnicode=true&amp;characterEncodi
             [ delete the minEvictableIdleTimeMillis, timeBetweenEvictionRunsMillis and
maxActive params here ]
           />

   <Manager className="org.apache.catalina.session.PersistentManager" saveOnRestart="false"/>

</Context>
```

   The URL above assumes a UTF-8 database - ie. created with `create database crowddb character set utf8;`. If you don't specify character set utf8 you risk getting 'Data truncation: Data too long for column' errors.

> ⚠ MySQL closes idle connection after 8 hours, so the `autoReconnect=true` is necessary to tell the driver to reconnect

2. Delete the minEvictableIdleTimeMillis, timeBetweenEvictionRunsMillis and maxActive attributes (which are only needed for HSQL, and degrade performance otherwise).

4. Configure Crowd to use MySQL

1. Edit the build.properties file located in the root of the standalone release and modify the hibernate.dialect to the following, please only choose one of the 3 available options depending on how you have configured your database server.

```
*For MySQL set:*
hibernate.dialect=org.hibernate.dialect.MySQLDialect
*For MySQL with InnoDB set:*
hibernate.dialect=org.hibernate.dialect.MySQLInnoDBDialect
*For MySQL with MyISAM set:*
hibernate.dialect=org.hibernate.dialect.MySQLMyISAMDDialect
```

2. Then run the ./build.sh or build.bat, this will configure crowd to use the MySQL dialect.

If you do not wish to edit this file and run the build script, you can edit the jdbc.properties (which the above script modifies) directly. The jdbc.properties file is located here: `crowd-webapp\WEB-INF\classes\jdbc.properties`, modify the file to the following:

```
# - Crowd Configuration Options

hibernate.connection.datasource=java\:comp/env/jdbc/CrowdDS
hibernate.dialect=org.hibernate.dialect.MySQLDialect
hibernate.transaction.factory_class=org.hibernate.transaction.JDBCTransactionFactory

...
```

Next steps

You should now have an application server configured to connect to a database, and Crowd configured to use the correct database. Now start up Crowd and watch the logs for any errors.

Related Topics

- 1. System Requirements
- 2. Installing and Configuring Crowd
  - 2.1 Changing the Port that Crowd uses
- 3. Connecting Crowd to a Database
  - 3.1 HSQL DB
  - 3.2 MS SQL Server
  - 3.3 MySQL
  - 3.4 Oracle
  - 3.5 PostgreSQL
- 4. Running the Setup Wizard

3.4 Oracle

This page last changed on Apr 10, 2007 by rosie@atlassian.com.

To connect Crowd to Oracle,

1. Configure Oracle

1. Create a database user which Crowd will connect as (e.g. crowduser).
2. Create a database for Crowd to store data in (e.g. crowddb).
3. Ensure that the user has permission to connect to the database, and create and populate tables

2. Copy the Oracle driver to your application server

1. Download the Oracle JDBC driver from http://www.oracle.com/technology/software/tech/java/sqlj_jdbc/index.html.
2. Add the Oracle JDBC driver jar to the `common/lib` directory.

3. Configure your application server to connect to Oracle

1. Edit the file `conf/Catalina/localhost/crowd.xml` and customise the username, password, driverClassName and url parameters for the Datasource.

```
<Context path="/crowd" docBase="../../crowd-webapp" debug="0">

   <Resource name="jdbc/CrowdDS" auth="Container" type="javax.sql.DataSource"
             username="[enter db username here]"
             password="[enter db password here]"
             driverClassName="oracle.jdbc.driver.OracleDriver"
             url="jdbc:oracle:thin:@localhost:1521:crowddb"
             [ delete the minEvictableIdleTimeMillis, timeBetweenEvictionRunsMillis and
maxActive params here ]
           />

   <Manager className="org.apache.catalina.session.PersistentManager" saveOnRestart="false"/>

</Context>
```

2. Delete the minEvictableIdleTimeMillis, timeBetweenEvictionRunsMillis and maxActive attributes (which are only needed for HSQL, and degrade performance otherwise).

4. Configure Crowd to use Oracle

1. Edit the `build.properties file` (located in the root of the standalone release) and modify the hibernate.dialect to the following

```
hibernate.dialect=org.hibernate.dialect.OracleDialect
```

2. Then run the ./build.sh or build.bat, this will configure crowd to use the Oracle dialect.

If you do not wish to edit this file and run the build script, you can edit the jdbc.properties (which the above script modifies) directly. The jdbc.properties file is located here:

`crowd-webapp\WEB-INF\classes\jdbc.properties`, modify the file to the following:

```
# - Crowd Configuration Options

hibernate.connection.datasource=java\:comp/env/jdbc/CrowdDS
hibernate.dialect=org.hibernate.dialect.Oracle
hibernate.transaction.factory_class=org.hibernate.transaction.JDBCTransactionFactory

...
```

Next Steps

You should now have an application server configured to connect to a database, and Crowd configured to use the correct database. Now start up Crowd and watch the logs for any errors.

Related Topics

- 1. System Requirements
- 2. Installing and Configuring Crowd
  - 2.1 Changing the Port that Crowd uses
- 3. Connecting Crowd to a Database
  - 3.1 HSQL DB
  - 3.2 MS SQL Server
  - 3.3 MySQL
  - 3.4 Oracle
  - 3.5 PostgreSQL
- 4. Running the Setup Wizard

3.5 PostgreSQL

This page last changed on Apr 10, 2007 by rosie@atlassian.com.

To connect Crowd to PostgreSQL,

1. Configure PostgreSQL

1. Create a database user which Crowd will connect as (e.g. crowduser).
2. Create a database for Crowd to store data in (e.g. crowddb).
3. Ensure that the user has permission to connect to the database, and create and populate tables

2. Copy the PostgreSQL driver to your application server

1. Download the PostgreSQL JDBC driver from http://jdbc.postgresql.org/download.html. Get the JDBC 3 driver specific to your Postgres version, eg. + postgresql-8.x-xxx.jdbc3.jar.
2. Add the PostgreSQL JDBC driver jar to the common/lib directory.

3. Configure your application server to connect to PostgreSQL

1. Edit the conf/Catalina/localhost/crowd.xml and customise the username, password, driverClassName and url parameters for the Datasource.

```
<Context path="/crowd" docBase="../../crowd-webapp" debug="0">

   <Resource name="jdbc/CrowdDS" auth="Container" type="javax.sql.DataSource"
             username="[enter db username here]"
             password="[enter db password here]"
             driverClassName="org.postgresql.Driver"
             url="jdbc:postgresql://host:port/database" [ see also
http://jdbc.postgresql.org/doc.html) ]"
             [ delete the minEvictableIdleTimeMillis, timeBetweenEvictionRunsMillis and
maxActive params here ]
           />

   <Manager className="org.apache.catalina.session.PersistentManager" saveOnRestart="false"/>

</Context>
```

2. Delete the minEvictableIdleTimeMillis, timeBetweenEvictionRunsMillis and maxActive attributes (which are only needed for HSQL, and degrade performance otherwise).

4. Configure Crowd to use PostgreSQL

1. Edit the build.properties file located in the root of the standalone release and modify the hibernate.dialect to the following

```
hibernate.dialect=org.hibernate.dialect.PostgreSQLDialect
```

2. Then run the ./build.sh or build.bat, this will configure crowd to use the PostgreSQL dialect.

If you do not wish to edit this file and run the build script, you can edit the jdbc.properties (which the

above script modifies) directly. The jdbc.properties file is located here: `crowd-webapp\WEB-INF\classes\jdbc.properties`, modify the file to the following:

```
# - Crowd Configuration Options

hibernate.connection.datasource=java\:comp/env/jdbc/CrowdDS
hibernate.dialect=org.hibernate.dialect.PostgreSQLDialect
hibernate.transaction.factory_class=org.hibernate.transaction.JDBCTransactionFactory

...
```

Next Steps

You should now have an application server configured to connect to a database, and Crowd configured to use the correct database. Now start up Crowd and watch the logs for any errors.

Related Topics

- 1. System Requirements
- 2. Installing and Configuring Crowd
  - 2.1 Changing the Port that Crowd uses
- 3. Connecting Crowd to a Database
  - 3.1 HSQL DB
  - 3.2 MS SQL Server
  - 3.3 MySQL
  - 3.4 Oracle
  - 3.5 PostgreSQL
- 4. Running the Setup Wizard

4. Running the Setup Wizard

This page last changed on Apr 10, 2007 by rosie@atlassian.com.

Welcome to the Setup Wizard

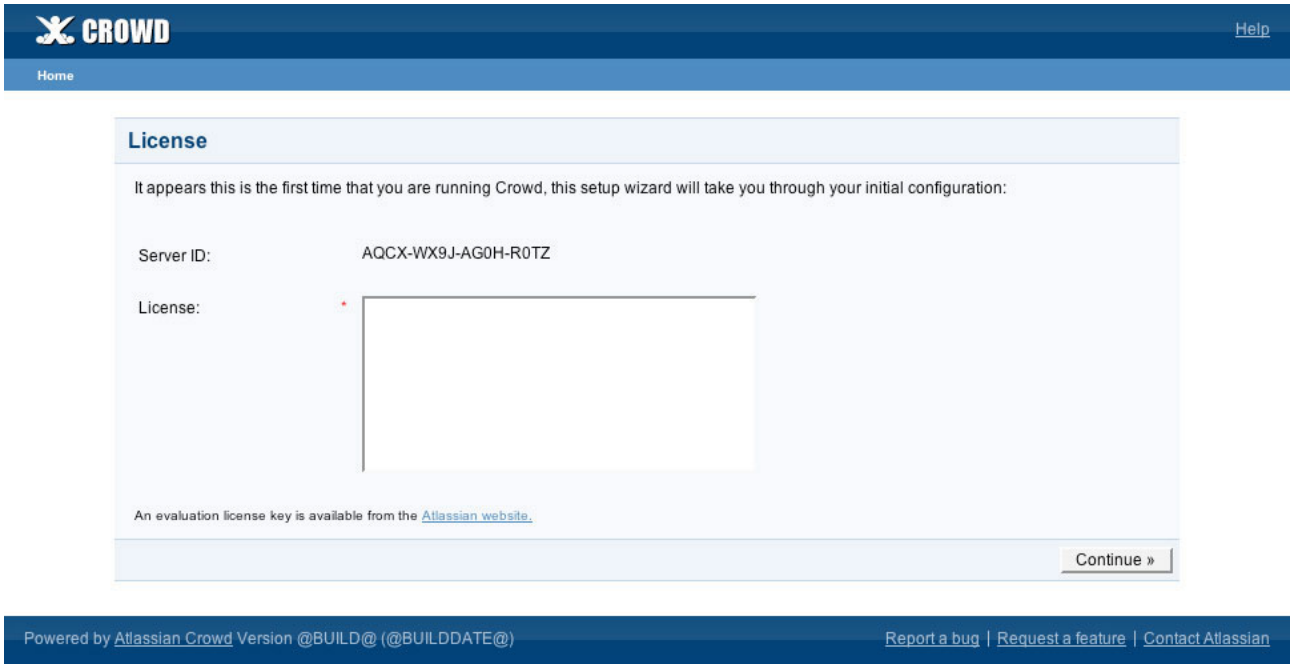
To access the Crowd Administration Console and run the Setup Wizard, go to the URL http://localhost:8095/console or http://localhost:8095/crowd/console.

When accessing the Crowd Administration Console for the first time, you will be presented with the Setup Wizard which will prompt you for a set of default values. Note that all of these values can be adjusted later if required.

1. Licensing

Crowd licenses are based on the number of end-users who will login to one or more of the applications that are integrated with Crowd. Evaluation licenses may be obtained from the Atlassian website.

Screenshot 1: 'License'



2. Options

This part of the setup process controls the general options of the Crowd server.

Screenshot 2: 'Options'

- The Deployment Title specifies a unique name for your Crowd instance. The Deployment Title can be used when sending email notifications.
- The Domain is used when setting HTTP authentication cookies in a user's browser. If this attribute is not the correct, single sign-on will not work when the user switches between applications.
- The Session Timeout controls how long a session will be considered valid during any period of inactivity. This is in minutes and must be greater than `0`.

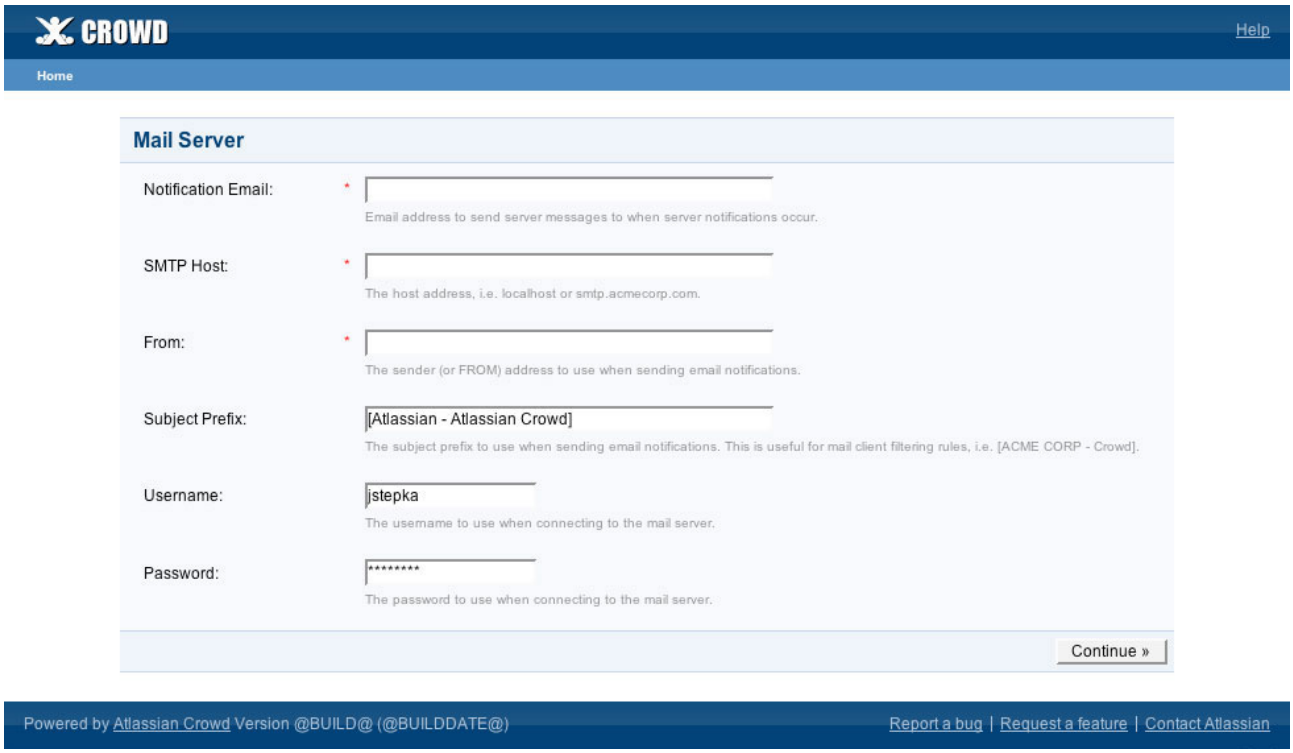
3. Mail Server

Crowd can send email notifications to users during special events such as when a password is reset.

Enter the details of your mail server, and the username and password (if required) that Crowd will use to log in to your mail server:

- Notification Email — The email address which will receive notifications about server events.
- SMTP Host — The hostname of the SMTP mail server, e.g. 'localhost' or 'smtp.acme.com'
- From — The email address from which password notifications will be sent to users.
- Subject Prefix — The prefix which will appear at the start of the email subject, for all emails generated by Crowd. This can be useful for email client programs that offer filtering rules.
- Username — The username that your Crowd server will use when it logs into your mail server.
- Password — The password that your Crowd server will use when it logs into your mail server.

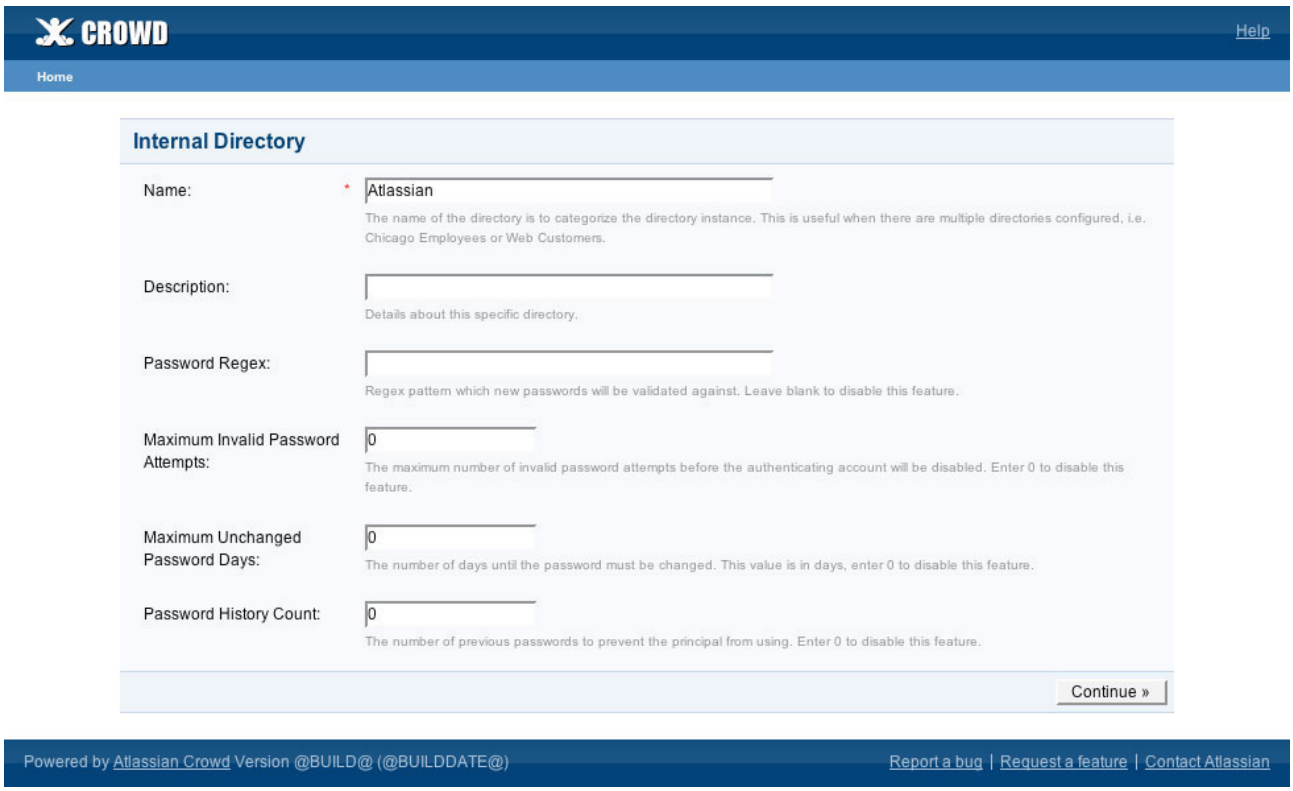
Screenshot 3: 'Mail Server'



4. Default Directory

A default directory needs to be configured. For information about configuring different types of directories (Internal, LDAP or Custom) please see 2.2 Adding a Directory.

Screenshot 4: 'Default Directory'

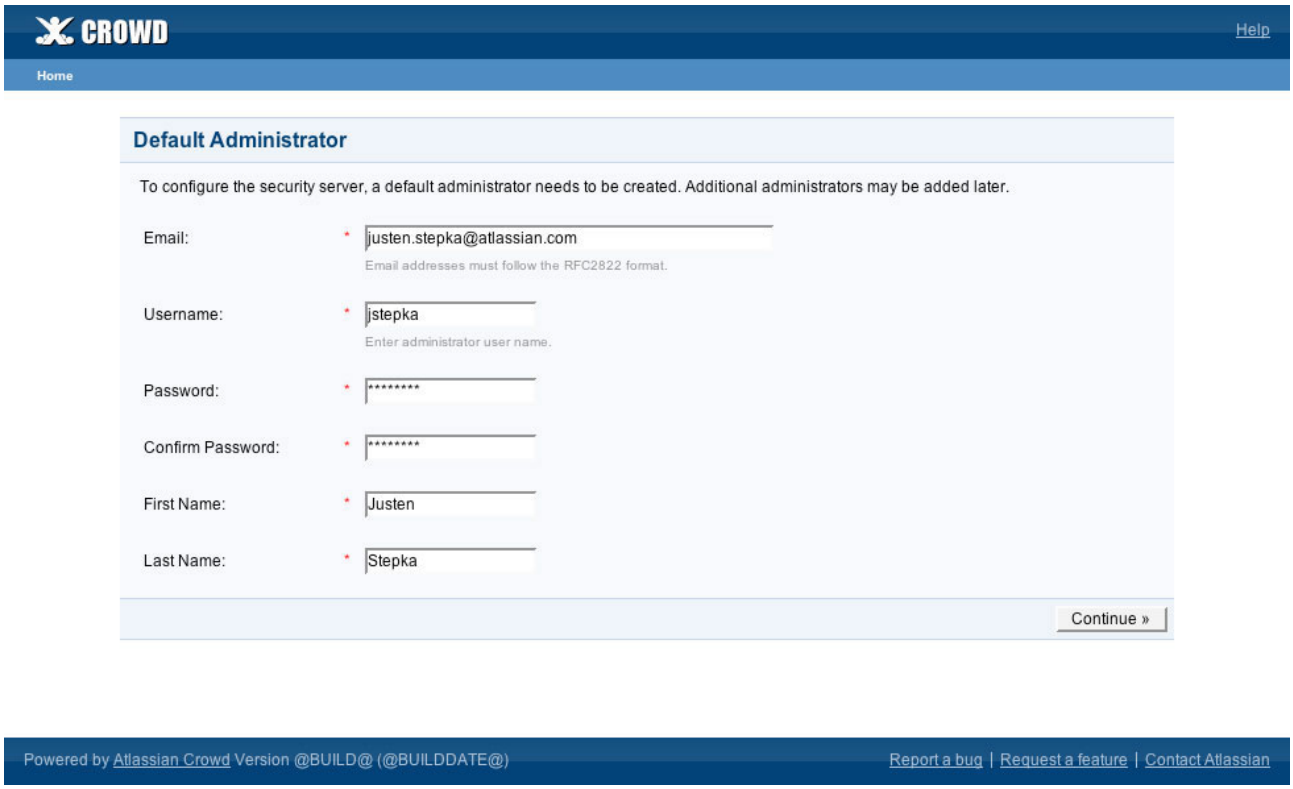


The default group `crowd-administrators` will be automatically created in the default directory.

5. Default Administrator

A default Crowd administrator needs to be created. The default administrator will be automatically added to the default group `crowd-administrators`, thereby giving them rights to access the Crowd Administration Console.

Screenshot 5: 'Detault Administrator'

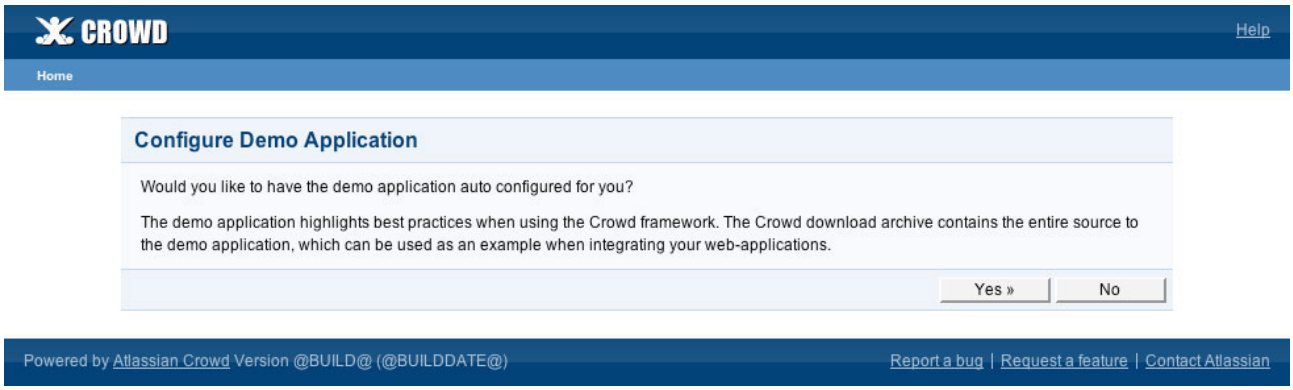


6. 'Demo' Application

An option to auto-configure the 'demo' application server is available to assist you with quickly setting up and configuring Crowd. It is recommended that you choose 'Yes'.

The 'demo' application highlights best practices for using the Crowd framework. The Crowd download archive contains the entire source for the 'demo' application, which can be used as an example when integrating your custom web applications.

Screenshot 6: 'Demo Application'



Setup Complete

You are now ready to use the Crowd Administration Console. For details, please see the Crowd Administration Guide.

Related Topics

- 1. System Requirements
- 2. Installing and Configuring Crowd
  - 2.1 Changing the Port that Crowd uses
- 3. Connecting Crowd to a Database
  - 3.1 HSQL DB
  - 3.2 MS SQL Server
  - 3.3 MySQL
  - 3.4 Oracle
  - 3.5 PostgreSQL
- 4. Running the Setup Wizard

Upgrading Crowd

This page last changed on Apr 02, 2007 by justin.

Upgrading Crowd

Before your begin:

- Please make sure you read the release notes for the version of Crowd you are upgrading to.
- Please backup your Crowd data (see 5.4 Backing Up and Restoring Data) and backup your external database (if you have one).

Database Configuration

You will need to check if the `crowd.xml` in the `conf/catalina/localhost/` directory is hooked up to a database. If it is, then you will need to modify the `crowd.xml` from the new archive to correspond to the previous one.

Do not forget to copy over any libs necessary for your JDBC connection. If you are using the in memory database (HSQL), you will need to copy over the contents of the `database` folder to the new archive `download` folder.

If you are not using the in-memory HSQL DB database, please make sure to also copy over the `jdbc.properties` file from the Crowd `WEB-INF/classes` folder to the new `WEB-INF/classes` folder.

The Crowd Administration Console

Copy the old `jdbc.properties` and `crowd.properties` files from the `WEB-INF/classes` folder to the new `WEB-INF/classes` folder. The `application.password` in this file is critical. If you do not do this the Crowd console will not work.

See Also...

- Crowd 1.0 Upgrade Notes

Related Topics

- Crowd Release Notes
- Installing Crowd
- Upgrading Crowd

Crowd 1.0 Upgrade Notes

This page last changed on Mar 25, 2007 by justin.

- All LDAP configuration now need to have filters set
- If you are using PostgreSQL you need to change the column name attributevalues.attributevalueid to attributevalues.ATTRIBUTEVALUEID (make it uppercase).

Crowd Knowledge Base

This page last changed on Mar 11, 2007 by rosie@atlassian.com.

General FAQ

Concepts:

- What is single sign-on (SSO)?
- What is authorisation?
- What is authentication?
- What is centralised authentication?
- What is identity management?
- What is a directory?

Technical:

- How does Crowd work? How is Crowd an "application security framework"?
- What is an application connector?
- What is a directory connector?
- How many users can Crowd manage?
- How many applications can be used with Crowd?
- We already have an LDAP server for Confluence and/or JIRA. Do we really need Crowd?

Compatibility:

- What are Crowd's system requirements?
- What directories and applications does Crowd support out-of-the-box?
- How can Crowd be connected to new or currently unsupported applications?
- How does Crowd integrate with other Atlassian products?
- Does Crowd include kerberos integration?
- Does Crowd support SAML or Liberty Alliance?

Deployment FAQ

- Self Signed Certificate

External Resources

Integration FAQ

- IBM Websphere Integration

Deployment FAQ

This page last changed on Mar 11, 2007 by rosie@atlassian.com.

- Self Signed Certificate

Self Signed Certificate

This page last changed on Nov 30, 2006 by justen.stepka@atlassian.com.

I have a self Signed Certificate

You will need to add the self-signed certificate to your JDK truststore using the JDK keytool:
http://java.sun.com/j2se/1.3/docs/tooldocs/win32/keytool.html

External Resources

This page last changed on Mar 11, 2007 by rosie@atlassian.com.

Integration FAQ

This page last changed on Apr 01, 2007 by rosie@atlassian.com.

IBM Websphere Integration

This page last changed on Apr 01, 2007 by justin.

If your client application is running in Websphere, there is a known problem with Websphere's XML libraries.

Crowd uses XFire to handle the requests between the client application (JIRA, Confluence, Bamboo etc.) and Crowd, XFire requires a newer version of an XML library than what is shipped with Websphere 5.1.

More information and a link to a newer version of the relevant JAR file is available on the XFire website

You will need to add the qname.jar file to the `WebSphere\AppServer\lib` directory and remove the old file.