Space Details

<table>
<thead>
<tr>
<th>Key:</th>
<th>CROWD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name:</td>
<td>Crowd 1.2</td>
</tr>
<tr>
<td>Description:</td>
<td>Latest documentation for Crowd single sign-on and identity management</td>
</tr>
<tr>
<td>Creator (Creation Date):</td>
<td><a href="mailto:justen.stepka@atlassian.com">justen.stepka@atlassian.com</a> (Sep 28, 2006)</td>
</tr>
<tr>
<td>Last Modifier (Mod. Date):</td>
<td>smaddox (Nov 26, 2007)</td>
</tr>
</tbody>
</table>

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
            - Configuring an SSL Certificate for 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
        - 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
      - 2.4.4 Importing Users from CSV Files
        - 2.4.4.1 Configuring the CSV Importer
        - 2.4.4.2 Mapping CSV Fields to Crowd Fields
        - 2.4.4.3 Confirming the CSV Importer Configuration
        - 2.4.4.4 Viewing the Results of the Import
      - 2.4.5 Importing Users from Atlassian Bamboo
  - 3. Managing Applications
    - 3.1 Using the Application Browser
    - 3.2 Adding an Application
      - 3.2.01 Integrating Crowd with Apache
      - 3.2.02 Integrating Crowd with Subversion
      - 3.2.03 Integrating Crowd with Atlassian Confluence
3.2.3.1 Configuring Confluence for NTLM SSO
3.2.04 Integrating Crowd with Atlassian CrowdID
3.2.05 Integrating Crowd with Atlassian FishEye
3.2.06 Integrating Crowd with Atlassian JIRA
   3.2.6.1 Configuring JIRA for NTLM SSO
3.2.07 Integrating Crowd with Jive Forums
   3.2.7.1 Jive SSO
3.2.08 Integrating Crowd with Atlassian Bamboo
3.2.09 Integrating Crowd with Acegi Security
   Integrating AppFuse - a Crowd-Acegi Integration Tutorial
3.2.10 Integrating Crowd with a Custom Application
3.3 Mapping a Directory to an Application
   3.3.1 Specifying the Directory Order for an Application
   3.3.2 Specifying an Application's Directory Permissions
   Example of Directory Permissions
3.4 Specifying which Groups can access an Application
3.5 Specifying an Application's Address or Hostname
3.6 Testing a User's Login to an Application
3.7 Managing an Application's Session
3.8 Deleting or Deactivating an Application
4. Managing Principals, Groups and Roles
   4.01 Using the Principal Browser
   4.02 Adding a Principal
   4.03 Deleting or Deactivating a Principal
   4.04 Managing a Principal's Session
   4.05 Editing a Principal's Details and Password
   4.06 Specifying a Principal's Attributes
   4.07 Editing a Principal's Group and Role Membership
   4.08 Granting Crowd Administration Rights to a User
   4.09 Using the Group Browser and Role Browser
   4.10 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
    • Maven 2 Integration
• SOAP API
  • Axis Client Stub Generation
  • Microsoft .NET Client

Creating a Custom Directory Connector

Crowd Installation & Upgrade Guide
• Crowd Release Notes
  • 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
  • Crowd 1.1.0 Release Notes
  • Crowd 1.1.1 Release Notes
  • Crowd 1.1.2 Release Notes
  • Crowd 1.2 Release Notes

Installing Crowd
• 1. System Requirements
  • 1.1 Setting JAVA_HOME
• 2. Installing Crowd and CrowdID
  • 2.1 Connecting Crowd to a Database
    • 2.1.1 HSQL DB
    • 2.1.2 MS SQL Server
- 2.1.3 MySQL
- 2.1.4 Oracle
- 2.1.5 PostgreSQL
- 2.2 Connecting CrowdID to a Database
  - 2.2.1 HSQL DB
  - 2.2.2 MS SQL Server
  - 2.2.3 MySQL
  - 2.2.4 Oracle
  - 2.2.5 PostgreSQL
- 2.3 Installing Crowd and CrowdID WAR Distribution
  - 2.3.1 Installing Crowd WAR on JBoss
- 3. Running the Setup Wizard
- 4. Configuring Crowd
  - 4.1 Important Files
    - The crowd.properties File
  - 4.2 Changing the Port that Crowd uses
- 5. Installing Crowd as a Windows Service
  - 5.1 Specifying Startup Order of Windows Services
  - 5.2 Changing the User for the Crowd Windows Service
  - 5.3 Removing the Crowd Windows Service
  - 5.4 Troubleshooting Crowd as a Windows Service
- Upgrading Crowd
  - Crowd 1.0 Upgrade Notes
  - Crowd 1.1 Upgrade Notes
  - Crowd 1.2 Upgrade Notes
- Crowd Knowledge Base
  - Deployment FAQ
    - Recovering your Console application password
    - Resetting the Domain Cookie Value
    - Self Signed Certificate
  - Integration FAQ
    - All Integrations
      - If I delete a principal from Crowd, how will this affect integrated applications?
    - Atlassian Product Integration
      - Application Caching
    - JIRA integration
    - Public Signup Setup
    - IBM Websphere Integration
- CrowdID Administration Guide
  - 1. About CrowdID
    - 1.1 How CrowdID works with Crowd
      - 1.1.1 Determining the name of the CrowdID application
      - 1.1.2 Locating the Crowd Server that CrowdID is using
• 1.1 How OpenID sites interact with CrowdID
• 2. Allowing users to access CrowdID
  • 2.1 Granting CrowdID access rights to a user
  • 2.2 Granting CrowdID administration rights to a user
• 3. Specifying the sites to which users can login
  • 3.1 Allowing all hosts
  • 3.2 Allowing all except specified hosts ('Blacklist')
  • 3.3 Allowing specified hosts only ('Whitelist')
• 4. Configuring CrowdID system settings
  • 4.1 Specifying the CrowdID URL
  • 4.2 Enabling localhost authentication
  • 4.3 Enabling immediate authentication requests
  • 4.4 Enabling communication with stateless clients

CrowdID User Guide
• 1. Getting started with CrowdID
  • 1.1 What is OpenID?
  • 1.2 What is CrowdID?
  • 1.3 What is an OpenID URL or identifier?
  • 1.4 Viewing the CrowdID page
• 2. Logging in to a website using OpenID
  • 2.1 Does the website support OpenID?
  • 2.2 Entering your OpenID URL
  • 2.3 Logging in to CrowdID
  • 2.4 Allowing or denying a login
  • 2.5 Providing additional profile information to a website
• 3. Viewing your always-approved websites
• 4. Viewing your login history
• 5. Updating your profile
• 6. Using more than one profile
  • 6.1 Adding a profile
  • 6.2 Choosing a profile for a website
  • 6.3 Setting a default profile
  • 6.4 Deleting a profile
• 7. Changing or resetting your password
  • 7.1 Changing your password
  • 7.2 Resetting your password

__newreleaseCrowd

TreeNavigation
Crowd Documentation

This page last changed on Nov 26, 2007 by smaddox.

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 selected applications by user and group
- Easily connect Crowd's application framework to new web applications

If you have a question about using Crowd, please contact our support team. You may also want to check out the mailing lists and forums:

- Crowd Announcements
- Crowd General Forum
- Crowd Developers Forum

Other handy links:

- Crowd Knowledge Base
- Javadoc
- JIRA Issue Tracker for Crowd

Download

You can download the Crowd documentation in PDF, HTML or XML formats.

Previous Versions

Crowd 1.1 Documentation
Crowd 1.0 Documentation

Table of Contents

Crowd Administration Guide

- 1. Getting Started
- 2. Managing Directories
- 3. Managing Applications
- 4. Managing Principals, Groups and Roles
- 5. System Administration

Crowd Development Hub

- Creating a Crowd Client for your Custom Application
- Creating a Custom Directory Connector
Crowd Installation & Upgrade Guide

- Crowd Release Notes
- Installing Crowd
- Upgrading Crowd

Crowd Knowledge Base

- Deployment FAQ
- Integration FAQ

CrowdID Administration Guide

- 1. About CrowdID
- 2. Allowing users to access CrowdID
- 3. Specifying the sites to which users can login
- 4. Configuring CrowdID system settings

CrowdID User Guide

- 1. Getting started with CrowdID
- 2. Logging in to a website using OpenID
- 3. Viewing your always-approved websites
- 4. Viewing your login history
- 5. Updating your profile
- 6. Using more than one profile
- 7. Changing or resetting your password
Crowd Administration Guide

This page last changed on Aug 09, 2007 by rosie@atlassian.com.

Crowd is a web-based single sign-on (SSO) tool that simplifies application provisioning and identity management.

The Crowd Administration Guide is for people who have Crowd administration rights.

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
        - Configuring an SSL Certificate for 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
    - 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
    - 2.4.4 Importing Users from CSV Files
      - 2.4.4.1 Configuring the CSV Importer
      - 2.4.4.2 Mapping CSV Fields to Crowd Fields
      - 2.4.4.3 Confirming the CSV Importer Configuration
      - 2.4.4.4 Viewing the Results of the Import
    - 2.4.5 Importing Users from Atlassian Bamboo

• 3. Managing Applications
  • 3.1 Using the Application Browser
  • 3.2 Adding an Application
    - 3.2.01 Integrating Crowd with Apache
    - 3.2.02 Integrating Crowd with Subversion
    - 3.2.03 Integrating Crowd with Atlassian Confluence
      - 3.2.3.1 Configuring Confluence for NTLM SSO
    - 3.2.04 Integrating Crowd with Atlassian CrowdID
    - 3.2.05 Integrating Crowd with Atlassian FishEye
    - 3.2.06 Integrating Crowd with Atlassian JIRA
      - 3.2.6.1 Configuring JIRA for NTLM SSO
    - 3.2.07 Integrating Crowd with Jive Forums
    - 3.2.08 Integrating Crowd with Atlassian Bamboo
    - 3.2.09 Integrating Crowd with Acegi Security
      - Integrating AppFuse - a Crowd-Acegi Integration Tutorial
    - 3.2.10 Integrating Crowd with a Custom Application
  • 3.3 Mapping a Directory to an Application
    - 3.3.1 Specifying the Directory Order for an Application
    - 3.3.2 Specifying an Application's Directory Permissions
      - Example of Directory Permissions
  • 3.4 Specifying which Groups can access an Application
  • 3.5 Specifying an Application's Address or Hostname
  • 3.6 Testing a User's Login to an Application
  • 3.7 Managing an Application's Session
  • 3.8 Deleting or Deactivating an Application

• 4. Managing Principals, Groups and Roles
4.01 Using the Principal Browser
4.02 Adding a Principal
4.03 Deleting or Deactivating a Principal
4.04 Managing a Principal's Session
4.05 Editing a Principal's Details and Password
4.06 Specifying a Principal's Attributes
4.07 Editing a Principal's Group and Role Membership
4.08 Granting Crowd Administration Rights to a User
4.09 Using the Group Browser and Role Browser
4.10 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
     - 5.2.6.1 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
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

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

Application Connectors

- Atlassian JIRA
- Atlassian Confluence
- Atlassian Bamboo
- Atlassian Fisheye
- Apache
- Subversion
- Jive Forums
- Atlassian CrowdID
- Acegi
- NTLM for JIRA
- NTLM for Confluence

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

The Crowd Administration Console allows you to:

- 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,


   Only authorised administrators can login to the Crowd Administration 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

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
    - Configuring an SSL Certificate for 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
  - 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
  - 2.4.4 Importing Users from CSV Files
    - 2.4.4.1 Configuring the CSV Importer
    - 2.4.4.2 Mapping CSV Fields to Crowd Fields
    - 2.4.4.3 Confirming the CSV Importer Configuration
    - 2.4.4.4 Viewing the Results of the Import
  - 2.4.5 Importing Users from Atlassian Bamboo
2.1 Using the Directory Browser

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.

You created one default directory when you setup Crowd. To add more directories, see 2.2 Adding a Directory.

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
    - Microsoft Active Directory
    - Configuring an SSL Certificate for Microsoft Active Directory
    - SunONE
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
- 2.4.4 Importing Users from CSV Files
  - 2.4.4.1 Configuring the CSV Importer
  - 2.4.4.2 Mapping CSV Fields to Crowd Fields
  - 2.4.4.3 Confirming the CSV Importer Configuration
  - 2.4.4.4 Viewing the Results of the Import
- 2.4.5 Importing Users from Atlassian Bamboo

Crowd Documentation
2.2 Adding a Directory

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 (Microsoft 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'

![Add Directory screenshot]

Screenshot 2: 'Select Directory Type'

![Select Directory Type screenshot]
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**
    - Configuring an SSL Certificate for 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**
  - **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**
  - **2.4.4 Importing Users from CSV Files**
    - **2.4.4.1 Configuring the CSV Importer**
    - **2.4.4.2 Mapping CSV Fields to Crowd Fields**
    - **2.4.4.3 Confirming the CSV Importer Configuration**
    - **2.4.4.4 Viewing the Results of the Import**
  - **2.4.5 Importing Users from Atlassian Bamboo**

[Link to Crowd Documentation]
2.2.1 Configuring an Internal Directory

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'

<table>
<thead>
<tr>
<th>Internal Directory Attributes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>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.</td>
</tr>
<tr>
<td>Description</td>
<td>Details about this specific directory.</td>
</tr>
<tr>
<td>Active</td>
<td>Only deselect if you wish to prevent all users ('principals') within the directory from accessing all mapped applications.</td>
</tr>
<tr>
<td>Password Regex</td>
<td>Regex pattern which new passwords will be validated against. Leave blank to disable this feature.</td>
</tr>
</tbody>
</table>

For compatibility between Atlassian products we must use ATLASIAN-SHA1.
Regex pattern which new passwords will be validated against. The regular expression format used is the `java.util.regex.Pattern`. For example, for an alphanumeric password of at least 8 characters, you could use the pattern: `*\[A-Za-z0-9\]{8,}*` 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.

**Maximum 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.

**Password Encryption**
If you wish to [import users](#) into this directory from another Atlassian product, specify 'ATLASSIAN-SHA1' in order to ensure password compatibility.

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](#)
  - [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](#)
  - [2.4.4 Importing Users from CSV Files](#)
    - [2.4.4.1 Configuring the CSV Importer](#)
    - [2.4.4.2 Mapping CSV Fields to Crowd Fields](#)
    - [2.4.4.3 Confirming the CSV Importer Configuration](#)
    - [2.4.4.4 Viewing the Results of the Import](#)
  - [2.4.5 Importing Users from Atlassian Bamboo](#)
2.2.2 Configuring an LDAP Directory Connector

This page last changed on Aug 09, 2007 by justen.stepka@atlassian.com.

Crowd provides built-in connectors for the most popular LDAP directory servers (Microsoft 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
7. Click the 'Test Connection' button to verify that Crowd can successfully connect to the directory.
8. Click the 'Continue' button.
9. This will display the 'Configuration' tab (see Screenshot 3 below). Fill in the configuration details for your groups, roles and principals (users), as described in the tables below Screenshot 3. Also please see LDAP Object Structures (below).
10. Click the 'Test Search' button to verify that Crowd can successfully locate groups/roles/principals within the directory.
11. 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'

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>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'.</td>
</tr>
<tr>
<td>Description</td>
<td>Details about this specific directory.</td>
</tr>
<tr>
<td>Active</td>
<td></td>
</tr>
</tbody>
</table>
Only deselect this if you wish to prevent all users ('principals') within the directory from accessing all mapped applications.

**Screenshot 2: 'Connector'**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connector</td>
<td>The directory connector to use when communicating with the directory server.</td>
</tr>
<tr>
<td>URL</td>
<td>The connection URL to use when connecting to the directory server, e.g.: ldap://localhost:389, or port 636 for SSL.</td>
</tr>
<tr>
<td>Secure SSL</td>
<td>Specifies if the connection to the directory server is a SSL connection.</td>
</tr>
<tr>
<td>Use Node Referrals</td>
<td>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.</td>
</tr>
<tr>
<td>Use Paged Results</td>
<td>Use the LDAP control extension for simple paging of search results. Retrieves chunks of data rather</td>
</tr>
</tbody>
</table>
than all of the search results at once. This feature may be necessary when using Microsoft Active Directory if more than 999 results are returned for any given search.

**Paged Results Size**
Enter the desired page size i.e. the maximum number of search results to be returned per page, when paged results are enabled. Defaults to 999 results. This option is available from Crowd 1.1.1.

**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

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

**Screenshot 3: 'Configuration'**

Create Directory Connector

<table>
<thead>
<tr>
<th>Details</th>
<th>Connector</th>
<th>Configuration</th>
<th>Permissions</th>
</tr>
</thead>
</table>

**Group Configuration**

**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**
`group`
The LDAP user object class type to use when loading groups.

**Group Object Filter**
`objectCategory=Group`
The filter to use when searching group objects.

**Group Name Attribute**
`cn`
The attribute field to use when loading the group name.

**Group Description Attribute**
`description`
The attribute field to use when loading the group description.

**Group Members Attribute**
`member`
The attribute field to use when loading the group members.

**Role Configuration**

**Role DN**

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

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group DN</td>
<td>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.</td>
</tr>
<tr>
<td>Group Object Class</td>
<td>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.</td>
</tr>
<tr>
<td>Group Object Filter</td>
<td>The filter to use when searching group objects.</td>
</tr>
<tr>
<td>Group Name Attribute</td>
<td>The attribute field to use when loading the group's name.</td>
</tr>
<tr>
<td>Group Description Attribute</td>
<td>The attribute field to use when loading the group's description.</td>
</tr>
<tr>
<td>Group Members Attribute</td>
<td>The attribute field to use when loading the group's members.</td>
</tr>
</tbody>
</table>

### Role Configuration

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Role DN</td>
<td>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.</td>
</tr>
<tr>
<td>Role Object Class</td>
<td>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.</td>
</tr>
<tr>
<td>Role Object Filter</td>
<td>The filter to use when searching role objects.</td>
</tr>
<tr>
<td>Role Name Attribute</td>
<td>The attribute field to use when loading the role's name.</td>
</tr>
<tr>
<td>Role Description Attribute</td>
<td>The attribute field to use when loading the role's description.</td>
</tr>
<tr>
<td>Role Members Attribute</td>
<td>The attribute field to use when loading the role's members.</td>
</tr>
</tbody>
</table>
Principal Configuration
(In Crowd, users are known as principals.)

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>User DN</td>
<td>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.</td>
</tr>
<tr>
<td>User Object Class</td>
<td>The LDAP user object class type to use when loading principals.</td>
</tr>
<tr>
<td>User Object Filter</td>
<td>The filter to use when searching user objects.</td>
</tr>
<tr>
<td>User Name</td>
<td>The attribute field to use when loading the principal's username.</td>
</tr>
<tr>
<td>User First Name</td>
<td>The attribute field to use when loading the principal's first name.</td>
</tr>
<tr>
<td>User Last Name</td>
<td>The attribute field to use when loading the principal's last name.</td>
</tr>
<tr>
<td>User Email</td>
<td>The attribute field to use when loading the principal's email.</td>
</tr>
<tr>
<td>User Group</td>
<td>The attribute field to use when loading the principal's groups.</td>
</tr>
<tr>
<td>User Password</td>
<td>The attribute field to use when manipulating a principal's password.</td>
</tr>
</tbody>
</table>

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.

Supported Attributes:
Crowd's LDAP connectors support the adding and updating of the following principal attributes when integrating with an LDAP server:

- surname
- given name
- email
- password

If you need support for additional LDAP attributes, the Crowd LDAP connector can be extended to provide support for whatever attributes you need.

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
      - Configuring an SSL Certificate for 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
  - 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
  - 2.4.4 Importing Users from CSV Files
    - 2.4.4.1 Configuring the CSV Importer
    - 2.4.4.2 Mapping CSV Fields to Crowd Fields
    - 2.4.4.3 Confirming the CSV Importer Configuration
    - 2.4.4.4 Viewing the Results of the Import
  - 2.4.5 Importing Users from Atlassian Bamboo

Crowd Documentation
### 2.2.2.1 Microsoft Active Directory

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'

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connector</td>
<td>The directory connector to use when communicating with the directory server.</td>
</tr>
<tr>
<td>URL</td>
<td>The connection URL to use when connecting to the directory server, e.g.: <code>ldap://localhost:389</code>, or port 636 for SSL.</td>
</tr>
<tr>
<td>Secure SSL</td>
<td>Specifies if the connection to the directory server is a SSL connection.</td>
</tr>
<tr>
<td>Use Node Referrals</td>
<td>Use the JNDI lookup <code>java.naming.referral</code> option. Generally needed for Active Directory servers configured without proper DNS, to prevent a <code>javax.naming.PartialResultException</code> error.</td>
</tr>
<tr>
<td>Use Paged Results</td>
<td>Use the LDAP control extension for simple paging of search results. Retrieves chunks of data rather than all of the results at once.</td>
</tr>
</tbody>
</table>

Document generated by Confluence on Nov 27, 2007 21:24
than all of the search results at once. This feature may be necessary when using Microsoft Active Directory if more than 999 results are returned for any given search.

### Paged Results Size

Enter the desired page size i.e. the maximum number of search results to be returned per page, when paged results are enabled. Defaults to 999 results. This option is available from Crowd 1.1.1.

### 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

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base DN</td>
<td>cn=users,dc=ad,dc=acmecorp,dc=com</td>
</tr>
<tr>
<td>User DN</td>
<td><a href="mailto:administrator@ad.acmecorp.com">administrator@ad.acmecorp.com</a></td>
</tr>
</tbody>
</table>

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 LDAP structure of your server.

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

### Configuring an SSL Certificate for Microsoft Active Directory

If you wish to use Crowd to add principals or change passwords in Microsoft Active Directory, you will need to install an SSL certificate generated by your Active Directory server and then install the certificate into your JVM keystore. Please read the instructions: [Configuring an SSL Certificate for Microsoft Active Directory](#).

## 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](#)
    - Configuring an SSL Certificate for 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](#)
  - [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
  ° 2.4.4 Importing Users from CSV Files
    - 2.4.4.1 Configuring the CSV Importer
    - 2.4.4.2 Mapping CSV Fields to Crowd Fields
    - 2.4.4.3 Confirming the CSV Importer Configuration
    - 2.4.4.4 Viewing the Results of the Import
  ° 2.4.5 Importing Users from Atlassian Bamboo

Crowd Documentation
Configuring an SSL Certificate for Microsoft Active Directory

You can configure Crowd to work with Microsoft Active Directory by setting up an LDAP connector in Crowd. If you wish to use Crowd to add principals or change passwords in Active Directory, you will need to install an SSL certificate generated by your Active Directory server and then install the certificate into your JVM keystore.

Prerequisites

Make sure that you have the following installed on your Windows server (domain controller):

<table>
<thead>
<tr>
<th>Required Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows 2000 Service Pack 2</td>
<td>Required if you are using Windows 2000</td>
</tr>
<tr>
<td>Internet Information Services (IIS)</td>
<td>This is required before you can install Windows Certificate Services.</td>
</tr>
<tr>
<td>Windows Certificate Services</td>
<td>This installs a certification authority (CA) which is used to issue certificates.</td>
</tr>
</tbody>
</table>

Step 1. Install the Microsoft Certificate Services

1. Using the Active Directory Control Panel – Add/Remove Programs administration tool:
   - Select 'Add/Remove Windows Components' to start the Windows Components Wizard.
   - Place check marks next to 'Certificate Services' and 'Internet Information Services (IIS)'.
   - Click 'Next'>.

2. Select 'Enterprise root CA' Certificate Authority Type and click 'Next>'.

![Windows Components Wizard](image)

Total disk space required: 2.1 MB
Space available on disk: 6016.3 MB
3. Enter a 'CA name' (server name) and click 'Next>'.

4. Leave the 'Data Storage Locations' as default and click 'Next>'.
5. The software installation process is complete. Click 'Finish'.

6. Click 'OK' to restart IIS.

7. You will now need to restart your Microsoft Active Directory Server.

Step 2. Obtain the Server Certificate

The steps above describe how to install the certification authority (CA) on your Microsoft Active Directory server. Next, you will need to add the Microsoft Active Directory server’s SSL certificate to the list of accepted certificates used by the JDK that runs your Crowd server.
The Active Directory certificate is automatically generated and placed in root of the C:\ drive, matching a file format similar to the tree structure of your Active Directory server, e.g. c:\crowd-ad2000.ad01.crowd.atlassian.com_ad01.crt.

You can also export the certificate by executing this command on the Active Directory server:

certutil -ca.cert crowd-client.crt

Step 3. Import the Server Certificate

Now you need to import the Active Directory certificate to the list of accepted certificates in your JDK runtime environment.

- Assuming your JDK is installed here C:\Program Files\Java\jdk1.5.0_12, you will need to run the following command:
  C:\Program Files\Java\jdk1.5.0_12\keytool -import -alias crowd_crt -file crowd-client.crt -keystore "C:\Program Files\Java\jdk1.5.0_12\jre\lib\security\cacerts"
- The keytool import will prompt you for a password during import. The default keystore password is changeit.
- When prompted Trust this certificate? [no]: enter 'yes' to confirm the Active Directory Server key import:

  Enter keystore password: changeit
  Owner: CN=ad01, C=US
  Issuer: CN=ad01, C=US
  Serial number: 15563d6677a4e9e4582d8a84be683f9
  Certificate fingerprints:
  Trust this certificate? [no]: yes
  Certificate was added to keystore

You may now use the Secure SSL option when connecting to an Active Directory server with Crowd's built in connectors.

Related Topics

2.2.2.1 Microsoft Active Directory
2.2.2.2 SunONE

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'

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connector</td>
<td>The directory connector to use when communicating with the directory server.</td>
</tr>
<tr>
<td>URL</td>
<td>The connection URL to use when connecting to the directory server, e.g.: <code>ldap://localhost:389</code>, or port 639 for SSL.</td>
</tr>
<tr>
<td>Secure SSL</td>
<td>Specifies if the connection to the directory server is a SSL connection.</td>
</tr>
<tr>
<td>Use Node Referrals</td>
<td>Use the JNDI lookup java.naming.referral option. Generally needed for Active Directory servers configured without proper DNS, to prevent a <code>javax.naming.PartialResultException: Unprocessed Continuation Reference(s)</code> error.</td>
</tr>
<tr>
<td>Use Paged Results</td>
<td>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.</td>
</tr>
</tbody>
</table>
Directory if more than 999 results are returned for any given search.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base DN</td>
<td>Enter the root distinguished name to use when running queries versus the directory server, e.g.: o=acmecorp,c=com.</td>
</tr>
<tr>
<td>User DN</td>
<td>The username that Crowd will use when connecting to the directory server.</td>
</tr>
<tr>
<td>Password</td>
<td>The password that Crowd will use when connecting to the directory server.</td>
</tr>
</tbody>
</table>

**Configuration details for SunONE**

<table>
<thead>
<tr>
<th>SunONE Example</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base DN</td>
<td>dc=acmecorp,dc=com</td>
</tr>
<tr>
<td>User DN</td>
<td>cn=Directory Manager</td>
</tr>
</tbody>
</table>

**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](#)
  - [2.2.2 Configuring an LDAP Directory Connector](#)
    - [2.2.2.1 Microsoft Active Directory](#)
      - Configuring an SSL Certificate for 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](#)
  - [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](#)
  - [2.4.4 Importing Users from CSV Files](#)
    - Configuring the CSV Importer
    - Mapping CSV Fields to Crowd Fields
    - Confirming the CSV Importer Configuration
    - Viewing the Results of the Import
  - [2.4.5 Importing Users from Atlassian Bamboo](#)

[2.4.4](#) Configuring a Custom Directory Connector

[Crowd Documentation](#)
2.2.2.3 OpenLDAP

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

Screenshot: 'Connector — OpenLDAP'

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connector</td>
<td>The directory connector to use when communicating with the directory server.</td>
</tr>
<tr>
<td>URL</td>
<td>The connection URL to use when connecting to the directory server, e.g.: ldap://localhost:389, or port 639 for SSL.</td>
</tr>
<tr>
<td>Secure SSL</td>
<td>Specifies if the connection to the directory server is a SSL connection.</td>
</tr>
<tr>
<td>Use Node Referrals</td>
<td>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.</td>
</tr>
<tr>
<td>Use Paged Results</td>
<td>Use the LDAP control extension for simple paged results option. Retrieves chunks of data rather than all of the results at once. This feature</td>
</tr>
</tbody>
</table>
may be necessary when using Microsoft Active Directory if more than 999 results are returned for any given search.

### Password Encryption
Select the type of encryption that the directory uses.

### 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**

<table>
<thead>
<tr>
<th>OpenLDAP Directory Example</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base DN</td>
<td>dc=example,dc=com</td>
</tr>
<tr>
<td>User DN</td>
<td>cn=Manager,dc=example,dc=com</td>
</tr>
</tbody>
</table>

**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
    - Configuring an SSL Certificate for 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
  - 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
  - **2.4.4 Importing Users from CSV Files**
    - 2.4.4.1 Configuring the CSV Importer
    - 2.4.4.2 Mapping CSV Fields to Crowd Fields
    - 2.4.4.3 Confirming the CSV Importer Configuration
    - 2.4.4.4 Viewing the Results of the Import
  - 2.4.5 Importing Users from Atlassian Bamboo

[Crowd Documentation](#)
2.2.2.4 Apache Directory Server (ApacheDS)

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

Screenshot: 'Connector — Apache '

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connector</td>
<td>The directory connector to use when communicating with the directory server.</td>
</tr>
<tr>
<td>URL</td>
<td>The connection URL to use when connecting to the directory server, e.g.: ldap://localhost:389, or port 639 for SSL.</td>
</tr>
<tr>
<td>Secure SSL</td>
<td>Specifies if the connection to the directory server is a SSL connection.</td>
</tr>
<tr>
<td>Use Node Referrals</td>
<td>Use the JNDI lookup java.naming.refererral option. Generally needed for Active Directory servers configured without proper DNS, to prevent a 'javax.naming.PartialResultException: Unprocessed Continuation Reference(s)' error.</td>
</tr>
<tr>
<td>Use Paged Results</td>
<td>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.</td>
</tr>
<tr>
<td><strong>Base DN</strong></td>
<td>Enter the root distinguished name to use when running queries versus the directory server, e.g.: o=acmecorp,c=com.</td>
</tr>
<tr>
<td><strong>User DN</strong></td>
<td>The username that Crowd will use when connecting to the directory server.</td>
</tr>
<tr>
<td><strong>Password</strong></td>
<td>The password that Crowd will use when connecting to the directory server.</td>
</tr>
</tbody>
</table>

**Configuration details for ApacheDS**

<table>
<thead>
<tr>
<th><strong>OpenLDAP Directory Example</strong></th>
<th><strong>Value</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Base DN</strong></td>
<td>dc=example,dc=com</td>
</tr>
</tbody>
</table>

**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](#)
    - [Configuring an SSL Certificate for 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](#)
  - [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](#)
  - [2.4.4 Importing Users from CSV Files](#)
    - [2.4.4.1 Configuring the CSV Importer](#)
    - [2.4.4.2 Mapping CSV Fields to Crowd Fields](#)
    - [2.4.4.3 Confirming the CSV Importer Configuration](#)
    - [2.4.4.4 Viewing the Results of the Import](#)
  - [2.4.5 Importing Users from Atlassian Bamboo](#)

[2.2.2 Configuring an LDAP Directory Connector](#)

[2.1 Using the Directory Browser](#)

[2.2 Adding a Directory](#)

[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](#)

[2.4.4 Importing Users from CSV Files](#)

[2.4.4.1 Configuring the CSV Importer](#)

[2.4.4.2 Mapping CSV Fields to Crowd Fields](#)

[2.4.4.3 Confirming the CSV Importer Configuration](#)

[2.4.4.4 Viewing the Results of the Import](#)

[2.4.5 Importing Users from Atlassian Bamboo](#)

[Configuring an SSL Certificate for Microsoft Active 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](#)

[2.4.4 Importing Users from CSV Files](#)

[2.4.4.1 Configuring the CSV Importer](#)

[2.4.4.2 Mapping CSV Fields to Crowd Fields](#)

[2.4.4.3 Confirming the CSV Importer Configuration](#)

[2.4.4.4 Viewing the Results of the Import](#)

[2.4.5 Importing Users from Atlassian Bamboo](#)
### 2.2.2.5 Generic LDAP Directories

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'**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Connector</strong></td>
<td>The directory connector to use when communicating with the directory server.</td>
</tr>
<tr>
<td><strong>URL</strong></td>
<td>The connection URL to use when connecting to the directory server, e.g.: <code>ldap://localhost:389</code>, or port 639 for SSL.</td>
</tr>
<tr>
<td><strong>Secure SSL</strong></td>
<td>Specifies if the connection to the directory server is a SSL connection.</td>
</tr>
<tr>
<td><strong>Use Node Referrals</strong></td>
<td>Use the JNDI lookup <code>java.naming.reerral</code> option. Generally needed for Active Directory servers configured without proper DNS, to prevent a <code>javax.naming.PartialResultException: Unprocessed Continuation Reference(s)</code> error.</td>
</tr>
<tr>
<td><strong>Use Paged Results</strong></td>
<td>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.</td>
</tr>
</tbody>
</table>
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
      - Configuring an SSL Certificate for 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
  - 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
  - 2.4.4 Importing Users from CSV Files
    - 2.4.4.1 Configuring the CSV Importer
    - 2.4.4.2 Mapping CSV Fields to Crowd Fields
    - 2.4.4.3 Confirming the CSV Importer Configuration
    - 2.4.4.4 Viewing the Results of the Import
  - 2.4.5 Importing Users from Atlassian Bamboo

Crowd Documentation
2.2.3 Configuring a Custom Directory Connector

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'

<table>
<thead>
<tr>
<th><strong>Custom Directory Store Attributes</strong></th>
<th><strong>Description</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Name</strong></td>
<td>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.</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Details about this specific directory.</td>
</tr>
<tr>
<td><strong>Active</strong></td>
<td>Only deselect this if you wish to prevent all users ('principals') within the directory from accessing all mapped applications.</td>
</tr>
<tr>
<td><strong>Implementation Class</strong></td>
<td>Implementation of com.atlassian.crowd.integration.directory.RemoteDirectory Java interface. Must be in the Crowd CLASSPATH.</td>
</tr>
</tbody>
</table>
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
    - Configuring an SSL Certificate for 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
  ° 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
  ° 2.4.4 Importing Users from CSV Files
    - 2.4.4.1 Configuring the CSV Importer
    - 2.4.4.2 Mapping CSV Fields to Crowd Fields
    - 2.4.4.3 Confirming the CSV Importer Configuration
    - 2.4.4.4 Viewing the Results of the Import
  ° 2.4.5 Importing Users from Atlassian Bamboo

Crowd Documentation
2.3 Specifying Directory Permissions

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.

Directory permissions are defined at two levels:

1. Directory-level permissions are defined on the 'Permissions' tab of the 'View Directory' screen. These permissions apply to each application mapped to the directory, unless the application has its own application-level permissions.
2. Application-level directory permissions are defined on the 'Permissions' tab of the 'View Application' screen. If a permission is enabled at directory level, you can enable it for a specific application. For example, you could enable the 'Add Principal' permission on the 'Customers' directory in JIRA but disable the permission for Confluence.

Take a look at an example.

Disabling a directory-level permission will override any permissions enabled at application level. If a permission is enabled at application level and then subsequently disabled at directory level, the directory-level permission will apply. (The application-level permissions will be 'remembered' and will apply again if re-enabled at directory level.)

How do directory permissions affect the Crowd application (Crowd Administration Console)?

- If a particular permission is turned off at directory level, then no application can perform the related function - not even the Crowd application. So, for example, if you disable the 'Remove Principal' permission for a directory, then the Crowd Administration Console will not allow you to delete a principal from that directory.
- The Crowd application is not bound by application-level permissions.

Below, we tell you about directory-level permissions. You can also read more about application-level directory permissions.

**Directory-Level Permissions**

<table>
<thead>
<tr>
<th>Permission</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add Group</td>
<td>Allows applications to add groups to the directory.</td>
</tr>
<tr>
<td>Add Principal</td>
<td>Allows applications to add principals to the directory.</td>
</tr>
<tr>
<td>Add Role</td>
<td>Allows applications to add roles to the directory.</td>
</tr>
<tr>
<td>Modify Group</td>
<td>Allows applications to modify groups in the directory.</td>
</tr>
<tr>
<td>Modify Principal</td>
<td>Allows applications to modify principals in the directory.</td>
</tr>
<tr>
<td>Modify Role</td>
<td>Allows applications to modify roles in the directory.</td>
</tr>
<tr>
<td>Remove Group</td>
<td>Allows applications to delete groups from the directory.</td>
</tr>
<tr>
<td>Remove Principal</td>
<td>Allows applications to delete principals from the directory.</td>
</tr>
</tbody>
</table>
Consider carefully whether you allow the deletion of principals, as some applications contain historical data, e.g. documents that the user has created. Read more.

| Remove Role | Allows applications to delete roles from the directory. |

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

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.

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

- 3.3.2 Specifying an Application’s Directory Permissions
- 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
- Configuring an SSL Certificate for 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
- 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
  - 2.4.4 Importing Users from CSV Files
    - 2.4.4.1 Configuring the CSV Importer
    - 2.4.4.2 Mapping CSV Fields to Crowd Fields
    - 2.4.4.3 Confirming the CSV Importer Configuration
    - 2.4.4.4 Viewing the Results of the Import
  - 2.4.5 Importing Users from Atlassian Bamboo

Crowd Documentation
2.4 Importing Principals and Groups into a Directory

Once you have added a directory, you can import groups and users (or 'principals', as they are known in Crowd) 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 or file from which you want to import external users into Crowd:
   - 'Atlassian Importer' — see 2.4.1 Importing Users from Atlassian Confluence, 2.4.2 Importing Users from Atlassian JIRA and 2.4.5 Importing Users from Atlassian Bamboo
   - 'CSV Importer' — see 2.4.4 Importing Users from CSV Files
   - 'JIVE' — 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
      - Configuring an SSL Certificate for 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
  - 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
2.4.4 Importing Users from CSV Files
   - 2.4.4.1 Configuring the CSV Importer
   - 2.4.4.2 Mapping CSV Fields to Crowd Fields
   - 2.4.4.3 Confirming the CSV Importer Configuration
   - 2.4.4.4 Viewing the Results of the Import
2.4.5 Importing Users from Atlassian Bamboo
2.4.1 Importing Users from Atlassian Confluence

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.

It is recommended that you import your Confluence users into an Internal Directory that has its 'Password Encryption' set to 'ATLASSIAN-SHA1'. Otherwise, users' passwords will not be copied across to Crowd.

**Before you begin**

You will need to have installed the Confluence instance's database JDBC driver in the Crowd CLASS-PATH.

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

1. Log in to the Crowd Administration Console.
2. Click the 'Directories' link in the top navigation bar.
3. This will display the Directory Browser. Click the 'Import Users' link.
4. This will display the 'Import Type' screen. Click the 'Atlassian Importer' button.
5. This will display the 'Options' screen. Complete the fields as follows:
   - 'Atlassian Product' — Select 'Confluence'.
   - 'Directory' — Select the directory that is mapped to the Confluence application.
   - 'Import Passwords' — Select this checkbox if you wish to import the users' passwords from Confluence. You can only import passwords if the Crowd directory is using the 'Atlassian SHA1' encryption method.
   - 'Product Database URL' — Type the URL of your Confluence instance's database. The exact syntax will depend on which database you are using; see Database Configuration in the Confluence Configuration Guide.
   - 'Database Driver' — type the name of your Confluence instance's database JDBC driver (e.g. for MYSQL, type `com.mysql.jdbc.Driver`).
   - 'Username' — Type the username of the database user that Crowd will use to login to your Confluence instance's database.
   - 'Password' — Type the password of the database user Crowd will use to login to your Confluence instance's database.

   The import process will log in to the database, not into Confluence.
6. Click the 'Continue' button to import the users from your Confluence instance into your Crowd directory.
7. The 'Results' 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
    - Configuring an SSL Certificate for 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
  - 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
  - 2.4.4 Importing Users from CSV Files
    - 2.4.4.1 Configuring the CSV Importer
    - 2.4.4.2 Mapping CSV Fields to Crowd Fields
    - 2.4.4.3 Confirming the CSV Importer Configuration
    - 2.4.4.4 Viewing the Results of the Import
  - 2.4.5 Importing Users from Atlassian Bamboo

Crowd Documentation
2.4.2 Importing Users from Atlassian JIRA

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.

It is recommended that you import your JIRA users into an Internal Directory that has its 'Password Encryption' set to 'ATLASSIAN-SHA1'. Otherwise, users' passwords will not be copied across to Crowd.

**Before you begin**

You will need to have installed the JIRA instance's database JDBC driver in the Crowd CLASS-PATH.

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

1. Log in to the Crowd Administration Console.
2. Click the 'Directories' link in the top navigation bar.
3. This will display the Directory Browser. Click the 'Import Users' link.
4. This will display the 'Import Type' screen. Click the 'Atlassian Importer' button.
5. This will display the 'Options' screen. Complete the fields as follows:
   - 'Atlassian Product' — Select 'JIRA'.
   - 'Directory' — Select the directory that is mapped to the JIRA application.
   - 'Import Passwords' — Select this checkbox if you wish to import the users' passwords from JIRA. You can only import passwords if the Crowd directory is using the 'Atlassian SHA1' encryption method.
   - 'Product Database URL' — Type the URL of your JIRA instance's database. The exact syntax will depend on which database you are using; see Connecting JIRA to a Database in the JIRA Installation Guide.
   - 'Database Driver' — Type the name of your JIRA instance's database JDBC driver (e.g. for MySQL, type com.mysql.jdbc.Driver).
   - 'Username' — Type the username of the database user that Crowd will use to log in to your JIRA instance's database.
   - 'Password' — Type the password of the database user Crowd will use to log in to your JIRA instance's database.

   The import process will log in to the database, not into JIRA.

6. Click the 'Continue' button to import the users from your JIRA instance into your Crowd directory.
7. The 'Results' 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
    - Configuring an SSL Certificate for 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
  - 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
  - 2.4.4 Importing Users from CSV Files
    - 2.4.4.1 Configuring the CSV Importer
    - 2.4.4.2 Mapping CSV Fields to Crowd Fields
    - 2.4.4.3 Confirming the CSV Importer Configuration
    - 2.4.4.4 Viewing the Results of the Import
  - 2.4.5 Importing Users from Atlassian Bamboo

Crowd Documentation
2.4.3 Importing Users from Jive Forums

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 'Directories' link in the top navigation bar.
3. This will display the Directory Browser. Click the 'Import Users' link.
4. This will display the 'Import Type' screen. Click the 'JIVE' 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 Configuring an SSL Certificate for Microsoft Active Directory
    - 2.2.2.3 SunONE
    - 2.2.2.4 Apache Directory Server (ApacheDS)
    - 2.2.2.5 Generic LDAP Directories
  - 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
  - 2.4.4 Importing Users from CSV Files
    - 2.4.4.1 Configuring the CSV Importer
    - 2.4.4.2 Mapping CSV Fields to Crowd Fields
    - 2.4.4.3 Confirming the CSV Importer Configuration
    - 2.4.4.4 Viewing the Results of the Import
  - 2.4.5 Importing Users from Atlassian Bamboo

Crowd Documentation
2.4.4 Importing Users from CSV Files

You can copy users from an external directory or user base into Crowd via a CSV (comma-separated values) file. There are two phases involved:

1. Export your existing users and their group memberships from your external directory into a CSV file or files.
2. Import the users, groups and group memberships into a Crowd directory from the CSV files.

The CSV importer is available with Crowd 1.1.1 and later.

Preparing your CSV Files

You will need:

• a CSV file containing user information, and
• optionally, another CSV file containing group memberships.

Attached are simple examples of the CSV files:

• Example user CSV file
• Example group membership CSV file

The CSV Importer's 'File Mappings' screen allows you to match the CSV fields to Crowd's Principal and Group fields.

Formatting and location of the CSV files:

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Location</td>
<td>The CSV files must be on the local drive (e.g. C:) of the Crowd server.</td>
</tr>
<tr>
<td>Supported attributes</td>
<td>The CSV Importer does not support custom attributes. The supported attributes are shown in the drop-down lists on the 'File Mappings' screen.</td>
</tr>
<tr>
<td>Header row</td>
<td>The first row in each CSV file must be a header row. The CSV Importer will not import the information in the first row. The information in the first row is displayed in the column labelled 'CSV Header Row' on the 'File Mappings' screen.</td>
</tr>
<tr>
<td>Delimiter</td>
<td>The fields in the CSV file must be separated by a single-character delimiter. The CSV Importer's 'Configuration' screen lets you tell Crowd which delimiter you have used.</td>
</tr>
<tr>
<td>Passwords</td>
<td>You will need to decide whether to import your passwords into Crowd. And if you do import the passwords, you must choose to import them as either encrypted or clear text. Check the password encryption in the directory you are exporting users from, and compare it with the encryption method of the Crowd directory you want to import the users into. You can use Crowd's Directory Browser to view the directory's configuration details, including the encryption method. The CSV Importer's 'Configuration' screen</td>
</tr>
</tbody>
</table>
screen lets you tell the CSV Importer whether to encrypt the passwords.

To export information from your user directory into a CSV file,

1. Export the users from your external user directory or database into a CSV file. Your directory or user base should have an option to allow you to do this.
2. If you want to copy your existing group memberships into Crowd, export the groups and group memberships into another CSV file.

Importing the CSV Files into Crowd

Once you have prepared your CSV file(s), you can import the users and groups into a Crowd directory.

To import users and groups from CSV files,

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 'Import Users' link.
4. This will display the 'Import Type' screen. Click the 'CSV Importer' button.
5. This will display the 'Configuration' tab of the 'CSV Importer'.
6. Enter the details of the CSV files as described in 'Configuring the CSV Importer'.

Related Topics

- 2.4.4.1 Configuring the CSV Importer
- 2.4.4.2 Mapping CSV Fields to Crowd Fields
- 2.4.4.3 Confirming the CSV Importer Configuration
- 2.4.4.4 Viewing the Results of the Import

Crowd Documentation
2.4.4.1 Configuring the CSV Importer

Once you have started the CSV Importer, the 'Configuration' screen allows you to specify information about the Crowd directory and CSV file(s) involved in the import.

Refer to information on preparing your CSV files.

To configure the CSV importer,

1. Start the CSV Importer.
2. This will display the 'Configuration' screen. Complete the fields as follows:
   - 'Directory' — Select the Crowd user directory into which you want to import the users.
   - 'Are your passwords encrypted?' — Select 'Yes' if the passwords in your CSV file are already encrypted. Crowd will not re-encrypt the passwords during the import. Select 'No' if the passwords in your CSV file are not encrypted. Crowd will encrypt the passwords during the import, using the encryption method of the Crowd directory you are importing into.
   - 'Delimiter' — Type the single-character delimiter used to separate the fields in your CSV file(s).
   - 'User File' — Type the location of the CSV file containing the users you wish to import.
   - 'Group Membership File' — If you want to import groups and group memberships of your users, type the location of the CSV file containing the group membership information.
3. Click the 'Continue' button to map the CSV fields to the Crowd directory fields.

Screenshot: 'CSV Importer - Configuration'

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
    - Configuring an SSL Certificate for 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
  - 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
  ° 2.4.4 Importing Users from CSV Files
    - 2.4.4.1 Configuring the CSV Importer
    - 2.4.4.2 Mapping CSV Fields to Crowd Fields
    - 2.4.4.3 Confirming the CSV Importer Configuration
    - 2.4.4.4 Viewing the Results of the Import
  ° 2.4.5 Importing Users from Atlassian Bamboo

Crowd Documentation
2.4.4.2 Mapping CSV Fields to Crowd Fields

Once you have entered details on the Configuration screen of the CSV Importer, the 'File Mappings' screen allows you to match the CSV fields to the Principal and Group fields in Crowd. Crowd will use these mappings to import the information from the CSV file(s) into your Crowd directory.

Refer to information on preparing your CSV files.

The 'File Mappings' screen has two main sections:

- 'User Mappings' — Use this section to map the fields in your 'User' CSV file.
- 'Group Mappings' — Use this section to map the fields in your 'Group Membership' CSV file, if you have one. This section will only appear if you have specified a 'Group Membership File' on the Configuration screen.

Each section has the following columns:

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSV Header Row</td>
<td>This column shows the text from each field in the first row of your CSV file. The CSV Importer assumes that the first row is a header row.</td>
</tr>
<tr>
<td>Sample Row</td>
<td>This column shows the text from each field in the second row of your CSV file. This is done to help you with the mapping process.</td>
</tr>
<tr>
<td>Mapping</td>
<td>Each row in this column contains a drop-down list of the Crowd field names available for mapping. To map a Crowd field to a CSV field, select the appropriate Crowd field name from the drop-down list to match the CSV field shown in the 'CSV Header Row' column.</td>
</tr>
</tbody>
</table>

In the 'User Mappings' section, the 'Mapping' drop-down lists contain the following Crowd field names:

<table>
<thead>
<tr>
<th>Crowd field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>First Name</td>
<td>Required. One of the rows on the screen must map this value to the CSV field containing the users' first names.</td>
</tr>
<tr>
<td>Last Name</td>
<td>Required. One of the rows on the screen must map this value to the CSV field containing the users' last names.</td>
</tr>
<tr>
<td>Email Address</td>
<td>Required. One of the rows on the screen must map this value to the CSV field containing the users' email addresses.</td>
</tr>
<tr>
<td>Username</td>
<td>Required. One of the rows on the screen must map this value to the CSV field containing the usernames.</td>
</tr>
<tr>
<td>Password</td>
<td>If your CSV file contains passwords, map this value to the CSV field containing the passwords.</td>
</tr>
<tr>
<td>None</td>
<td>Select 'None' if the CSV field displayed under 'CSV Header Row' is not to be mapped to any Crowd</td>
</tr>
</tbody>
</table>
fields. These CSV fields will not be imported into Crowd.

In the 'Group Mappings' section (if present), the 'Mapping' drop-down lists contain the following Crowd field names:

<table>
<thead>
<tr>
<th>Crowd field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group Name</td>
<td>Required. One of the rows on the screen must map this value to the CSV field containing the names of the groups.</td>
</tr>
<tr>
<td>Username</td>
<td>Required. One of the rows on the screen must map this value to the CSV field containing the usernames.</td>
</tr>
<tr>
<td>None</td>
<td>Select 'None' if the CSV field displayed under 'CSV Header Row' is not to be mapped to any Crowd fields. These CSV fields will not be imported into Crowd.</td>
</tr>
</tbody>
</table>

To map the CSV fields to Crowd fields,

1. Start the [CSV Importer](#).
2. Complete the details on the 'Configuration screen' and click the 'Continue' button.
3. This will display the 'File Mappings' screen. Complete the mappings in the 'User Mappings' section as follows:
   - In the 'CSV Header Row' column, find the field which contains your users' first names — select 'First Name' from the drop-down list in the 'Mapping' column.
   - In the 'CSV Header Row' column, find the field which contains your users' last names — select 'Last Name' from the drop-down list in the 'Mapping' column.
   - In the 'CSV Header Row' column, find the field which contains your users' email addresses — select 'Email Address' from the drop-down list in the 'Mapping' column.
   - In the 'CSV Header Row' column, find the field which contains the usernames — select 'Username' from the drop-down list in the 'Mapping' column.
   - In the 'CSV Header Row' column, find the field which contains your users' passwords — select 'Password' from the drop-down list in the 'Mapping' column.
   - Select 'None' from the drop-down lists for all unmatched rows.
4. Complete the mappings in the 'Group Mappings' section (if present) as follows:
   - In the 'CSV Header Row' column, find the field which contains the group names — select 'Group Name' from the drop-down list in the 'Mapping' column.
   - In the 'CSV Header Row' column, find the field which contains the usernames — select 'Username' from the drop-down list in the 'Mapping' column.
   - Select 'None' from the drop-down lists for all unmatched rows.
5. Click the 'Continue' button to confirm the CSV configuration.

[Screenshot: 'CSV Importer - File Mappings']
Related Topics

- 2.4.4.1 Configuring the CSV Importer
- 2.4.4.2 Mapping CSV Fields to Crowd Fields
- 2.4.4.3 Confirming the CSV Importer Configuration
- 2.4.4.4 Viewing the Results of the Import

Crowd Documentation
2.4.4.3 Confirming the CSV Importer Configuration

The 'Confirmation' screen allows you to review your configuration and mapping before performing the CSV import.

To confirm the CSV configuration and mapping,

1. Review the information shown on the 'Confirmation' screen.
2. Click the 'Continue' button to import the users from your CSV file into your Crowd directory.
3. Once the import is complete, Crowd will display the 'Results' screen.

Screenshot: 'CSV Importer - Confirmation'

Related Topics

- 2.4.4.1 Configuring the CSV Importer
- 2.4.4.2 Mapping CSV Fields to Crowd Fields
- 2.4.4.3 Confirming the CSV Importer Configuration
- 2.4.4.4 Viewing the Results of the Import

Crowd Documentation
2.4.4.4 Viewing the Results of the Import

The 'Results' screen shows the outcome of the CSV import.

The CSV Importer adds to the Crowd directory, but does not update or delete existing information:

- If the Username already exists in Crowd, the CSV Importer does not overwrite the information for that user even if the Username exists in the CSV file with different user information.
- The CSV Importer does not remove users from Crowd.
- If your 'Group Membership' CSV file contains additional group(s) for a user, the additional group(s) and group membership(s) will be imported.
- Existing group memberships will not be changed or removed.
- The 'Results' screen will show number of duplicate usernames in the CSV file which were ignored i.e. not imported.
- The 'Results' screen will show number of duplicate group names in the CSV file which were ignored i.e. not imported.

Screenshot: 'CSV Importer - Results'

<table>
<thead>
<tr>
<th>CSV Importer</th>
</tr>
</thead>
</table>

Below are the results of your import, if there are failures please consult the log files.

<table>
<thead>
<tr>
<th>Users imported:</th>
<th>Users that already exist:</th>
<th>Groups imported:</th>
<th>Groups that already exist:</th>
<th>Memberships imported:</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>justinluke,jeesmith,nobody</td>
<td>0</td>
<td>confluence-admins, xi-admins, admins, jira-admins</td>
<td>5</td>
</tr>
</tbody>
</table>

Related Topics

- 2.4.4.1 Configuring the CSV Importer
- 2.4.4.2 Mapping CSV Fields to Crowd Fields
- 2.4.4.3 Confirming the CSV Importer Configuration
- 2.4.4.4 Viewing the Results of the Import

Crowd Documentation
2.4.5 Importing Users from Atlassian Bamboo

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

We recommend that you import your Bamboo users into an internal Crowd directory that has its 'Password Encryption' set to 'ATLASSIAN-SHA1'. Otherwise, users' passwords will not be copied across to Crowd.

⚠️ Before you begin

You will need to have installed the Bamboo instance's database JDBC driver in the Crowd CLASS-PATH.

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

1. Log in to the Crowd Administration Console.
2. Click the 'Directories' link in the top navigation bar.
3. This will display the Directory Browser. Click the 'Import Users' link.
4. This will display the 'Import Type' screen. Click the 'Atlassian Importer' button.
5. This will display the 'Options' screen. Complete the fields as follows:
   - 'Atlassian Product' — Select 'Bamboo'.
   - 'Directory' — Select the directory that you have mapped to the Bamboo application in Crowd.
   - 'Import Passwords' — Select this checkbox if you wish to import the users' passwords from Bamboo. You can only import passwords if the Crowd directory is using the 'Atlassian SHA1' encryption method.
   - 'Product Database URL' — Type the URL of your Bamboo instance's database. The exact syntax will depend on which database you are using. See Database Configuration in the Bamboo Installation Guide.
   - 'Database Driver' — Type the name of your Bamboo instance's database JDBC driver (e.g. for MYSQL, type com.mysql.jdbc.Driver).
   - 'Username' — Type the username of the database user that Crowd will use to log in to your Bamboo instance's database.
   - 'Password' — Type the password of the database user Crowd will use to log in to your Bamboo instance's database.

⚠️ The import process will log in to the database, not into Bamboo.
6. Click the 'Continue' button to import the users from your Bamboo instance into your Crowd directory.
7. The 'Results' 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 Bamboo Users'
Next Step

To give the imported groups access to the Bamboo 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
      - Configuring an SSL Certificate for 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
  - 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
  - 2.4.4 Importing Users from CSV Files
    - 2.4.4.1 Configuring the CSV Importer
    - 2.4.4.2 Mapping CSV Fields to Crowd Fields
    - 2.4.4.3 Confirming the CSV Importer Configuration
    - 2.4.4.4 Viewing the Results of the Import
  - 2.4.5 Importing Users from Atlassian Bamboo
3. Managing 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.

- 3.1 Using the Application Browser
- 3.2 Adding an Application
  - 3.2.01 Integrating Crowd with Apache
  - 3.2.02 Integrating Crowd with Subversion
  - 3.2.03 Integrating Crowd with Atlassian Confluence
    - 3.2.3.1 Configuring Confluence for NTLM SSO
  - 3.2.04 Integrating Crowd with Atlassian CrowdID
  - 3.2.05 Integrating Crowd with Atlassian FishEye
  - 3.2.06 Integrating Crowd with Atlassian JIRA
    - 3.2.6.1 Configuring JIRA for NTLM SSO
  - 3.2.07 Integrating Crowd with Jive Forums
    - 3.2.7.1 Jive SSO
  - 3.2.08 Integrating Crowd with Atlassian Bamboo
  - 3.2.09 Integrating Crowd with Acegi Security
    - Integrating AppFuse - a Crowd-Acegi Integration Tutorial
  - 3.2.10 Integrating Crowd with a Custom Application
- 3.3 Mapping a Directory to an Application
  - 3.3.1 Specifying the Directory Order for an Application
  - 3.3.2 Specifying an Application's Directory Permissions
    - Example of Directory Permissions
- 3.4 Specifying which Groups can access an Application
- 3.5 Specifying an Application's Address or Hostname
- 3.6 Testing a User's Login to an Application
- 3.7 Managing an Application's Session
- 3.8 Deleting or Deactivating an Application
3.1 Using the Application Browser

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 three 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) configured during setup. Its main purpose is to provide an example of how to integrate custom applications with Crowd. To access the 'demo' application, go to http://localhost:8095/demo.
- 'crowdid' — this is the 'CrowdID application which you (optionally) configured during setup. It allows you to provide OpenID services to your end-users. For details please see the CrowdID Administration Guide and the CrowdID User Guide. To access CrowdID, go to http://localhost:8095/openidserver.

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 application's details, click the 'View' link.

Screenshot 1: 'Application Browser'

<table>
<thead>
<tr>
<th>Name</th>
<th>Applications</th>
<th>Principals</th>
<th>Groups</th>
<th>Roles</th>
<th>Sessions</th>
<th>Directories</th>
<th>Options</th>
<th>System Info</th>
<th>Backup</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Active</th>
<th>All</th>
<th>Description</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>confuse</td>
<td>true</td>
<td></td>
<td>Allassen Confluence wiki</td>
<td>View</td>
</tr>
<tr>
<td>user</td>
<td>true</td>
<td></td>
<td>Allassen User</td>
<td>View</td>
</tr>
<tr>
<td>crowd-openid-service</td>
<td>true</td>
<td></td>
<td>Crowd OpenID Service</td>
<td>View</td>
</tr>
<tr>
<td>demo</td>
<td>true</td>
<td></td>
<td>Crowd demo application</td>
<td>View</td>
</tr>
</tbody>
</table>

Screenshot 2: 'View Application'
Related Topics

- 3.1 Using the Application Browser
- 3.2 Adding an Application
  - 3.2.01 Integrating Crowd with Apache
  - 3.2.02 Integrating Crowd with Subversion
  - 3.2.03 Integrating Crowd with Atlassian Confluence
    - 3.2.3.1 Configuring Confluence for NTLM SSO
  - 3.2.04 Integrating Crowd with Atlassian CrowdID
  - 3.2.05 Integrating Crowd with Atlassian FishEye
  - 3.2.06 Integrating Crowd with Atlassian JIRA
    - 3.2.6.1 Configuring JIRA for NTLM SSO
  - 3.2.07 Integrating Crowd with Jive Forums
    - 3.2.7.1 Jive SSO
  - 3.2.08 Integrating Crowd with Atlassian Bamboo
  - 3.2.09 Integrating Crowd with Acegi Security
    - Integrating AppFuse - a Crowd-Acegi Integration Tutorial
  - 3.2.10 Integrating Crowd with a Custom Application
- 3.3 Mapping a Directory to an Application
  - 3.3.1 Specifying the Directory Order for an Application
  - 3.3.2 Specifying an Application’s Directory Permissions
    - Example of Directory Permissions
- 3.4 Specifying which Groups can access an Application
- 3.5 Specifying an Application’s Address or Hostname
- 3.6 Testing a User's Login to an Application
- 3.7 Managing an Application's Session
- 3.8 Deleting or Deactivating an Application

Crowd Documentation
3.2 Adding an Application

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:
  - 3.2.01 Integrating Crowd with Apache
  - 3.2.02 Integrating Crowd with Subversion
  - 3.2.03 Integrating Crowd with Atlassian Confluence
  - 3.2.04 Integrating Crowd with Atlassian CrowdID
  - 3.2.05 Integrating Crowd with Atlassian FishEye
  - 3.2.06 Integrating Crowd with Atlassian JIRA
  - 3.2.07 Integrating Crowd with Jive Forums
  - 3.2.08 Integrating Crowd with Atlassian Bamboo
  - 3.2.09 Integrating Crowd with Acegi Security
  - 3.2.10 Integrating Crowd with a Custom Application

To add an application to Crowd,

1. Log in 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 'Permissions' tab and set the directory permissions for the application.
8. Click the 'Remote Addresses' tab and specify the IP address or hostname of the application. (The default is 'localhost'.)
9. Click the 'Update' button to save your changes.
10. If you wish, you can click the 'Config Test' tab and verify that a user can log in to the application.
Add Application

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>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.</td>
</tr>
<tr>
<td>Description</td>
<td>A short description of the application. Note: A web URL is often helpful.</td>
</tr>
<tr>
<td>Active</td>
<td>Only deselect this if you wish to prevent all users (from all directories) from accessing this application.</td>
</tr>
<tr>
<td>Password</td>
<td>The password which the application will use when it authenticates against the Crowd framework as a client.</td>
</tr>
<tr>
<td>Default Directory</td>
<td>A directory that contains relevant users. Note: Additional directories can be added later.</td>
</tr>
</tbody>
</table>

Next Steps

After adding an application, you may want to:
- map additional directories to the application, and
- set each directory's permissions for the application.

Related Topics

- 3.1 Using the Application Browser
- 3.2 Adding an Application
  - 3.2.01 Integrating Crowd with Apache
  - 3.2.02 Integrating Crowd with Subversion
  - 3.2.03 Integrating Crowd with Atlassian Confluence
    - 3.2.3.1 Configuring Confluence for NTLM SSO
  - 3.2.04 Integrating Crowd with Atlassian CrowdID
  - 3.2.05 Integrating Crowd with Atlassian FishEye
  - 3.2.06 Integrating Crowd with Atlassian JIRA
    - 3.2.6.1 Configuring JIRA for NTLM SSO
  - 3.2.07 Integrating Crowd with Jive Forums
    - 3.2.7.1 Jive SSO
  - 3.2.08 Integrating Crowd with Atlassian Bamboo
  - 3.2.09 Integrating Crowd with Acegi Security
- Integrating AppFuse - a Crowd-Acegi Integration Tutorial
  ° 3.2.10 Integrating Crowd with a Custom Application
- 3.3 Mapping a Directory to an Application
  ° 3.3.1 Specifying the Directory Order for an Application
  ° 3.3.2 Specifying an Application’s Directory Permissions
  - Example of Directory Permissions
- 3.4 Specifying which Groups can access an Application
- 3.5 Specifying an Application’s Address or Hostname
- 3.6 Testing a User’s Login to an Application
- 3.7 Managing an Application’s Session
- 3.8 Deleting or Deactivating an Application

Crowd Documentation
3.2.01 Integrating Crowd with Apache

Crowd provides a number of modules that allow you to configure Crowd to authenticate HTTP Authentication requests made to an Apache web server.

The following features are provided:

- Use Crowd to password-protect resources on your website.
- Configure website locations to restrict access to specific Crowd groups or users.

Note: These instructions assume some Unix system and Apache configuration knowledge.

Using Subversion under Apache?

Crowd's Subversion connector allows you to password-protect a Subversion repository and provide fine grained access by group or user. Read more.

Prerequisites

- Apache web server version 2.0 or above with the mod_perl module installed and configured.
- The following third-party Perl modules:
  - SOAP::Lite (v0.69 or greater required)
  - Digest::SHA1
  - Error
  - Cache::Cache

Installation and Configuration

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

Installing the Third-Party Perl Modules

Download the required Perl modules from CPAN using the links above and install them as follows:

```bash
tar xvzf Cache-Cache-1.05.tar.gz
cd Cache-Cache-1.05
perl Makefile.PL
make
make install
```

See [http://search.cpan.org/~jhi/perl-5.8.0/pod/perlmodinstall.pod](http://search.cpan.org/~jhi/perl-5.8.0/pod/perlmodinstall.pod) for a detailed description of the various ways of installing Perl modules on your system.

Installing the Crowd Perl Modules

1. Download the three Crowd module files attached to this page:

<table>
<thead>
<tr>
<th>Attached file</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.2.01 Integrating Crowd with Apache^Crowd-</td>
<td>Crowd authentication, authorisation and perl module for Apache 2</td>
</tr>
<tr>
<td>Apache-Connector-1.2.zip</td>
<td></td>
</tr>
</tbody>
</table>

2. Extract the Crowd-Apache-Connection archive file and install the three modules using the same procedure as for the third party modules. Crowd authentication module for Apache 2:

```bash
unzip Crowd-Apache-Connector-1.2.zip
cd Crowd-Apache-Connector-1.2/Apache-CrowdAuth-1.2/
perl Makefile.PL
make
```
make install

Crowd authorisation module for Apache 2:

unzip Crowd-Apache-Connector-1.2.zip
cd Crowd-Apache-Connector-1.2/Apache-CrowdAuth-1.2/
perl Makefile.PL
make
make install

Perl interface to Crowd's SOAP API (required by other modules):

unzip Crowd-Apache-Connector-1.2.zip
cd Crowd-Apache-Connector-1.2/Atlassian-Crowd-1.2/
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.

Configuring 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:8095/crowd/services/SecurityServer
  PerlSetVar CrowdCacheEnabled on
  PerlSetVar CrowdCacheLocation /tmp/CrowdAuth
  PerlSetVar CrowdCacheExpiry 300

  require valid-user

</Directory>

<table>
<thead>
<tr>
<th>Command</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AuthName crowd</td>
<td>Defines the <strong>realm</strong> of the authentication. This information is typically provided to the user in the dialog box popped up by their browser</td>
</tr>
<tr>
<td><strong>AuthType</strong> Basic</td>
<td>Tells apache to use basic authentication</td>
</tr>
<tr>
<td>PerlAuthenHandler Apache::CrowdAuth</td>
<td>Tells Apache to delegate authentication to the CrowdAuth module</td>
</tr>
<tr>
<td>Command</td>
<td>Explanation</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td>PerlAuthzHandler Apache::CrowdAuthz</td>
<td>Tells Apache to use the Apache::CrowdAuthz for authorisation</td>
</tr>
<tr>
<td>PerlSetVar CrowdAllowedUsers johnh,kevinr</td>
<td>Allow only the users johnh or kevinr to access the location</td>
</tr>
<tr>
<td>PerlSetVar CrowdAllowedGroups developers,crowd-administrators</td>
<td>Allow only members of the developers or crowd-administrators groups to access the location</td>
</tr>
</tbody>
</table>

**Configuring Authorisation**

If you want to restrict access to a certain Directory or Location in your Apache configuration to a subset of Crowd users and/or groups, add the following lines to your configuration:

```
<Location>

PerlAuthzHandler Apache::CrowdAuthz

PerlSetVar CrowdAllowedUsers johnh,kevinr
PerlSetVar CrowdAllowedGroups developers,crowd-administrators

</Location>
```

**Note:**

- Typically, only one of the CrowdAllowedUsers or CrowdAllowedGroups would be needed for a particular location. You can define both. If you do, then access is granted if either is satisfied.
- If the CrowdCacheEnabled setting is on, then authorisation checks are cached in order to increase performance. This means that changes to group membership in Crowd may not be reflected immediately in user access.
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.

<table>
<thead>
<tr>
<th>Apache Log Error Message</th>
<th>Possible Cause and Next Steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>CrowdAppName or CrowdAppPassword is not defined</td>
<td>One or both of the CrowdAppName or CrowdAppPassword parameters is missing from the Apache config file.</td>
</tr>
<tr>
<td>Failed to authenticate application</td>
<td>The attempt to authenticate the application with Crowd failed. Check the values of the CrowdAppName or CrowdAppPassword parameters.</td>
</tr>
<tr>
<td>Failed to authenticate principal</td>
<td>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 successful authentications unless the LogLevel is info or above.</td>
</tr>
<tr>
<td>User token not found in SOAP response for user &lt;user&gt;</td>
<td>Internal SOAP protocol error.</td>
</tr>
<tr>
<td>error 500...at CrowdAuth.pm...</td>
<td>Indicates that Apache can’t connect to the Crowd SOAP service.</td>
</tr>
<tr>
<td>error 404...at CrowdAuth.pm...</td>
<td>Indicates that the URL used to connect to the Crowd SOAP service is incorrect. Check the value of the CrowdSOAPURL parameter.</td>
</tr>
<tr>
<td>failed to resolve handler 'Apache::CrowdAuth': Can't locate Apache/CrowdAuth.pm ...</td>
<td>The CrowdAuth.pm file isn't located on the Perl include path (or it has incorrect permissions).</td>
</tr>
<tr>
<td>failed to resolve handler 'Apache::CrowdAuth': Can't locate SOAP/Lite.pm...</td>
<td>The SOAP:Lite module hasn't been installed.</td>
</tr>
<tr>
<td>Can't locate object method &quot;call&quot; via package &quot;SOAP::SOM&quot; at ...</td>
<td>This message indicates a missing or old installation of SOAP::Lite. Try installing (or reinstalling) version 0.69 SOAP::Lite. On Windows, you will get this error if you haven't manually upgraded the SOAP::Lite ppm (see below)</td>
</tr>
</tbody>
</table>

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 Perl Modules on Windows

The required modules (Digest::SHA1, Error, Cache::FileCache, SOAP::Lite) are available through the Perl Package Manager utility.

CrowdAuth needs a newer version of SOAP::Lite than is supplied with ActivePerl (0.69 vs 0.55). A prebuilt ppm of the correct version can be installed from the University of Winnipeg's repository using the following command:

C:\>ppm install http://theoryx5.uwinnipeg.ca/ppms/SOAP-Lite.ppd

Installing Apache::CrowdAuth on Windows

Extract Apache-CrowdAuth-0.06.zip using Winzip or equivalent...

cd Apache-CrowdAuth-0.06
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
  - 3.2.01 Integrating Crowd with Apache
  - 3.2.02 Integrating Crowd with Subversion
  - 3.2.03 Integrating Crowd with Atlassian Confluence
    - 3.2.3.1 Configuring Confluence for NTLM SSO
  - 3.2.04 Integrating Crowd with Atlassian CrowdID
  - 3.2.05 Integrating Crowd with Atlassian FishEye
  - 3.2.06 Integrating Crowd with Atlassian JIRA
    - 3.2.6.1 Configuring JIRA for NTLM SSO
  - 3.2.07 Integrating Crowd with Jive Forums
    - 3.2.7.1 Jive SSO
  - 3.2.08 Integrating Crowd with Atlassian Bamboo
  - 3.2.09 Integrating Crowd with Acegi Security
    - Integrating AppFuse - a Crowd-Acegi Integration Tutorial
  - 3.2.10 Integrating Crowd with a Custom Application
- 3.3 Mapping a Directory to an Application
  - 3.3.1 Specifying the Directory Order for an Application
  - 3.3.2 Specifying an Application’s Directory Permissions
    - Example of Directory Permissions
- 3.4 Specifying which Groups can access an Application
- 3.5 Specifying an Application’s Address or Hostname
- 3.6 Testing a User’s Login to an Application
- 3.7 Managing an Application’s Session
- 3.8 Deleting or Deactivating an Application

Crowd Documentation
3.2.02 Integrating Crowd with Subversion

This page last changed on Nov 26, 2007 by smaddox.

⚠️ Install the Crowd Apache connector first

To use the Subversion connector, you will need to have the Crowd Apache Connector already installed.

Crowd's Subversion connector allows you to password-protect a Subversion repository and provide fine grained access by group or user.

Prerequisites

- Crowd Apache Connector.

Configuring Crowd Authentication for Subversion

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

Example:

```
<Location /svn>
    # Uncomment this to enable the repository, DAV svn
    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:8095/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.

Configuring Crowd Authorisation for Subversion

To restrict Subversion repository access to certain groups and/or users, you can use the Apache::CrowdAuthz module and the CrowdAllowedGroups and CrowdAllowedUsers directives.

For more fine-grained access, the CrowdAuthzSVNAccessFile is provided. For example:

```
PerlSetVar CrowdAuthzSVNAccessFile /etc/apache2/dav_svn.authz
```

The CrowdAuthzSVNAccessFile setting lets you define a file where you can configure group and user access on a per-directory level.
The format of the file is the same as that used by Subversion's own authorisation module, mod_authz_svn. Here's a small example:

```bash
# Everyone has read access to the repository
# (unless modified below).
[/]
* = r

# Members of the bazdevelopers group can
# read and write to the BazWord project
[/BazWord]
bazdevelopers = rw

# Members of the foodevelopers group can read and write
# to the FooCalc project
[/FooCalc]
foodevelopers = rw

# Members of foodevelopers can read the branches
directory but only user julia (the release manager)
# can write to this path
[/FooCalc/branches]
julia = rw
foodevelopers = r

# peterc is a contractor, so he's denied all access to the statistics
# module (which is full of trade secrets).
[/FooCalc/trunk/statistics]
peterc =
```

Some notes:

- The format is a series of one or more repository paths (minus the leading URL) followed by one or more group or user directives for each path.
- You don't have to include every single path. If an exact path match isn't found, the settings for the nearest parent directory are used.
- A user or group can be set to one of:
  - rw: read and write access.
  - r: read-only access.
  - blank: no access.
- Group names are indicated by a leading '@' character.
- Lines starting with a '#' are comments.
- Note that the [groups] section of the file described in the Subversion docs is ignored by Apache::CrowdAuthz as group memberships come from Crowd.
- Don't prefix the paths in the file with the repository name (e.g. '[calc:/foo]') (see note on SVNParentPath below).

Some notes:

- The format is a series of one or more repository paths (minus the leading URL) followed by one or more group or user directives for each path.
- You don't have to include every single path. If an exact path match isn't found, the settings for the nearest parent directory are used.
- A user or group can be set to one of:
  - rw: read and write access.
  - r: read-only access.
  - blank: no access.
- Group names are indicated by a leading '@' character.
- Lines starting with a '#' are comments.
- Note that the [groups] section of the file described in the Subversion docs is ignored by Apache::CrowdAuthz as group memberships come from Crowd.
- Don't prefix the paths in the file with the repository name (e.g. '[calc:/foo]') (see note on SVNParentPath below).

For a detailed description of this file format, see the Subversion documentation.

Related Topics

- 3.1 Using the Application Browser
- 3.2 Adding an Application
  - 3.2.01 Integrating Crowd with Apache
  - 3.2.02 Integrating Crowd with Subversion
  - 3.2.03 Integrating Crowd with Atlassian Confluence
    - 3.2.3.1 Configuring Confluence for NTLM SSO
  - 3.2.04 Integrating Crowd with Atlassian CrowdID
  - 3.2.05 Integrating Crowd with Atlassian FishEye
  - 3.2.06 Integrating Crowd with Atlassian JIRA
- 3.2.6.1 Configuring JIRA for NTLM SSO
  - 3.2.07 Integrating Crowd with Jive Forums
  - 3.2.7.1 Jive SSO
  - 3.2.08 Integrating Crowd with Atlassian Bamboo
  - 3.2.09 Integrating Crowd with Acegi Security
    - Integrating AppFuse - a Crowd-Acegi Integration Tutorial
  - 3.2.10 Integrating Crowd with a Custom Application
- 3.3 Mapping a Directory to an Application
  - 3.3.1 Specifying the Directory Order for an Application
  - 3.3.2 Specifying an Application's Directory Permissions
    - Example of Directory Permissions
- 3.4 Specifying which Groups can access an Application
- 3.5 Specifying an Application's Address or Hostname
- 3.6 Testing a User's Login to an Application
- 3.7 Managing an Application's Session
- 3.8 Deleting or Deactivating an Application

Crowd Documentation
3.2.03 Integrating Crowd with Atlassian Confluence

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 (SSO) for Confluence versions 2.5.6 and later with Crowd 1.1.2 and later.

If you are using NTLM for Windows authentication, you may want to read about configuring Crowd's Confluence NTLM plugin for single sign on.

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 2.5.6 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. 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:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>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.</td>
</tr>
<tr>
<td>Description</td>
<td>A short description of the application. Note: A web URL is often helpful.</td>
</tr>
<tr>
<td>Active</td>
<td>Only deselect this if you wish to prevent all users (from all directories) from accessing this application.</td>
</tr>
<tr>
<td>Password</td>
<td>The password which the application will use when it authenticates against the Crowd framework as a client.</td>
</tr>
<tr>
<td>Default Directory</td>
<td>A directory that contains relevant users. Note: Additional directories can be added later.</td>
</tr>
</tbody>
</table>

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 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:

<table>
<thead>
<tr>
<th>Copy From</th>
<th>Copy To</th>
</tr>
</thead>
<tbody>
<tr>
<td>CROWD/client/crowd-core-x.x.x.jar</td>
<td>CONFLUENCE/confluence/WEB-INF/lib</td>
</tr>
<tr>
<td>CROWD/client/crowd-atlassian-user-x.x.x.jar</td>
<td>CONFLUENCE/confluence/WEB-INF/lib</td>
</tr>
<tr>
<td>CROWD/client/conf/crowd.properties</td>
<td>CONFLUENCE/confluence/WEB-INF/classes</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>application.name</td>
<td>confluence</td>
</tr>
<tr>
<td>application.password</td>
<td>set a password</td>
</tr>
</tbody>
</table>
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). Confluence does not use any of the other attributes of the crowd.properties file.

### 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. Complete one of the following sub-steps, depending on your version of Confluence:
   - For Confluence versions earlier than 2.5.6, please upgrade to the latest stable version of Confluence.
2. Edit the CONFLUENCE/confluence/WEB-INF/classes/atlassian-user.xml file so that the contents of the file is:

   ```xml
   <atlassian-user>
   <repositories>
   <crowd key="crowd" name="Crowd Repository"/>
   </repositories>
   </atlassian-user>
   ```

3. At this stage, Confluence is set up for centralised authentication. If you wish, you can now enable single sign-on (SSO) to Confluence.

   ❗ Skip this step if you are using the Confluence NTLM plugin to enable SSO. Instead, follow the instructions on [configuring Confluence for NTLM SSO](#).

   Edit CONFLUENCE/confluence/webapp/WEB-INF/classes/seraph-config.xml. Comment out the authenticator node:

   ```xml
   <!--authenticator class= "com.atlassian.confluence.user.ConfluenceAuthenticator" -->
   ```

   and add a new one:

   ```xml
   <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](#).

- If you are using Confluence 2.5.5 or earlier, this step is required i.e. you must turn on external user management in Confluence.
- If you have imported Confluence users into Crowd, you may want to delay turning on 'External User Management' for a week or two, to give users time to reset their passwords. (Because users' passwords are encrypted in Confluence's database, they will not be copied across to Crowd.)
2.4 (Optional) Tune the Cache

When utilising the atlassian-user and Crowd framework together with Confluence, 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 via the Crowd Options menu. (Note that this caching in the Crowd application is enabled by default.)

Confluence will obtain all necessary information for the period specified by the cache configuration - see 5.2 Configuring Caching for an Application. If a change or addition occurs in Crowd to users, groups and roles, these changes will not be visible in Confluence until the cache expires for that specific item (i.e. for the particular user, group or role).

The default value for the application cache is 5 minutes (300 seconds). To increase the performance of your application, consider changing the cache value to one or two hours (3600 or 7200 seconds).

See Crowd in Action

- You should now be able to log in using principals belonging to the confluence-users group. Try adding a principal to the group using Crowd — you should be able to log in 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
  - 3.2.01 Integrating Crowd with Apache
  - 3.2.02 Integrating Crowd with Subversion
  - 3.2.03 Integrating Crowd with Atlassian Confluence
    - 3.2.3.1 Configuring Confluence for NTLM SSO
  - 3.2.04 Integrating Crowd with Atlassian CrowdID
  - 3.2.05 Integrating Crowd with Atlassian FishEye
  - 3.2.06 Integrating Crowd with Atlassian JIRA
    - 3.2.6.1 Configuring JIRA for NTLM SSO
  - 3.2.07 Integrating Crowd with Jive Forums
  - 3.2.7.1 Jive SSO
  - 3.2.08 Integrating Crowd with Atlassian Bamboo
  - 3.2.09 Integrating Crowd with Acegi Security
    - Integrating AppFuse - a Crowd-Acegi Integration Tutorial
  - 3.2.10 Integrating Crowd with a Custom Application
- 3.3 Mapping a Directory to an Application
  - 3.3.1 Specifying the Directory Order for an Application
  - 3.3.2 Specifying an Application’s Directory Permissions
    - Example of Directory Permissions
- 3.4 Specifying which Groups can access an Application
- 3.5 Specifying an Application’s Address or Hostname
- 3.6 Testing a User’s Login to an Application
- 3.7 Managing an Application’s Session
- 3.8 Deleting or Deactivating an Application
3.2.3.1 Configuring Confluence for NTLM SSO

Out of the box, Confluence does not support Single Sign On (SSO) functionality. This page describes how to set up Confluence with NTLM SSO functionality using the Confluence NTLM plugin, Crowd, and Active Directory (AD) as your LDAP user repository.

Summary

The Confluence NTLM plugin enables the following authentication scenario:

- A user in a Windows domain logs into the Windows network, using their Active Directory username/password.
- Then, when they open Confluence in an Internet Explorer browser, they are seamlessly logged into Confluence.

The Crowd component then allows you to manage all users and groups in Active Directory. Crowd automatically ensures that users and groups are synchronised between AD and Confluence. For example, if a user/group is added/deleted from AD it will be automatically added/deleted from Confluence.

Components

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Confluence NTLM plugin</td>
<td>NTLM is the protocol used by Windows for authentication. The Confluence NTLM plugin takes care of the Windows domain / Active Directory login to Confluence. You must be running a Windows Domain Controller with accounts set up in AD in order to use this plugin. If NTLM authentication is not available, the plugin allows standard form-based login to Confluence. Note: This plugin is not officially supported by Atlassian.</td>
</tr>
<tr>
<td>Crowd</td>
<td>Crowd takes care of the synchronisation of users/groups between Active Directory and Confluence. You will need to create an SSL connection between Crowd and the AD server if you would like to create users through Crowd. AD will not allow Crowd to add users or change their passwords unless the communication occurs over a secure connection.</td>
</tr>
<tr>
<td>Active Directory (AD) on Windows 2003 Server</td>
<td>Active Directory (AD) on Windows 2003 Server — you must already have an AD instance set up and running with a domain controller.</td>
</tr>
<tr>
<td>Confluence</td>
<td>The machine running Confluence must be part of the Windows domain or installed on the same box as the domain controller.</td>
</tr>
</tbody>
</table>

Steps

1. Back up your Confluence installation files and data:
   - Confluence Home directory. (See Confluence's Important Directories and Files for how to locate this).
   - Confluence installation directory (if you are using Confluence Standalone) or your Confluence webapp (if you are using Confluence EAR–WAR).
   - Your database (if you are not using the embedded database).
2. Download the Confluence NTLM plugin.
3. Install the plugin, following the instructions on the plugin documentation page.
4. In the `ldaputil.properties` file, insert the appropriate LDAP and Domain Controller information along with other parameters.
5. Install and configure Crowd.
6. Create a directory in Crowd for the AD LDAP server.
7. Create the Confluence application in Crowd and configure Crowd and Confluence to talk to each other, as described in Integrating Crowd with Atlassian Confluence.

When following the above instructions, do not change the `seraph-config.xml` file to enable Crowd's SSO functionality. (I.e. don’t change the authenticator node to read `<authenticator class="com.atlassian.crowd.integration.seraph.ConfluenceAuthenticator"/>`.) Instead of Crowd’s SSO authentication, we’ll be using the Confluence NTLM plugin.

8. In AD, create the groups confluence-users and confluence-administrators. They should then appear in Crowd.
9. In AD, create an admin user and make them a member of the above groups in AD.
10. Create any additional groups that you would like in AD.
11. Log in to the Windows domain using your desktop login and then open Confluence in an Internet Explorer browser. You should be logged in automatically.

Additional Crowd Performance Tips

- Change the default cache setting timeout in the file `<CONFLUENCE>\WEB-INF\classes\crowd-ehcache.xml`. For performance reasons, increase the object caching to 7,200 seconds (2 hours): `timeToIdleSeconds="7200" timeToLiveSeconds="7200"`. This reduces the frequency of the requests from Crowd to the LDAP server when changes to LDAP objects (such as a group name or user attribute) are made, thus reducing the performance overhead.

- Turn on the ‘Use Paged Results’ option in the directory connector tab for the directory you’ve set up in Crowd.
3.2.04 Integrating Crowd with Atlassian CrowdID

Atlassian CrowdID is a free add-on to Crowd. It gives administrators a secure way to provide OpenID accounts for their users.

When installing Crowd 1.1+ the Crowd Setup Wizard allows you to install CrowdID with Crowd. If you chose to install CrowdID during the installation process of Crowd, there is no need for further configuration. The CrowdID server will be up and running at http://localhost:8095/openidserver

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. This guide assumes that CrowdID was NOT installed with the installation of Crowd. If CrowdID was installed using the Crowd Setup Wizard, there is no need for further configuration.

Step 1. Configuring Crowd to talk to CrowdID

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

The CrowdID application will need to locate users from a directory configured in Crowd. You will need to set up a directory in Crowd for CrowdID. For information on how to do this, see 2.2 Adding a Directory. We will assume that the directory is called CrowdID 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 CrowdID Directory to house CrowdID users.

CrowdID also requires an administrator group to exist in the directory. You need to ensure that a crowd-administrators groups exist in the CrowdID Directory. Any user in this group will have CrowdID administrator access.

The Crowd documentation has more information on creating groups, creating principals and assigning principals to groups.

1.2 Define the CrowdID Application in Crowd

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

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

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name:</td>
<td>*crowd-openid-server</td>
</tr>
<tr>
<td>Description:</td>
<td>CrowdID</td>
</tr>
<tr>
<td>Active:</td>
<td></td>
</tr>
<tr>
<td>Password:</td>
<td>*</td>
</tr>
<tr>
<td>Confirm Password:</td>
<td>*</td>
</tr>
<tr>
<td>Default Directory:</td>
<td>*CrowdID Directory</td>
</tr>
</tbody>
</table>

Document generated by Confluence on Nov 27, 2007 21:24
<table>
<thead>
<tr>
<th>Name</th>
<th>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.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>A short description of the application. Note: A web URL is often helpful.</td>
</tr>
<tr>
<td>Active</td>
<td>Only deselect this if you wish to prevent all users (from all directories) from accessing this application.</td>
</tr>
<tr>
<td>Password</td>
<td>The password which the application will use when it authenticates against the Crowd framework as a client.</td>
</tr>
<tr>
<td>Default Directory</td>
<td>A directory that contains relevant users. Note: Additional directories can be added later.</td>
</tr>
</tbody>
</table>

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

1.3 Specify which users can log in to CrowdID

Now that Crowd is aware of the CrowdID 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 entire CrowdID Directory to authenticate:

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

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

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

- If CrowdID is on a different host to Crowd
  If you are running the CrowdID 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 CrowdID is remote to Crowd, add the IP address of your CrowdID server and ensure the "Status" field is set to "true". Remove the entry for localhost.

- If CrowdID 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 CrowdID (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 CrowdID to talk to Crowd

Edit CROWD/crowd-openidservlet-webapp/WEB-INF/classes/crowd.properties. Change the following properties:

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>application.name</td>
<td>crowd-openid-server</td>
</tr>
<tr>
<td>application.password</td>
<td>set a password</td>
</tr>
<tr>
<td>application.login.url</td>
<td><a href="http://localhost:8095/openidserver">http://localhost:8095/openidserver</a></td>
</tr>
<tr>
<td>crowd.server.url</td>
<td><a href="http://localhost:8095/crowd/services/">http://localhost:8095/crowd/services/</a></td>
</tr>
<tr>
<td>session.validationinterval</td>
<td>Set to 0, if you want authentication checks to occur on each request. Otherwise set to the number of minutes between request to validate if the user is logged in or out of the Crowd SSO server. Setting this value to 1 or higher will increase the performance of Crowd's integration.</td>
</tr>
</tbody>
</table>

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). The application.login.url should point to the correct host and port of the Crowd application.

See CrowdID in Action

- Go to http://localhost:8095/openidserver and login with any principal in the CrowdID Directory.

Related Topics

- 3.1 Using the Application Browser
- 3.2 Adding an Application
  - 3.2.01 Integrating Crowd with Apache
  - 3.2.02 Integrating Crowd with Subversion
  - 3.2.03 Integrating Crowd with Atlassian Confluence
    - 3.2.3.1 Configuring Confluence for NTLM SSO
  - 3.2.04 Integrating Crowd with Atlassian CrowdID
  - 3.2.05 Integrating Crowd with Atlassian FishEye
  - 3.2.06 Integrating Crowd with Atlassian JIRA
    - 3.2.6.1 Configuring JIRA for NTLM SSO
  - 3.2.07 Integrating Crowd with Jive Forums
    - 3.2.7.1 Jive SSO
  - 3.2.08 Integrating Crowd with Atlassian Bamboo
  - 3.2.09 Integrating Crowd with Acegi Security
    - Integrating AppFuse - a Crowd-Acegi Integration Tutorial
  - 3.2.10 Integrating Crowd with a Custom Application
- 3.3 Mapping a Directory to an Application
  - 3.3.1 Specifying the Directory Order for an Application
  - 3.3.2 Specifying an Application's Directory Permissions
    - Example of Directory Permissions
- 3.4 Specifying which Groups can access an Application
- 3.5 Specifying an Application's Address or Hostname
- 3.6 Testing a User's Login to an Application
- 3.7 Managing an Application's Session
- 3.8 Deleting or Deactivating an Application

Crowd Documentation
3.2.05 Integrating Crowd with Atlassian FishEye

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.

2.1 Install the Crowd Client Libraries into FishEye

Copy the Crowd integration libraries and configuration files as described in the 3.2.10 Integrating Crowd with a Custom Application documentation. This involves copying all client library JARs to the library folder of FishEye:

The version numbers have been omitted. Select the JAR which matches the name. This listing has been verified with FishEye 1.3.1.

<table>
<thead>
<tr>
<th>Files to Copy</th>
<th>Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>atlassian-crowd/client/crowd-core.jar</td>
<td>fisheye/lib</td>
</tr>
<tr>
<td>atlassian-crowd/client/crowd-atlassian-user.jar</td>
<td>fisheye/lib</td>
</tr>
<tr>
<td>atlassian-crowd/crowd-webapp/WEB-INF/lib/commons-codec.jar</td>
<td>fisheye/lib</td>
</tr>
<tr>
<td>atlassian-crowd/crowd-webapp/WEB-INF/lib/commons-httpclient.jar</td>
<td>fisheye/lib</td>
</tr>
<tr>
<td>atlassian-crowd/crowd-webapp/WEB-INF/lib/commons-lang.jar</td>
<td>fisheye/lib</td>
</tr>
<tr>
<td>atlassian-crowd/crowd-webapp/WEB-INF/lib/jdom-1.0.jar</td>
<td>fisheye/lib</td>
</tr>
<tr>
<td>atlassian-crowd/crowd-webapp/WEB-INF/lib/stax-api-1.0.1.jar</td>
<td>fisheye/lib</td>
</tr>
<tr>
<td>atlassian-crowd/crowd-webapp/WEB-INF/lib/wsd14j.jar</td>
<td>fisheye/lib</td>
</tr>
<tr>
<td>atlassian-crowd/crowd-webapp/WEB-INF/lib/wstx-asl.jar</td>
<td>fisheye/lib</td>
</tr>
<tr>
<td>atlassian-crowd/crowd-webapp/WEB-INF/lib/xfire-all.jar</td>
<td>fisheye/lib</td>
</tr>
</tbody>
</table>
2.2 Configure FishEye to use Crowd's Authenticator

Log in as an admin to FishEye and navigate to Users/Security. Select Setup Custom authentication and enter the following classname for the authenticator:

com.atlassian.crowd.integration.fisheye.FisheyeAuthenticator

Leave the cache and auto-add settings to their default values. This will mean authentication calls to Crowd will be cached (improves performance) and that users will be automatically enrolled into FishEye after their initial login to FishEye via Crowd.

FishEye requires you to pass in the configuration attributes for Crowd. Specify your configuration data, matching the values set in Step 1, through the properties editor:

<table>
<thead>
<tr>
<th>Configuration Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>application.name</td>
<td>fisheye</td>
</tr>
<tr>
<td>application.password</td>
<td>password</td>
</tr>
<tr>
<td>application.login.url</td>
<td><a href="http://localhost:8080/">http://localhost:8080/</a></td>
</tr>
<tr>
<td>crowd.server.url</td>
<td><a href="http://localhost:8095/crowd/services/">http://localhost:8095/crowd/services/</a></td>
</tr>
<tr>
<td>session.isauthenticated</td>
<td>session.isauthenticated</td>
</tr>
<tr>
<td>session.tokenkey</td>
<td>session.tokenkey</td>
</tr>
<tr>
<td>session.validationinterval</td>
<td>0</td>
</tr>
<tr>
<td>session.lastvalidation</td>
<td>session.lastvalidation</td>
</tr>
</tbody>
</table>

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
  - 3.2.01 Integrating Crowd with Apache
  - 3.2.02 Integrating Crowd with Subversion
  - 3.2.03 Integrating Crowd with Atlassian Confluence
    - 3.2.3.1 Configuring Confluence for NTLM SSO
  - 3.2.04 Integrating Crowd with Atlassian CrowdID
  - 3.2.05 Integrating Crowd with Atlassian FishEye
  - 3.2.06 Integrating Crowd with Atlassian JIRA
3.2.6.1 Configuring JIRA for NTLM SSO
- 3.2.07 Integrating Crowd with Jive Forums
  - 3.2.07.1 Jive SSO
  - 3.2.08 Integrating Crowd with Atlassian Bamboo
  - 3.2.09 Integrating Crowd with Acegi Security
    - Integrating AppFuse - a Crowd-Acegi Integration Tutorial
  - 3.2.10 Integrating Crowd with a Custom Application

3.3 Mapping a Directory to an Application
- 3.3.1 Specifying the Directory Order for an Application
- 3.3.2 Specifying an Application's Directory Permissions
  - Example of Directory Permissions

3.4 Specifying which Groups can access an Application
- 3.5 Specifying an Application's Address or Hostname
- 3.6 Testing a User's Login to an Application
- 3.7 Managing an Application's Session
- 3.8 Deleting or Deactivating an Application

Crowd Documentation
3.2.06 Integrating Crowd with Atlassian JIRA

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 3.7.4 and later.

If you are using NTLM for Windows authentication, you may want to read about configuring Crowd’s JIRA NTLM plugin for single sign-on.

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:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>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.</td>
</tr>
<tr>
<td>Description</td>
<td>A short description of the application. Note: A web URL is often helpful.</td>
</tr>
<tr>
<td>Active</td>
<td>Only deselect this if you wish to prevent all users (from all directories) from accessing this application.</td>
</tr>
<tr>
<td>Password</td>
<td>The password which the application will use when it authenticates against the Crowd framework as a client.</td>
</tr>
<tr>
<td>Default Directory</td>
<td>A directory that contains relevant users. Note: Additional directories can be added later.</td>
</tr>
</tbody>
</table>

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.
With this example, only principals who are members of the jira-users, jira-developers and jira-administrators groups will be able to authenticate against JIRA.

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:

<table>
<thead>
<tr>
<th>Copy From</th>
<th>Copy To</th>
</tr>
</thead>
<tbody>
<tr>
<td>CROWD/client/*.jar</td>
<td>JIRA/atlassian-jira/WEB-INF/lib</td>
</tr>
<tr>
<td>CROWD/client/conf/crowd.properties</td>
<td>JIRA/atlassian-jira/WEB-INF/classes</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>application.name</td>
<td>jira</td>
</tr>
</tbody>
</table>
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:

   ```xml
   <!-- 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>
   -->
   </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>:

   ```xml
   <propertyset name="crowd" class="com.atlassian.crowd.integration.osuser.CrowdPropertySet"/>
   ```
3. At this stage, JIRA is set up for centralised authentication. If you wish, you can now enable single sign-on (SSO) to JIRA.

   ! Skip this step if you are using the JIRA NTLM plugin to enable SSO. Instead, follow the instructions on configuring JIRA for NTLM SSO.

   Edit JIRA/atlassian-jira/WEB-INF/classes/seraph-config.xml. Change the authenticator node to read:

   ```xml
   <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, you can configure JIRA to allow external user management. Go to the JIRA Administration Console. In the General Configuration section, turn 'External user management' and 'External password management' on or off. (See the JIRA Administrator's Guide for details).

JIRA with external user management ON:

This is recommended, because it allows you to use Crowd's powerful cross-directory user administration features.

If you turn external user management on, the following functions can no longer be performed from within the JIRA administration interface: adding users, adding groups, editing users, editing groups.

⚠️ If you are using Crowd 1.1.1 or earlier, you must turn external user management on in JIRA.

JIRA with external user management OFF:

This means that you can allow signup via JIRA, and you can manage your users within JIRA. Changes will flow through to Crowd.

JIRA has an automatic group membership feature. This means that any new user added through JIRA will automatically be a member of all groups which have the JIRA Users permission. In this way, you can ensure that a new user is automatically added to several groups when they sign up with JIRA.

⚠️ Any group or user changes will cascade to all directories assigned to the JIRA application in Crowd. For example, if user 'jbloggs' registers in JIRA, 'jbloggs' will be added to every Crowd directory linked with the JIRA application.

2.4 (Optional) Tune the Cache

When utilising the atlassian-user and Crowd framework together with JIRA, 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 via the Crowd Options menu. (Note that this caching in the Crowd application is enabled by default.)

JIRA will obtain all necessary information for the period specified by the cache configuration - see 5.2 Configuring Caching for an Application. If a change or addition occurs in Crowd to users, groups and roles, these changes will not be visible in JIRA until the cache expires for that specific item (i.e. for the particular user, group or role).

⚠️ The default value for the application cache is 5 minutes (300 seconds). To increase the performance of your application, consider changing the cache value to one or two hours (3600 or 7200 seconds).

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

A problem occurs in JIRA if a user is removed after that user has participated in an issue i.e. if JIRA refers to the user. If the user is internally managed by JIRA, JIRA will prevent the removal of the user but if the user is managed by an external system such as 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
  ° 3.2.01 Integrating Crowd with Apache
  ° 3.2.02 Integrating Crowd with Subversion
  ° 3.2.03 Integrating Crowd with Atlassian Confluence
    - 3.2.3.1 Configuring Confluence for NTLM SSO
  ° 3.2.04 Integrating Crowd with Atlassian CrowdID
  ° 3.2.05 Integrating Crowd with Atlassian FishEye
  ° 3.2.06 Integrating Crowd with Atlassian JIRA
    - 3.2.6.1 Configuring JIRA for NTLM SSO
  ° 3.2.07 Integrating Crowd with Jive Forums
    - 3.2.7.1 Jive SSO
  ° 3.2.08 Integrating Crowd with Atlassian Bamboo
  ° 3.2.09 Integrating Crowd with Acegi Security
    - Integrating AppFuse - a Crowd-Acegi Integration Tutorial
  ° 3.2.10 Integrating Crowd with a Custom Application
• 3.3 Mapping a Directory to an Application
  ° 3.3.1 Specifying the Directory Order for an Application
  ° 3.3.2 Specifying an Application's Directory Permissions
    - Example of Directory Permissions
• 3.4 Specifying which Groups can access an Application
• 3.5 Specifying an Application's Address or Hostname
• 3.6 Testing a User's Login to an Application
• 3.7 Managing an Application's Session
• 3.8 Deleting or Deactivating an Application

Crowd Documentation
3.2.6.1 Configuring JIRA for NTLM SSO

Out of the box, JIRA does not support Single Sign On (SSO) functionality. This page describes how to set up JIRA with NTLM SSO functionality using the JIRA NTLM plugin, Crowd, and Active Directory (AD) as your LDAP user repository.

Summary

The JIRA NTLM plugin enables the following authentication scenario:

- A user in a Windows domain logs into the Windows network, using their Active Directory username/password.
- Then, when they open JIRA in an Internet Explorer browser, they are seamlessly logged into JIRA.

The Crowd component then allows you to manage all users and groups in Active Directory. Crowd automatically ensures that users and groups are synchronised between AD and JIRA. For example, if a user/group is added/deleted from AD it will be automatically added/deleted from JIRA.

Components

| JIRA NTLM plugin | NTLM is the protocol used by Windows for authentication. The JIRA NTLM plugin takes care of the Windows domain / Active Directory login to JIRA. You must be running a Windows Domain Controller with accounts set up in AD in order to use this plugin. If NTLM authentication is not available, the plugin allows standard form-based login to JIRA. Note: This plugin is not officially supported by Atlassian. |
| Crowd | Crowd takes care of the synchronisation of users/groups between Active Directory and JIRA. You will need to create an SSL connection between Crowd and the AD server if you would like to create users through Crowd. AD will not allow Crowd to add users or change their passwords unless the communication occurs over a secure connection. |
| Active Directory (AD) on Windows 2003 Server | Active Directory (AD) on Windows 2003 Server — you must already have an AD instance set up and running with a domain controller. |
| JIRA | The machine running JIRA must be part of the Windows domain or installed on the same box as the domain controller. |

Steps

1. Back up your entire JIRA installation directory and run an XML backup of your data.
2. Download the JIRA NTLM plugin.
3. Read the README file included in the plugin zip file, and then follow the instructions in the INSTALL file to install the plugin.
4. In the ntlm_ldap.properties file, insert the appropriate LDAP and Domain Controller information along with other parameters.
5. Install and configure Crowd.
6. **Create a directory** in Crowd for the AD LDAP server.

7. Create the JIRA application in Crowd and configure Crowd and JIRA to talk to each other, as described in [Integrating Crowd with Atlassian JIRA](#).

   When following the above instructions, **do not change the seraph-config.xml file to enable Crowd's SSO functionality.** (I.e. don't change the authenticator node to read `<authenticator class="com.atlassian.crowd.integration.seraph.JIRAAuthenticator"/>`.) Instead of Crowd's SSO authentication, we'll be using the JIRA NTLM plugin.

8. In AD, create the groups jira-users, jira-developers, and jira-administrators. They should then appear in Crowd.

9. In AD, create an admin user and make them a member of the above three groups in AD.

10. Create any additional groups that you would like in AD.

11. Log into the Windows domain using your desktop login and then open JIRA in an Internet Explorer browser. You should be logged in automatically.

### Additional Crowd Performance Tips

- Change the default cache setting timeout in the file `<JIRA>/WEB-INF/classes/crowd-ehcache.xml`. For performance reasons, increase the object caching to 7,200 seconds (2 hours):
  ```xml
timeToIdleSeconds="7200" timeToLiveSeconds="7200".
```
  This reduces the frequency of the requests from Crowd to the LDAP server when changes to LDAP objects (such as a group name or user attribute) are made, thus reducing the performance overhead.

- Turn on the 'Use Paged Results' option in the [directory connector tab](#) for the directory you've set up in Crowd.
3.2.07 Integrating Crowd with Jive Forums

This page last changed on Nov 26, 2007 by smaddox.

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.

Crowd provides centralised authentication and single sign-on (SSO) for Jive Forums version 5.5.x.

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 Jive Forums 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:
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>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.</td>
</tr>
<tr>
<td>Description</td>
<td>A short description of the application. Note: A web URL is often helpful.</td>
</tr>
<tr>
<td>Active</td>
<td>Only deselect this if you wish to prevent all users (from all directories) from accessing this application.</td>
</tr>
<tr>
<td>Password</td>
<td>The password which the application will use when it authenticates against the Crowd framework as a client.</td>
</tr>
<tr>
<td>Default Directory</td>
<td>A directory that contains relevant users. Note: Additional directories can be added later.</td>
</tr>
</tbody>
</table>

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 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:

<table>
<thead>
<tr>
<th>Copy From</th>
<th>Copy To</th>
</tr>
</thead>
<tbody>
<tr>
<td>CROWD/client/crowd-core-*.jar</td>
<td>JIVEFORUMS/WEB-INF/lib</td>
</tr>
<tr>
<td>CROWD/client/crowd-integration-jive-*.jar</td>
<td>JIVEFORUMS/WEB-INF/lib</td>
</tr>
<tr>
<td>CROWD/client/lib/log4j-1.2.8.jar</td>
<td>JIVEFORUMS/WEB-INF/lib</td>
</tr>
<tr>
<td>CROWD/client/lib/ehcache-1.2.3.jar</td>
<td>JIVEFORUMS/WEB-INF/lib</td>
</tr>
<tr>
<td>CROWD/client/conf/crowd.properties</td>
<td>JIVEFORUMS/WEB-INF/classes</td>
</tr>
<tr>
<td>CROWD/client/conf/crowd-ehcache.xml</td>
<td>JIVEFORUMS/WEB-INF/classes</td>
</tr>
</tbody>
</table>

2. 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.

3. Edit JIVEFORUMS/WEB-INF/classes/crowd.properties. Change the following properties:

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>application.name</td>
<td>jiveforums</td>
</tr>
<tr>
<td>application.password</td>
<td>set a password</td>
</tr>
</tbody>
</table>

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:

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

7. 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:**
   
   If you would like Crowd to manage your user groups, add the following group manager:
   
   com.atlassian.crowd.integration.jive.CrowdGroupManager

   You can safely leave this field empty if you do not want Crowd to manage your groups.

   **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.

8. In the 'Email Settings' screen, re-enter your email configuration details and click 'Continue'.
9. 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) in Crowd. Without this user, you will not be able to access the administration console of Jive Forums.

10. 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
  - 3.2.01 Integrating Crowd with Apache
  - 3.2.02 Integrating Crowd with Subversion
  - 3.2.03 Integrating Crowd with Atlassian Confluence
    - 3.2.3.1 Configuring Confluence for NTLM SSO
  - 3.2.04 Integrating Crowd with Atlassian CrowdID
  - 3.2.05 Integrating Crowd with Atlassian FishEye
  - 3.2.06 Integrating Crowd with Atlassian JIRA
    - 3.2.6.1 Configuring JIRA for NTLM SSO
  - 3.2.07 Integrating Crowd with Jive Forums
  - 3.2.08 Integrating Crowd with Atlassian Bamboo
  - 3.2.09 Integrating Crowd with Acegi Security
    - Integrating AppFuse - a Crowd-Acegi Integration Tutorial
  - 3.2.10 Integrating Crowd with a Custom Application
- 3.3 Mapping a Directory to an Application
  - 3.3.1 Specifying the Directory Order for an Application
  - 3.3.2 Specifying an Application's Directory Permissions
    - Example of Directory Permissions
- 3.4 Specifying which Groups can access an Application
- 3.5 Specifying an Application's Address or Hostname
- 3.6 Testing a User's Login to an Application
- 3.7 Managing an Application's Session
- 3.8 Deleting or Deactivating an Application

Crowd Documentation
3.2.7.1 Jive SSO

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 user 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 log out from Jive's admin console -> admin is still logged in (support issue filed with Jive Forums).
9. authenticated Jive admin attempts to log out 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 log in to 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, you should 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. So they will make it hard for a banned user to switch to a non-banned user. 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 log out 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 log in, the user will have to wait until the client cache is cleared before he/she can log in 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**
  - 3.2.01 Integrating Crowd with Apache
  - 3.2.02 Integrating Crowd with Subversion
  - 3.2.03 Integrating Crowd with Atlassian Confluence
    - 3.2.3.1 Configuring Confluence for NTLM SSO
  - 3.2.04 Integrating Crowd with Atlassian CrowdID
  - 3.2.05 Integrating Crowd with Atlassian FishEye
  - 3.2.06 Integrating Crowd with Atlassian JIRA
    - 3.2.6.1 Configuring JIRA for NTLM SSO
  - 3.2.07 Integrating Crowd with Jive Forums
    - 3.2.7.1 Jive SSO
  - 3.2.08 Integrating Crowd with Atlassian Bamboo
  - 3.2.09 Integrating Crowd with Acegi Security
    - Integrating AppFuse - a Crowd-Acegi Integration Tutorial
  - 3.2.10 Integrating Crowd with a Custom Application
- **3.3 Mapping a Directory to an Application**
  - 3.3.1 Specifying the Directory Order for an Application
  - 3.3.2 Specifying an Application's Directory Permissions
    - Example of Directory Permissions
- **3.4 Specifying which Groups can access an Application**
- **3.5 Specifying an Application's Address or Hostname**
- **3.6 Testing a User's Login to an Application**
- **3.7 Managing an Application's Session**
- **3.8 Deleting or Deactivating an Application**

Crowd Documentation
3.2.08 Integrating Crowd with Atlassian Bamboo

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.

Currently Crowd supports centralised authentication and single sign-on for Bamboo versions 1.2.2 and later.

Due to incompatible atlassian-user libraries, Bamboo releases prior to 1.2.2 are not compatible with latest version of Crowd. We recommend that you upgrade to the latest version of Bamboo before attempting to integrate Crowd.

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 Bamboo (version 1.1.2 or later). Refer to the Bamboo Installation Guide for detailed information on how to do this. We will refer to the Bamboo root folder as BAMBOO. For the purposes of this document, we will assume that the Standalone (ie. the easier) installation method of Bamboo has been used. If you need to install Bamboo as an EAR/WAR, simply explode the EAR/WAR and make the necessary changes as described below, and repackage the EAR/WAR.
3. After Bamboo is set up, make sure Bamboo is not running when you begin the integration process described below.

Step 1. Configuring Crowd to talk to Bamboo

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

The Bamboo application will need to authenticate users against a directory configured in Crowd. You will need to set up a directory in Crowd for Bamboo. For more information on how to do this, see 2.2 Adding a Directory. We will assume that the directory is called Bamboo 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 Bamboo Directory to house Bamboo users.

Bamboo also an administrative group to exist in the directory in order to access the administration features. You will need to create two groups in the Bamboo Directory:

1. bamboo-admin
2. bamboo-user - optional

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

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

- If you have an existing Bamboo deployment and would like to import existing users (principals) and groups into Crowd, use the Bamboo Importer tool by navigating to Principals > Import Users > Bamboo. Select the Bamboo Directory as the directory into which Bamboo users will be imported. For details please see 2.4.5 Importing Users from Atlassian Bamboo. 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 Bamboo users, make sure you use Crowd to create at least one principal in the Bamboo Directory and assign them to both the bamboo-user and bamboo-admin group. The Crowd documentation has more information on creating groups, creating principals and assigning principals to groups.

1.2 Define the Bamboo Application in Crowd

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

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

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>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.</td>
</tr>
<tr>
<td>Description</td>
<td>A short description of the application. Note: A web URL is often helpful.</td>
</tr>
<tr>
<td>Active</td>
<td>Only deselect this if you wish to prevent all users (from all directories) from accessing this application.</td>
</tr>
<tr>
<td>Password</td>
<td>The password which the application will use when it authenticates against the Crowd framework as a client.</td>
</tr>
<tr>
<td>Default Directory</td>
<td>A directory that contains relevant users. Note: Additional directories can be added later.</td>
</tr>
</tbody>
</table>

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

1.3 Specify which users can log in to Bamboo

Now that Crowd is aware of the Bamboo application, Crowd needs to know which users can authenticate (log in) to Bamboo via Crowd. You can either allow entire directories to authenticate, or just particular groups within the directories. In our example, we will allow the bamboo-user and bamboo-admin groups within the Bamboo Directory to authenticate:
If you are not using a bamboo-user group as a security restriction, you will need to enable all to authenticate, otherwise only bamboo-admin group members will be able to login to Bamboo.

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

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

- If Bamboo is on a different host to Crowd
  If you are running Bamboo 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 Bamboo is remote to Crowd, add the IP address of your Bamboo server and ensure the "Status" field is set to "true". Remove the entry for localhost.

- If Bamboo 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 Bamboo (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 Bamboo to talk to Crowd

2.1 Install the Crowd Client Libraries into Bamboo

Bamboo 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 Bamboo application by editing the standalone application, which is an exploded WAR stored in BAMBOO/webapp.

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

<table>
<thead>
<tr>
<th>Copy From</th>
<th>Copy To</th>
</tr>
</thead>
<tbody>
<tr>
<td>CROWD/client/crowd-core-x.x.x.jar</td>
<td>BAMBOO/webapp/WEB-INF/lib</td>
</tr>
<tr>
<td>CROWD/client/crowd-atlassian-user-x.x.x.jar</td>
<td>BAMBOO/webapp/WEB-INF/lib</td>
</tr>
<tr>
<td>CROWD/client/conf/crowd.properties</td>
<td>BAMBOO/webapp/WEB-INF/classes</td>
</tr>
</tbody>
</table>

There is no need to copy across anything from CROWD/client/lib. All the required libraries from there already exist in Bamboo.

2. Edit BAMBOO/webapp/WEB-INF/classes/crowd.properties. Change the following properties:

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>application.name</td>
<td>bamboo</td>
</tr>
<tr>
<td>application.password</td>
<td>set a password</td>
</tr>
<tr>
<td>crowd.server.url</td>
<td><a href="http://localhost:8095/crowd/services/">http://localhost:8095/crowd/services/</a></td>
</tr>
<tr>
<td>session.validationinterval</td>
<td>Set to 0, if you want authentication checks to occur on each request. Otherwise set to the number of minutes between request to validate if the user is logged in or out of the Crowd SSO server. Setting this value to 1 or higher will increase the performance of Crowd's integration.</td>
</tr>
</tbody>
</table>

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). Bamboo does not use any of the other attributes of the crowd.properties file.

2.2 Configure Bamboo to use Crowd's Authenticator

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

1. Complete one of the following sub-steps, depending on your version of Bamboo:
   • For Bamboo versions earlier than 2.5.6, please upgrade to the latest stable version of Bamboo.

2. Edit the Bamboo/webapp/WEB-INF/classes/atlassian-user.xml file so that the contents of the file is:

   `<atlassian-user>
   <repositories>
   <crowd key= "crowd" name= "Crowd Repository" />
   </repositories>
   </atlassian-user>

3. At this stage, Bamboo is set up for centralised authentication. If you wish to enable single sign-on (SSO) to Bamboo, edit BAMBOO/webapp/WEB-INF/classes/seraph-config.xml. Comment out the authenticator node:

   `<!--<authenticator class= "com.atlassian.bamboo.user.authentication.BambooAuthenticator"/>-->

   and add a new one:

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

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

2.3 Disable 'External User Management' in Bamboo

For Bamboo to integrate successfully with Crowd, Bamboo’s 'External User Management' option needs to be unchecked:

**Security and Permission**

You can change the following security and permission related settings for Bamboo.

<table>
<thead>
<tr>
<th>Change Global Security and Permission Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Enable External User Management!</td>
</tr>
<tr>
<td>Enable this option if you are delegating your user management to another user management system (e.g. Crowd).</td>
</tr>
<tr>
<td>- Enable Signup?</td>
</tr>
<tr>
<td>This will allow users to sign up for an account to Bamboo.</td>
</tr>
<tr>
<td>- Enable contact details to be displayed?</td>
</tr>
<tr>
<td>This will allow Bamboo users contact details to be visible. Disabling this option will hide the email address, IM address, and the group the user is in.</td>
</tr>
</tbody>
</table>

2.4 (Optional) 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.

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

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

2. Bamboo will also require the latest version of Atlassian Seraph. Copy this JAR file into Bamboo's `/bamboo/webapp/WEB-INF/lib` directory and remove the old file.
2.5 (Optional) Tune the Cache

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 via the Crowd Options menu. (Note that this caching in the Crowd application is enabled by default.)

Bamboo will obtain all necessary information for the period specified by the cache configuration - see 5.2 Configuring Caching for an Application. If a change or addition occurs in Crowd to users, groups and roles, these changes will not be visible in Bamboo until the cache expires for that specific item (i.e. for the particular user, group or role).

The default value for the application cache is 5 minutes (300 seconds). To increase the performance of your application, consider changing the cache value to one or two hours (3600 or 7200 seconds).

See Crowd in Action

- You should now be able to log in using principals belonging to the bamboo-user group. Try adding a principal to the group using Crowd — you should be able to log in to Bamboo using this newly created principal. That's centralised authentication in action!
- If you have enabled SSO, you can try adding the Bamboo Directory and bamboo-admin 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 Bamboo administrators to log in to the Crowd Administration Console. Try logging in to Crowd as a Bamboo administrator, and then point your browser at Bamboo. You should be logged in as the same principal in Bamboo. That's single sign-on in action!

Related Topics

- 3.1 Using the Application Browser
- 3.2 Adding an Application
  - 3.2.01 Integrating Crowd with Apache
  - 3.2.02 Integrating Crowd with Subversion
  - 3.2.03 Integrating Crowd with Atlassian Confluence
    - 3.2.3.1 Configuring Confluence for NTLM SSO
  - 3.2.04 Integrating Crowd with Atlassian CrowdID
  - 3.2.05 Integrating Crowd with Atlassian FishEye
  - 3.2.06 Integrating Crowd with Atlassian JIRA
    - 3.2.6.1 Configuring JIRA for NTLM SSO
  - 3.2.07 Integrating Crowd with Jive Forums
    - 3.2.7.1 Jive SSO
  - 3.2.08 Integrating Crowd with Atlassian Bamboo
  - 3.2.09 Integrating Crowd with Acegi Security
    - Integrating AppFuse - a Crowd-Acegi Integration Tutorial
  - 3.2.10 Integrating Crowd with a Custom Application
- 3.3 Mapping a Directory to an Application
  - 3.3.1 Specifying the Directory Order for an Application
  - 3.3.2 Specifying an Application's Directory Permissions
    - Example of Directory Permissions
- 3.4 Specifying which Groups can access an Application
- 3.5 Specifying an Application's Address or Hostname
- 3.6 Testing a User's Login to an Application
- 3.7 Managing an Application's Session
- 3.8 Deleting or Deactivating an Application

Crowd Documentation
3.2.09 Integrating Crowd with Acegi Security

Crowd provides centralised authentication and single sign-on connectors for the web security framework Acegi. Acegi provides a modular and highly configurable approach to authentication and authorisation for J2EE applications.

The connectors are available with Crowd 1.2 and later and have been developed and tested with Acegi 1.0.5.

Please consult the Acegi quick start guide or reference guide for a thorough insight into the Acegi framework. You might also find useful information in our Crowd-Acegi integration tutorial.

⚠️ This guide assumes developer-level knowledge

This guide assumes you have Crowd 1.2 or later installed and that you want to integrate your Acegi-based web application with Crowd's security server. This guide is more for developers than administrators.

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. Have your Acegi-based custom application ready for tweaking. We will refer to your custom application as 'AcegiApp'.

Step 1. Configuring Crowd to Talk to your Acegi Application

Crowd needs to be aware that AcegiApp will be making authentication requests to Crowd. In brief, you will need to do the following:

1. Add the AcegiApp application to Crowd.
2. Add and configure the directories visible to AcegiApp.
3. Add and map the groups which are allowed to authenticate with AcegiApp.

Please see 3.2 Adding an Application for a detailed guide.

Step 2. Installing the Crowd Acegi Connector

2.1 Adding the Crowd Acegi Connector to your Acegi Application

You will need to add the Crowd Acegi connector library and its associated dependencies to your Acegi application. You can do this manually by copying over the JAR files to your Acegi application or, if your Acegi application is a Maven project, you can add the Crowd Acegi connector as a project dependency.

2.1.1 Manually Adding the Crowd Acegi Connector Libraries

⚠️ Follow either 2.1.1 or 2.1.2 (not both).

Copy the Crowd integration libraries and configuration files. This is described in the Client Configuration documentation. You will need to at least copy across the following files to your Acegi application:

<table>
<thead>
<tr>
<th>Copy From</th>
<th>Copy To</th>
</tr>
</thead>
<tbody>
<tr>
<td>CROWD/client/crowd-core-*.jar</td>
<td>AcegiApp/WEB-INF/lib</td>
</tr>
<tr>
<td>CROWD/client/crowd-integration-acegi-*.jar</td>
<td>AcegiApp/WEB-INF/lib</td>
</tr>
</tbody>
</table>
2.1.2 Adding the Crowd Acegi Connector as a Maven Dependency

Follow either 2.1.1 or 2.1.2 (not both).

If your Acegi application is a Maven project, we recommend you use Maven to add the Crowd Acegi connector as a Maven dependency to your application's pom.xml:

```xml
<dependency>
    <groupId>com.atlassian.crowd</groupId>
    <artifactId>crowd-integration-acegi</artifactId>
    <version>1.2</version>
</dependency>
```

See more information on Maven 2 integration.

2.2 Configuring the Crowd Acegi Connector Properties

The Crowd Acegi connector needs to be configured with the details of the Crowd server.

Once CWD-622 is resolved, this configuration will be Spring injected.

1. Copy the default crowd.properties file to the classpath of your Acegi application:

<table>
<thead>
<tr>
<th>Copy From</th>
<th>Copy To</th>
</tr>
</thead>
<tbody>
<tr>
<td>CROWD/client/conf/crowd.properties</td>
<td>AcegiApp/WEB-INF/classes</td>
</tr>
</tbody>
</table>

2. Edit the crowd.properties and populate the following fields appropriately:

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>application.name</td>
<td>Same as application name defined when adding the application to Crowd in Step 1.</td>
</tr>
<tr>
<td>application.password</td>
<td>Same as application password defined when adding the application to Crowd in Step 1.</td>
</tr>
<tr>
<td>crowd.server.url</td>
<td><a href="http://localhost:8095/crowd/services/">http://localhost:8095/crowd/services/</a></td>
</tr>
<tr>
<td>session.validationinterval</td>
<td>This is the time interval between requests which validate whether the user is logged in or out of the Crowd SSO server. Set to 0, if you want authentication checks to occur on each request. Otherwise set to the number of minutes you wish to wait between requests. Setting this value to 1 or higher will increase the performance of Crowd's integration.</td>
</tr>
</tbody>
</table>

Step 3. Configuring your Acegi Application to Use the Crowd Acegi Connector

There are two ways you can integrate your application with Crowd:

- Centralised user management: The user repository available to your application will be the user repository allocated to your application via Crowd. This means that your application will use the centralised user repository for retrieving user details as well as performing authentication.
- Single sign-on: In addition to centralised authentication, SSO will be available to your application. If any other SSO-enabled applications (such as JIRA, Confluence, or your own custom applications) are integrated with Crowd, then SSO behaviour will be established across these applications. If you sign in to one application, you are signed in to all applications. If you sign out of one application, you are signed out of all applications.

First, open the applicationContext.xml file relevant to your application, which contains the Acegi configuration. You are likely to have a bean configuration similar to this snippet:
<bean id="filterChainProxy" class="org.acegisecurity.util.FilterChainProxy">
    <property name="filterInvocationDefinitionSource">
        CONVET_URL_TO_LOWERCASE_BEFORE_COMPARISON
        PATTERN_TYPE_APACHE_ANT
        /images/**=#NONE#
        /scripts/**=#NONE#
        /styles/**=#NONE#
        /**=httpSessionContextIntegrationFilter,logoutFilter,authenticationProcessingFilter,securityContextHolderAwareRequestFilter,rememberMeProcessingFilter,anonymousProcessingFilter,exceptionTranslationFilter,filterInvocationInterceptor
    </property>
</bean>

3.1 Configuring Centralised User Management

Perform the following updates to your Acegi Spring configuration:

1. Add the definition of the CrowdUserDetailsService:

   <bean id="crowdUserDetailsService" class="com.atlassian.crowd.integration.acegi.CrowdUserDetailsService">
   </bean>

2. Add the definition of the CrowdAuthenticationProvider:

   <bean id="crowdAuthenticationProvider" class="com.atlassian.crowd.integration.acegi.CrowdAuthenticationProvider">
       <property name="userDetailsService" ref="crowdUserDetailsService"/>
   </bean>

3. Update the definition of your AuthenticationManager / ProviderManager to use the CrowdAuthenticationProvider. If you need multiple authentication providers, you can append the CrowdAuthenticationProvider to your list.

   <bean id="authenticationManager" class="org.acegisecurity.providers.ProviderManager">
       <property name="providers">
           <list>
               <ref local="crowdAuthenticationProvider"/>
               ...
           </list>
       </property>
   </bean>

Further extensions

- If you have an existing user data model, then you can extend or wrap the CrowdDetailsService to cater for user objects within your application domain.
- If you require users within Crowd to be created in your application's persistence model so that you can store application-specific user data, you can extend the CrowdAuthenticationProvider to create records for successfully authenticated Crowd users.

Crowd's remote API

We recommend that applications do not store the Crowd users locally. Rather, applications should query users via Crowd's remote API.

3.2 Configuring Single Sign-On (SSO)

SSO is optional and requires centralised user management

Single sign-on is optional. If you wish to configure SSO you must first configure centralised user management as described in step 3.1 above.

Perform the following additional updates to your Acegi Spring configuration:
1. Update the definition of the AuthenticationProcessingFilter to use the CrowdAuthenticationProcessingFilter:

```xml
<bean id="authenticationProcessingFilter" class="com.atlassian.crowd.integration.acegi.CrowdAuthenticationProcessingFilter">
    <property name="authenticationManager" ref="authenticationManager" />
    <property name="authenticationFailureUrl" value="/login.jsp?error=true" />
    <property name="defaultTargetUrl" value="/" />
    <property name="filterProcessesUrl" value="/j_security_check" />
    ...
</bean>
```

2. Add the definition of the CrowdLogoutHandler:

```xml
<bean id="crowdLogoutHandler" class="com.atlassian.crowd.integration.acegi.CrowdLogoutHandler" />
```

3. Update the definition of the LogoutFilter to use the CrowdLogoutHandler:

```xml
<bean id="logoutFilter" class="org.acegisecurity.ui.logout.LogoutFilter">
    <constructor-arg value="/index.jsp" />
    <constructor-arg>
        <list>
            <ref bean="rememberMeServices" />
            <ref bean="crowdLogoutHandler" />
            <bean class="org.acegisecurity.ui.logout.SecurityContextLogoutHandler" />
        </list>
    </constructor-arg>
    <property name="filterProcessesUrl" value="/logout.jsp" />
</bean>
```

Step 4. Restarting your Acegi Application

Bounce your application. You should now have centralised authentication and single sign-on with Crowd.

Authorisation

For the purposes of Crowd integration with Acegi, you should map Acegi's roles to Crowd's groups. To put it another way: in order to use Acegi's authorisation features, users in Crowd will have their Acegi roles specified by their group names.

For example if user 'admin' is in the 'crowd-admin' group, then the user 'admin' will be authorised to view pages restricted to the 'crowd-admin' role in Acegi.

Related Topics

- **3.1 Using the Application Browser**
- **3.2 Adding an Application**
- **3.3 Mapping a Directory to an Application**
- **3.4 Specifying which Groups can access an Application**
- **3.5 Specifying an Application's Address or Hostname**
- **3.6 Testing a User's Login to an Application**
- **3.7 Managing an Application's Session**
- **3.8 Deleting or Deactivating an Application**

Crowd Documentation
Integrating AppFuse - a Crowd-Acegi Integration Tutorial

AppFuse provides a sweet starting point for developing web applications. You choose the frameworks, AppFuse generates the skeleton application.

At its core, the web security of AppFuse applications relies on the modular and extensible Acegi authentication framework. In this tutorial, we look at a basic integration of Crowd with Acegi, using an application generated by AppFuse.

Step 1. Get AppFuse

In this tutorial, we will be using the Struts2-basic archetype to create the project, but the other types should be similar. For more information, consult the AppFuse quickstart guide. In particular, it outlines the database requirements for AppFuse.

1. Create the project.

   maven archetype:create -DarchetypeGroupId=org.appfuse.archetypes -DarchetypeArtifactId=appfuse-basic-struts -DremoteRepositories=http://static.appfuse.org/releases -DarchetypeVersion=2.0 -DgroupId=com.mycompany.app -DartifactId=myproject

2. Since we will be editing the core Acegi configuration, we will need the full source code of the application.

   cd myproject
   maven appfuse:full-source

3. Build it.

   maven clean install

4. Run it.

   maven jetty:run-war -Dmaven.test.skip=true

5. Play with it.

   http://localhost:8080/

6. Shut it down.

   ctrl+c

Step 2. Let Crowd Know about AppFuse

Add appfuse as an application via the Crowd Console. See Adding an Application for more information.

Step 3. Add the Crowd Acegi Connector to AppFuse

Open up the pom.xml and add the Crowd Acegi connector as a project dependency:

<dependencies>
<dependency>
<groupId>com.atlassian.crowd</groupId>
<artifactId>crowd-integration-acegi</artifactId>
.VERSION>1.2</version>
</dependency>
...
</dependencies>

You will also need to create the file myproject/src/main/resources/crowd.properties:

application.name: appfuse
application.password: password
application.login.url: http://localhost:8095/crowd/
crowd.server.url: http://localhost:8095/crowd/services/
In particular, the application name and password must match the values defined for the application added in Step 2.

**Step 4. Hook Up Centralised Authentication**

AppFuse neatly stores all the Acegi configuration in `myproject/src/main/webapp/WEB-INF/security.xml`. In order to get centralised authentication, we will need to set up Acegi to use the wrapped authenticator class we just created. Edit the Acegi beans in `security.xml`:

1. Add the definition of the CrowdUserDetailsService:

   ```xml
   <bean id="crowdUserDetailsService" class="com.atlassian.crowd.integration.acegi.CrowdUserDetailsService"/>
   ```

2. Add the definition of the CrowdAuthenticationProvider which will delegate Acegi's authentication requests to Crowd:

   ```xml
   <bean id="crowdAuthenticationProvider" class="com.atlassian.crowd.integration.acegi.CrowdAuthenticationProvider">
   <property name="userDetailsService" ref="crowdUserDetailsService"/>
   </bean>
   ```

3. Replace the DaoAuthenticationProvider with our authenticator in the authentication manager:

   ```xml
   <bean id="authenticationManager" class="org.acegisecurity.providers.ProviderManager">
   <property name="providers">
   <list>
   <ref local="crowdAuthenticationProvider"/>
   <!--<ref local="daoAuthenticationProvider"/>--->
   <ref local="anonymousAuthenticationProvider"/>
   <ref local="rememberMeAuthenticationProvider"/>
   </list>
   </property>
   </bean>
   ```

4. Now do a:

   ```
   mvn jetty:run-war -Dmaven.test.skip=true
   ```

5. Head over to `http://localhost:8080/`. You should now be able to authenticate the users in your Crowd repository that meet all of the following conditions:

   - They are in a Crowd directory assigned to the AppFuse application in Crowd [see more].
   - They are in Crowd groups named `ROLE_USERS` or `ROLE_ADMIN`. You will need to add these groups and assign the user as a member of either group. These Crowd group names map to the Acegi authorisation roles defined in the AppFuse application.
   - They are allowed to authenticate with the AppFuse application because EITHER they are in a group allowed to authenticate with Crowd [see more] OR their container directory allows all users to authenticate [see more].

Congratulations. You have centralised authentication 😊

**Application-level centralised user management**

One quirk you may notice is that you can't view the profile details of users who exist in Crowd, but did not exist in AppFuse prior to the Crowd integration. Although it's possible to authenticate a Crowd user 'dude' and still run AppFuse as 'dude', 'dude' will not be in AppFuse's local database. AppFuse makes use of a database-backed user management system. In order to achieve application-level centralised user management, AppFuse will need to delegate its calls to create, retrieve, update and delete users to Crowd via...
Crowd's remote API. This will prevent data redundancy and eliminate the hassle of data synchronisation. This is beyond the scope of this short tutorial.

Step 5. Hook Up Single Sign-On

Enabling single sign-on (SSO) requires a little more tweaking of the `security.xml`:

1. Change the default processing filter to Crowd's SSO filter:
   ```
   <bean id="authenticationProcessingFilter" class="com.atlassian.crowd.integration.acegi.CrowdAuthenticationProcessingFilter">
   ... 
   </bean>
   ```

2. Add the definition of the CrowdLogoutHandler:
   ```
   <bean id="crowdLogoutHandler" class="com.atlassian.crowd.integration.acegi.CrowdLogoutHandler"/>
   ```

3. Update the definition of the LogoutFilter to use the CrowdLogoutHandler. You may need to uncomment the logout filter.
   ```
   <bean id="logoutFilter" class="org.acegisecurity.ui.logout.LogoutFilter">
   ... 
   <constructor-arg>
   <list>
   <ref bean="rememberMeServices"/>
   <ref bean="crowdLogoutHandler"/>
   <bean class="org.acegisecurity.ui.logout.SecurityContextLogoutHandler"/>
   </list>
   </constructor-arg>
   ... 
   </bean>
   ```

4. If the logout filter is not defined in the filter invocation list, you will need to add it:
   ```
   <bean id="filterChainProxy" class="org.acegisecurity.util.FilterChainProxy">
   <property name="filterInvocationDefinitionSource">
   ... 
   /**=httpSessionContextIntegrationFilter,logoutFilter,authenticationProcessingFilter,securityContextHolderAwareRequestFilter,rememberMeProcessingFilter,anonymousProcessingFilter,exceptionTranslationFilter,filterInvocationInterceptor
   </property>
   ... 
   </bean>
   ```

5. Now repeat:
   ```
   mvn jetty:run-war -Dmaven.test.skip=true
   ```

SSO will only work for users that are able to authenticate with both applications and are authorised to use both applications. Try out the following:

- Log in to Crowd – you should be logged in to AppFuse.
- Log out of AppFuse – you should be logged out of Crowd.
- Log in to AppFuse; log out of Crowd; log in to Crowd as another user; refresh AppFuse – you should be logged in as the new user.

Congratulations, you have SSO 😊
3.2.10 Integrating Crowd with a Custom Application

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. Refer to the description of the attributes in the crowd.properties file.

Related Topics

• 3.1 Using the Application Browser
• 3.2 Adding an Application
  • 3.2.01 Integrating Crowd with Apache
  • 3.2.02 Integrating Crowd with Subversion
  • 3.2.03 Integrating Crowd with Atlassian Confluence
    • 3.2.3.1 Configuring Confluence for NTLM SSO
  • 3.2.04 Integrating Crowd with Atlassian CrowdID
  • 3.2.05 Integrating Crowd with Atlassian FishEye
  • 3.2.06 Integrating Crowd with Atlassian JIRA
    • 3.2.6.1 Configuring JIRA for NTLM SSO
  • 3.2.07 Integrating Crowd with Jive Forums
    • 3.2.7.1 Jive SSO
  • 3.2.08 Integrating Crowd with Atlassian Bamboo
  • 3.2.09 Integrating Crowd with Acegi Security
    • Integrating AppFuse - a Crowd-Acegi Integration Tutorial
  • 3.2.10 Integrating Crowd with a Custom Application
• 3.3 Mapping a Directory to an Application
  • 3.3.1 Specifying the Directory Order for an Application
  • 3.3.2 Specifying an Application’s Directory Permissions
    • Example of Directory Permissions
• 3.4 Specifying which Groups can access an Application
• 3.5 Specifying an Application’s Address or Hostname
• 3.6 Testing a User’s Login to an Application
• 3.7 Managing an Application’s Session
• 3.8 Deleting or Deactivating an Application
3.3 Mapping a Directory to an Application

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 allows 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. Log in 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:

   ![Up-arrow and Down-arrow](image)

   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.
8. Next, you should define the application’s ability to add/update principals in the directory. Click the 'Permissions' tab and set the directory permissions for the application.

Screenshot: 'Application---Map Directories'

Related Topics

- 3.1 Using the Application Browser
- 3.2 Adding an Application
  - 3.2.01 Integrating Crowd with Apache
  - 3.2.02 Integrating Crowd with Subversion
  - 3.2.03 Integrating Crowd with Atlassian Confluence
    - 3.2.3.1 Configuring Confluence for NTLM SSO
  - 3.2.04 Integrating Crowd with Atlassian CrowdID
  - 3.2.05 Integrating Crowd with Atlassian FishEye
3.2.06 Integrating Crowd with Atlassian JIRA
  - 3.2.6.1 Configuring JIRA for NTLM SSO
3.2.07 Integrating Crowd with Jive Forums
  - 3.2.7.1 Jive SSO
3.2.08 Integrating Crowd with Atlassian Bamboo
3.2.09 Integrating Crowd with Acgi Security
  - Integrating AppFuse - a Crowd-Acgi Integration Tutorial
3.2.10 Integrating Crowd with a Custom Application

3.3 Mapping a Directory to an Application
  - 3.3.1 Specifying the Directory Order for an Application
  - 3.3.2 Specifying an Application's Directory Permissions
    - Example of Directory Permissions
3.4 Specifying which Groups can access an Application
3.5 Specifying an Application's Address or Hostname
3.6 Testing a User's Login to an Application
3.7 Managing an Application's Session
3.8 Deleting or Deactivating an Application

Crowd Documentation
3.3.1 Specifying the Directory Order for an Application

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
  - 3.2.01 Integrating Crowd with Apache
  - 3.2.02 Integrating Crowd with Subversion
  - 3.2.03 Integrating Crowd with Atlassian Confluence
    - 3.2.03.1 Configuring Confluence for NTLM SSO
  - 3.2.04 Integrating Crowd with Atlassian CrowdID
  - 3.2.05 Integrating Crowd with Atlassian FishEye
  - 3.2.06 Integrating Crowd with Atlassian JIRA
    - 3.2.06.1 Configuring JIRA for NTLM SSO
  - 3.2.07 Integrating Crowd with Jive Forums
    - 3.2.07.1 Jive SSO
  - 3.2.08 Integrating Crowd with Atlassian Bamboo
  - 3.2.09 Integrating Crowd with Acegi Security
    - Integrating AppFuse - a Crowd-Acegi Integration Tutorial
  - 3.2.10 Integrating Crowd with a Custom Application
- 3.3 Mapping a Directory to an Application
  - 3.3.1 Specifying the Directory Order for an Application
  - 3.3.2 Specifying an Application's Directory Permissions
    - Example of Directory Permissions
- 3.4 Specifying which Groups can access an Application
- 3.5 Specifying an Application's Address or Hostname
- 3.6 Testing a User's Login to an Application
- 3.7 Managing an Application's Session
- 3.8 Deleting or Deactivating an Application

Crowd Documentation
3.3.2 Specifying an Application's Directory Permissions

When you map a directory to an application, you can also define the application’s ability to add/update/delete principals, groups and roles in the directory. To do this, use the 'Permissions' tab in the 'View Application' screen.

Directory permissions are defined at two levels:

1. Directory-level permissions are defined on the 'Permissions' tab of the 'View Directory' screen. These permissions apply to each application mapped to the directory, unless the application has its own application-level permissions.
2. Application-level directory permissions are defined on the 'Permissions' tab of the 'View Application' screen. If a permission is enabled at directory level, you can enable it for a specific application. For example, you could enable the 'Add Principal' permission on the 'Customers' directory in JIRA but disable the permission for Confluence.

Take a look at an example.

Disabling a directory-level permission will override any permissions enabled at application level. If a permission is enabled at application level and then subsequently disabled at directory level, the directory-level permission will apply. (The application-level permissions will be 'remembered' and will apply again if re-enabled at directory level.)

How do directory permissions affect the Crowd application (Crowd Administration Console)?

- If a particular permission is turned off at directory level, then no application can perform the related function - not even the Crowd application. So, for example, if you disable the 'Remove Principal' permission for a directory, then the Crowd Administration Console will not allow you to delete a principal from that directory.
- The Crowd application is not bound by application-level permissions.

For details on directory-level permissions, refer to the instructions on specifying directory permissions. Below are instructions on setting the application-level directory permissions.

<table>
<thead>
<tr>
<th>Permission</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add Group</td>
<td>Allows the application to add groups to the selected directory.</td>
</tr>
<tr>
<td>Add Principal</td>
<td>Allows the application to add principals to the selected directory.</td>
</tr>
<tr>
<td>Add Role</td>
<td>Allows the application to add roles to the selected directory.</td>
</tr>
<tr>
<td>Modify Group</td>
<td>Allows the application to modify groups in the selected directory.</td>
</tr>
<tr>
<td>Modify Principal</td>
<td>Allows the application to modify principals in the selected directory.</td>
</tr>
<tr>
<td>Modify Role</td>
<td>Allows the application to modify roles in the selected directory.</td>
</tr>
<tr>
<td>Remove Group</td>
<td>Allows the application to delete groups from the selected directory.</td>
</tr>
<tr>
<td>Remove Principal</td>
<td>Allows the application to delete principals from the selected directory.</td>
</tr>
</tbody>
</table>

Consider carefully whether you allow the deletion of principals, as some applications...
When you initially map a directory to an application, all of the application’s permissions are enabled by default. But note that disabling a directory-level permission will override any permissions enabled at application level.

To set the directory permissions for 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 next to the application you wish to update.
4. This will display the 'View Application' screen. Click the 'Permissions' tab.
5. This will display a list of directories that are currently mapped to the application, and a set of permission check-boxes. Select a directory from the list on the left.
6. The 'Permissions' check-boxes will change to show the application’s existing permissions for that directory.
   - To enable a directory permission, select the corresponding check-box.
   - To disable a directory permission, deselect the corresponding check-box.

On the application permissions screen, the words ‘(disabled globally)’ will appear next to any permission that is disabled at directory level.

Related Topics

- 2.3 Specifying Directory Permissions
- 3.1 Using the Application Browser
- 3.2 Adding an Application
3.2.01 Integrating Crowd with Apache
3.2.02 Integrating Crowd with Subversion
3.2.03 Integrating Crowd with Atlassian Confluence
   - 3.2.3.1 Configuring Confluence for NTLM SSO
3.2.04 Integrating Crowd with Atlassian CrowdID
3.2.05 Integrating Crowd with Atlassian FishEye
3.2.06 Integrating Crowd with Atlassian JIRA
   - 3.2.6.1 Configuring JIRA for NTLM SSO
3.2.07 Integrating Crowd with Jive Forums
   - 3.2.7.1 Jive SSO
3.2.08 Integrating Crowd with Atlassian Bamboo
3.2.09 Integrating Crowd with Acegi Security
   - Integrating AppFuse - a Crowd-Acegi Integration Tutorial
3.2.10 Integrating Crowd with a Custom Application

3.3 Mapping a Directory to an Application
   - 3.3.1 Specifying the Directory Order for an Application
   - 3.3.2 Specifying an Application’s Directory Permissions
      - Example of Directory Permissions
3.4 Specifying which Groups can access an Application
3.5 Specifying an Application’s Address or Hostname
3.6 Testing a User’s Login to an Application
3.7 Managing an Application’s Session
3.8 Deleting or Deactivating an Application

Crowd Documentation
Example of Directory Permissions

Let's assume that you want to:

- Allow self-registration (automatic signup) of new users in your 'Customers' directory via JIRA, and
- Disable self-registration via Confluence.

Here's how you would set the directory-level and application-level permissions in Crowd.

1. At directory level, enable the 'Add Principal' permission (and any other permissions you want):
   a. In the Crowd Administration Console, click the 'Directories' link in the top navigation bar.
   b. Select the 'Customers' directory.
   c. Click the 'Permissions' tab.
   d. Select the 'Add Principal' check-box.

2. At application level, make sure the 'Add Principal' permission is enabled for the JIRA application:
   a. Click the 'Applications' link in the top navigation bar.
   b. Click the 'View' link next to the JIRA application.
   c. In the 'View Application' screen, click the 'Permissions' tab.
   d. Select the 'Customers' directory.
   e. Select the 'Add Principal' check-box.

3. At application level, disable the 'Add Principal' permission the Confluence application:
   a. Click the 'Applications' link in the top navigation bar.
   b. Click the 'View' link next to the Confluence application.
   c. Click the 'Permissions' tab.
   d. Select the 'Customers' directory.
   e. Deselect the 'Add Principal' check-box.
In summary:
With the above application permissions, a person will be able to sign up for a user account via JIRA and this user will be created in the 'Customers' directory, but they will not be able to sign up for an account via Confluence.

Related Topics

- [2.3 Specifying Directory Permissions](#)
- [3.3.2 Specifying an Application's Directory Permissions](#)

Crowd Documentation
3.4 Specifying which Groups can access an Application

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
  - 3.2.01 Integrating Crowd with Apache
  - 3.2.02 Integrating Crowd with Subversion
3.2.03 Integrating Crowd with Atlassian Confluence
   - 3.2.3.1 Configuring Confluence for NTLM SSO
3.2.04 Integrating Crowd with Atlassian CrowdID
3.2.05 Integrating Crowd with Atlassian FishEye
3.2.06 Integrating Crowd with Atlassian JIRA
   - 3.2.6.1 Configuring JIRA for NTLM SSO
3.2.07 Integrating Crowd with Jive Forums
   - 3.2.7.1 Jive SSO
3.2.08 Integrating Crowd with Atlassian Bamboo
3.2.09 Integrating Crowd with Acegi Security
   - Integrating AppFuse - a Crowd-Acegi Integration Tutorial
3.2.10 Integrating Crowd with a Custom Application

3.3 Mapping a Directory to an Application
   - 3.3.1 Specifying the Directory Order for an Application
   - 3.3.2 Specifying an Application’s Directory Permissions
   - Example of Directory Permissions

3.4 Specifying which Groups can access an Application
3.5 Specifying an Application’s Address or Hostname
3.6 Testing a User’s Login to an Application
3.7 Managing an Application’s Session
3.8 Deleting or Deactivating an Application

Crowd Documentation
3.5 Specifying an Application's Address or Hostname

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.

**Screenshot: 'Application---Addresses'**

![View Application -- crowd](image)

<table>
<thead>
<tr>
<th>Address</th>
<th>Status</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>192.168.0.110</td>
<td>True</td>
<td>Remove</td>
</tr>
<tr>
<td>127.0.0.1</td>
<td>True</td>
<td>Remove</td>
</tr>
<tr>
<td>localhost</td>
<td>True</td>
<td>Remove</td>
</tr>
</tbody>
</table>

**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](#)
  - [3.2.01 Integrating Crowd with Apache](#)
  - [3.2.02 Integrating Crowd with Subversion](#)
  - [3.2.03 Integrating Crowd with Atlassian Confluence](#)
    - [3.2.3.1 Configuring Confluence for NTLM SSO](#)
  - [3.2.04 Integrating Crowd with Atlassian CrowdID](#)
  - [3.2.05 Integrating Crowd with Atlassian FishEye](#)
  - [3.2.06 Integrating Crowd with Atlassian JIRA](#)
    - [3.2.6.1 Configuring JIRA for NTLM SSO](#)
  - [3.2.07 Integrating Crowd with Jive Forums](#)
    - [3.2.7.1 Jive SSO](#)
  - [3.2.08 Integrating Crowd with Atlassian Bamboo](#)
  - [3.2.09 Integrating Crowd with Acegi Security](#)
    - [Integrating AppFuse - a Crowd-Acegi Integration Tutorial](#)
• 3.2.10 Integrating Crowd with a Custom Application
• 3.3 Mapping a Directory to an Application
  ° 3.3.1 Specifying the Directory Order for an Application
  ° 3.3.2 Specifying an Application’s Directory Permissions
    - Example of Directory Permissions
• 3.4 Specifying which Groups can access an Application
• 3.5 Specifying an Application’s Address or Hostname
• 3.6 Testing a User’s Login to an Application
• 3.7 Managing an Application’s Session
• 3.8 Deleting or Deactivating an Application

Crowd Documentation
3.6 Testing a User's Login to an Application

You can use an application's 'Config Test' tab to verify that a user will be able to log in to a given application, based on the principal, directory and group associations in Crowd.

Performing the Test

The test works like this:

1. You enter the username and password of the user you wish to verify has access to a given application.
2. Crowd searches for the principal with that username in the application's mapped directories, and verifies the password.
3. If the principal is not found or the password is invalid, the authentication fails the test.
4. Crowd checks whether the directory is set to allow all to authenticate.
5. If all can authenticate, the test passes.
6. Else, Crowd checks the group(s) to which the principal belongs and verifies whether those groups have access to the application.
7. If the user belongs to an allowed group, the test passes, otherwise it fails.

To test a user's login to an application,

1. Log in 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 verify.
4. This will display the 'View Application' screen. Click the 'Config Test' tab.
5. Enter the 'Username' and 'Password' that you wish to verify.
6. Click the 'Update' button.
7. A message appears above the 'Username', displaying one of the following:
   • 'Successful verification' – The authentication has passed the test.
   • 'Invalid verification' – The authentication has failed the test.

Below are some suggestions for the next steps you can take in each case.

Screenshot: 'Application---Config Test showing successful verification'

Successful Verification

If this test is successful, but the user is having trouble authenticating to an application, then the problem is caused by the connection between the application and Crowd rather than by user authentication.

Next step: Check the 'Application Sessions' tab in the Session Browser to see if the application is connected to Crowd.
Failed Verification

If the test declares the login to be invalid, this means that the configuration is incorrect within Crowd.

Next steps:
Check the following - all must be true to allow successful verification.

- The principal must belong to a directory which is mapped to this application.
- The password you used must be valid. In particular, check that the password is the one specified in the first directory in which the principal appears. (If the principal belongs to more than one directory, Crowd uses the first directory in which the principal appears, as determined by the directory order.)
- Either:
  - The directory must be set to allow all to authenticate.
  OR:
  - The principal must belong to a group which has access to the application.

Related Topics

- 3.1 Using the Application Browser
- 3.2 Adding an Application
  - 3.2.01 Integrating Crowd with Apache
  - 3.2.02 Integrating Crowd with Subversion
  - 3.2.03 Integrating Crowd with Atlassian Confluence
    - 3.2.3.1 Configuring Confluence for NTLM SSO
  - 3.2.04 Integrating Crowd with Atlassian CrowdID
  - 3.2.05 Integrating Crowd with Atlassian FishEye
  - 3.2.06 Integrating Crowd with Atlassian JIRA
    - 3.2.6.1 Configuring JIRA for NTLM SSO
  - 3.2.07 Integrating Crowd with Jive Forums
    - 3.2.7.1 Jive SSO
  - 3.2.08 Integrating Crowd with Atlassian Bamboo
  - 3.2.09 Integrating Crowd with Acegi Security
    - Integrating AppFuse - a Crowd-Acegi Integration Tutorial
  - 3.2.10 Integrating Crowd with a Custom Application
- 3.3 Mapping a Directory to an Application
  - 3.3.1 Specifying the Directory Order for an Application
  - 3.3.2 Specifying an Application’s Directory Permissions
    - Example of Directory Permissions
- 3.4 Specifying which Groups can access an Application
- 3.5 Specifying an Application’s Address or Hostname
- 3.6 Testing a User’s Login to an Application
- 3.7 Managing an Application’s Session
- 3.8 Deleting or Deactivating an Application

Crowd Documentation
3.7 Managing an Application's Session

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. Log in 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'

<table>
<thead>
<tr>
<th>Application Sessions</th>
<th>Principal Sessions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name:</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

To force an application to log out of Crowd,

1. Log in 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 3.8 Deleting or Deactivating an Application.

Related Topics

- 3.1 Using the Application Browser
- 3.2 Adding an Application
  - 3.2.01 Integrating Crowd with Apache
  - 3.2.02 Integrating Crowd with Subversion
  - 3.2.03 Integrating Crowd with Atlassian Confluence
    - 3.2.3.1 Configuring Confluence for NTLM SSO
  - 3.2.04 Integrating Crowd with Atlassian CrowdID
  - 3.2.05 Integrating Crowd with Atlassian FishEye
  - 3.2.06 Integrating Crowd with Atlassian JIRA
    - 3.2.6.1 Configuring JIRA for NTLM SSO
  - 3.2.07 Integrating Crowd with Jive Forums
    - 3.2.7.1 Jive SSO
  - 3.2.08 Integrating Crowd with Atlassian Bamboo
  - 3.2.09 Integrating Crowd with Acegi Security
- Integrating AppFuse - a Crowd-Acegi Integration Tutorial
  - 3.2.10 Integrating Crowd with a Custom Application
- 3.3 Mapping a Directory to an Application
  - 3.3.1 Specifying the Directory Order for an Application
  - 3.3.2 Specifying an Application's Directory Permissions
    - Example of Directory Permissions
- 3.4 Specifying which Groups can access an Application
- 3.5 Specifying an Application's Address or Hostname
- 3.6 Testing a User's Login to an Application
- 3.7 Managing an Application's Session
- 3.8 Deleting or Deactivating an Application

- 4.04 Managing a Principal's Session
- 5.2.5 Session Timeout

Crowd Documentation
3.8 Deleting or Deactivating an Application

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. Log in 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. Log in 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
  - 3.2.01 Integrating Crowd with Apache
  - 3.2.02 Integrating Crowd with Subversion
  - 3.2.03 Integrating Crowd with Atlassian Confluence
    - 3.2.3.1 Configuring Confluence for NTLM SSO
  - 3.2.04 Integrating Crowd with Atlassian CrowdID
  - 3.2.05 Integrating Crowd with Atlassian FishEye
  - 3.2.06 Integrating Crowd with Atlassian JIRA
    - 3.2.6.1 Configuring JIRA for NTLM SSO
  - 3.2.07 Integrating Crowd with Jive Forums
    - 3.2.7.1 Jive SSO
  - 3.2.08 Integrating Crowd with Atlassian Bamboo
  - 3.2.09 Integrating Crowd with Acegi Security
    - Integrating AppFuse - a Crowd-Acegi Integration Tutorial
  - 3.2.10 Integrating Crowd with a Custom Application
- 3.3 Mapping a Directory to an Application
  - 3.3.1 Specifying the Directory Order for an Application
  - 3.3.2 Specifying an Application's Directory Permissions
    - Example of Directory Permissions
- 3.4 Specifying which Groups can access an Application
- 3.5 Specifying an Application's Address or Hostname
- 3.6 Testing a User’s Login to an Application
• 3.7 Managing an Application's Session
• 3.8 Deleting or Deactivating an Application

Crowd Documentation
4. Managing Principals, Groups and Roles

This page last changed on Oct 25, 2007 by smaddox.

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. Note also that the crowd-administrators group confers Crowd administration rights to its members.

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.

- 4.01 Using the Principal Browser
- 4.02 Adding a Principal
- 4.03 Deleting or Deactivating a Principal
- 4.04 Managing a Principal's Session
- 4.05 Editing a Principal's Details and Password
- 4.06 Specifying a Principal's Attributes
- 4.07 Editing a Principal's Group and Role Membership
- 4.08 Granting Crowd Administration Rights to a User
- 4.09 Using the Group Browser and Role Browser
- 4.10 Adding a Group or Role
4.01 Using the Principal Browser

This page last changed on Jun 12, 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

- 4.01 Using the Principal Browser
- 4.02 Adding a Principal
- 4.03 Deleting or Deactivating a Principal
- 4.04 Managing a Principal's Session
- 4.05 Editing a Principal's Details and Password
- 4.06 Specifying a Principal's Attributes
- 4.07 Editing a Principal's Group and Role Membership
- 4.08 Granting Crowd Administration Rights to a User
- 4.09 Using the Group Browser and Role Browser
- 4.10 Adding a Group or Role

Crowd Documentation
4.02 Adding a Principal

In Crowd, users are referred to as principal entity objects (or just principals). Principals can either be imported into Crowd in bulk (see 2.4 Importing Principals and Groups into a Directory), or added individually as follows.

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. Select the directory in which to create the new principal.
5. Complete the fields as described in the table below, then click the 'Create' button.
6. After creating the principal, you will be able to specify the principal's attributes and group/role membership. If you wish, you can also verify that the user can log in to appropriate applications.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Email</td>
<td>The email address of the principal. Email addresses must follow the RFC2822 format.</td>
</tr>
<tr>
<td>Active</td>
<td>Only deselect this if you wish to deny access to the principal.</td>
</tr>
<tr>
<td>Username</td>
<td>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.</td>
</tr>
<tr>
<td>Password</td>
<td>The password of the principal.</td>
</tr>
<tr>
<td>First Name</td>
<td>The first name of the principal.</td>
</tr>
<tr>
<td>Last Name</td>
<td>The last name of the principal.</td>
</tr>
<tr>
<td>Directory</td>
<td>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.</td>
</tr>
</tbody>
</table>

Screenshot 1: 'Principal Browser'

<table>
<thead>
<tr>
<th>Applications</th>
<th>Principals</th>
<th>Groups</th>
<th>Roles</th>
<th>Sessions</th>
<th>Directory</th>
<th>Options</th>
<th>System Info</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Principals Browser</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Add Principal | Import Users**

Directory: All

Username: [Input Field]

Email: [Input Field]

Active: [Input Field]

Screenshot 2: 'Add Principal'
### Add Principal

- **Email:**
  - Email addresses must follow the RFC2822 format.
- **Active:**
- **Username:**
  - The unique name of the principal.
- **Password:**
- **Confirm Password:**
- **FirstName:**
- **LastName:**
- **Directory:**
  - The directory the principal belongs to.

**Create** | **Cancel**

---

**Related Topics**

- 4.01 Using the Principal Browser
- 4.02 Adding a Principal
- 4.03 Deleting or Deactivating a Principal
- 4.04 Managing a Principal's Session
- 4.05 Editing a Principal's Details and Password
- 4.06 Specifying a Principal's Attributes
- 4.07 Editing a Principal's Group and Role Membership
- 4.08 Granting Crowd Administration Rights to a User
- 4.09 Using the Group Browser and Role Browser
- 4.10 Adding a Group or Role

[Crowd Documentation](#)
4.03 Deleting or Deactivating a Principal

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.

Consider deactivating instead of deleting

We recommend that you deactivate a principal rather than delete them, in case some applications contain historical data, e.g. documents that the user has created. Read more.

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

1. 4.01 Using the Principal Browser
2. 4.02 Adding a Principal
3. 4.03 Deleting or Deactivating a Principal
4. 4.04 Managing a Principal's Session
5. 4.05 Editing a Principal's Details and Password
6. 4.06 Specifying a Principal's Attributes
7. 4.07 Editing a Principal's Group and Role Membership
8. 4.08 Granting Crowd Administration Rights to a User
9. 4.09 Using the Group Browser and Role Browser
10. 4.10 Adding a Group or Role

Crowd Documentation
4.04 Managing a Principal's Session

This page last changed on Nov 06, 2007 by smaddox.

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 4.03 Deleting or Deactivating a Principal.

See Also...

3.7 Managing an Application's Session
5.2.5 Session Timeout

Related Topics

• 4.01 Using the Principal Browser
• 4.02 Adding a Principal
• 4.03 Deleting or Deactivating a Principal
• 4.04 Managing a Principal's Session
• 4.05 Editing a Principal's Details and Password
• 4.06 Specifying a Principal's Attributes
• 4.07 Editing a Principal's Group and Role Membership
• 4.08 Granting Crowd Administration Rights to a User
• 4.09 Using the Group Browser and Role Browser
• 4.10 Adding a Group or Role
4.05 Editing a Principal's Details and Password

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

• 4.01 Using the Principal Browser
• 4.02 Adding a Principal
4.03 Deleting or Deactivating a Principal
4.04 Managing a Principal's Session
4.05 Editing a Principal's Details and Password
4.06 Specifying a Principal's Attributes
4.07 Editing a Principal's Group and Role Membership
4.08 Granting Crowd Administration Rights to a User
4.09 Using the Group Browser and Role Browser
4.10 Adding a Group or Role

Crowd Documentation
4.06 Specifying a Principal's Attributes

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

- 4.01 Using the Principal Browser
- 4.02 Adding a Principal
- 4.03 Deleting or Deactivating a Principal
- 4.04 Managing a Principal's Session
- 4.05 Editing a Principal's Details and Password
- 4.06 Specifying a Principal's Attributes
- 4.07 Editing a Principal's Group and Role Membership
- 4.08 Granting Crowd Administration Rights to a User
- 4.09 Using the Group Browser and Role Browser
• **4.10 Adding a Group or Role**

Crowd Documentation
4.07 Editing a Principal's Group and Role Membership

Within any given directory, you can choose the groups and roles to which each principal (i.e. 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 add, 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 from 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 remove, 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

- 4.01 Using the Principal Browser
- 4.02 Adding a Principal
- 4.03 Deleting or Deactivating a Principal
- 4.04 Managing a Principal's Session
- 4.05 Editing a Principal's Details and Password
- 4.06 Specifying a Principal's Attributes
- 4.07 Editing a Principal's Group and Role Membership
4.08 Granting Crowd Administration Rights to a User
4.09 Using the Group Browser and Role Browser
4.10 Adding a Group or Role

Crowd Documentation
4.08 Granting Crowd Administration Rights to a User

Members of the 'crowd-administrators' group have administration privileges — that is, the ability to:

- access the Crowd Administration Console and perform the functions described in the Crowd Administration Guide
- access the CrowdID 'Administration' menu and perform the functions described in the CrowdID Administration Guide.

The 'crowd-administrators' group is automatically created in your 'Default Directory' when you install Crowd (see 3. Running the Setup Wizard). If you need to grant Crowd administration rights to users in other directories, you can create a 'crowd-administrators' group in any or all of your other directories.

To grant administration privileges to a user,

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 which contains the principal (i.e. user) to whom you wish to grant administration rights.
4. Locate the principal, and click the 'View' link that corresponds to the principal.
5. This will display the 'Principal Details' screen. Click the 'Groups' tab.
6. A list of the principal's current groups (if any) will be displayed. Select the 'crowd-administrators' group from the drop-down box below the list, then click the 'Add' button.

If you wish, you can use a different or additional group to contain your Crowd administrators. To do this, map your chosen group(s) to the 'crowd' application as described in 3.4 Specifying which Groups can access an Application. Note that CrowdID administrators, however, must always belong to the 'crowd-administrators' groups.

RELATED TOPICS

- 4.01 Using the Principal Browser
- 4.02 Adding a Principal
- 4.03 Deleting or Deactivating a Principal
- 4.04 Managing a Principal's Session
- 4.05 Editing a Principal's Details and Password
- 4.06 Specifying a Principal's Attributes
- 4.07 Editing a Principal's Group and Role Membership
- 4.08 Granting Crowd Administration Rights to a User
- 4.09 Using the Group Browser and Role Browser
- 4.10 Adding a Group or Role

Crowd Documentation
4.09 Using the Group Browser and Role Browser

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. Note also that the crowd-administrators group confers Crowd administration rights to its members.

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. Log in 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.
5. To view the principals who are members of the group, click the ‘Principals’ tab. (This tab may take a while to load, depending upon the size of your user base.)

Screenshot 1: 'Group Browser'

![Group Browser Screenshot]

Screenshot 2: 'View/Update Group Details'

![View Group - Partners Screenshot]

Screenshot 3: 'View Members of a Group'

![View Members of a Group Screenshot]
Related Topics

- 4.01 Using the Principal Browser
- 4.02 Adding a Principal
- 4.03 Deleting or Deactivating a Principal
- 4.04 Managing a Principal's Session
- 4.05 Editing a Principal's Details and Password
- 4.06 Specifying a Principal's Attributes
- 4.07 Editing a Principal's Group and Role Membership
- 4.08 Granting Crowd Administration Rights to a User
- 4.09 Using the Group Browser and Role Browser
- 4.10 Adding a Group or Role

Crowd Documentation
4.10 Adding a Group or Role

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. Note also that the crowd-administrators group confers Crowd administration rights to its members.

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.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>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.</td>
</tr>
<tr>
<td>Description</td>
<td>A short description of the group or role.</td>
</tr>
<tr>
<td>Directory</td>
<td>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.</td>
</tr>
<tr>
<td>Active</td>
<td>Only deselect this if you wish to deny access to all members of the group or role.</td>
</tr>
</tbody>
</table>

**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

- 4.01 Using the Principal Browser
- 4.02 Adding a Principal
- 4.03 Deleting or Deactivating a Principal
- 4.04 Managing a Principal's Session
- 4.05 Editing a Principal's Details and Password
- 4.06 Specifying a Principal's Attributes
- 4.07 Editing a Principal's Group and Role Membership
- 4.08 Granting Crowd Administration Rights to a User
- 4.09 Using the Group Browser and Role Browser
- 4.10 Adding a Group or Role

Crowd Documentation
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
5.1 Viewing Crowd's System Information

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'

<table>
<thead>
<tr>
<th>System Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date: Monday, 02 Apr 2007</td>
</tr>
<tr>
<td>Time: 15:18:40</td>
</tr>
<tr>
<td>Timezone: Eastern Standard Time (New South Wales)</td>
</tr>
<tr>
<td>Java Version: 1.5.0_06</td>
</tr>
<tr>
<td>Java Vendor: Sun Microsystems Inc.</td>
</tr>
<tr>
<td>JVM Version: 1.5.0_06-b05</td>
</tr>
<tr>
<td>JVM Vendor: Sun Microsystems Inc.</td>
</tr>
<tr>
<td>JVM Runtime: Java HotSpot(TM) Client VM</td>
</tr>
<tr>
<td>Username: administrator</td>
</tr>
<tr>
<td>Operating System: Windows 2000sp2</td>
</tr>
<tr>
<td>Architecture: x86</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>JVM Statistics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Memory: 63 MB</td>
</tr>
<tr>
<td>Used Memory: 47 MB</td>
</tr>
<tr>
<td>Free Memory: 15 MB</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Runtime Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application Server: Apache Tomcat5.5.20</td>
</tr>
<tr>
<td>Database Dialect: org.hibernate.dialect.HSQLDialect</td>
</tr>
<tr>
<td>Version: 1.0.3</td>
</tr>
<tr>
<td>Build Number: 104</td>
</tr>
<tr>
<td>Build Date: Mar 22, 2007</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>License Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>License Server ID: ANB2-YN5T - ANB2-YN5T</td>
</tr>
</tbody>
</table>
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
    - 5.2.6.1 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

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

Crowd licenses are based on the number of end-users who will log in to 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 or on the Atlassian website. You will need to enter your license key into your Crowd server as described below.

![Note:

If the number of users who are allowed to log in to the Crowd framework exceeds the user license limit, no-one will be able to log in 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.
- We recommend that you allow only particular groups to log in to each application, rather than entire directories.

![Note:

Note that a mapped application can 'see' all users in a directory, even if not all of them can log in 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 log in to the application.]()

To enter your license key,

1. Log in 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.

The Licensing screen shows the number of users currently counted towards your license. This total is updated automatically at regular intervals. If you have recently added or removed users, the total may not be up to date when you view the screen. You can update the count immediately, as described below.

To recalculate your user total,

1. Log in to the Crowd Administration Console.
2. Click the 'Options' link in the top navigation bar.
3. Click the 'Licensing' tab.
4. Click the link labelled 'Recalculate your user total'.
   - The recalculation may take a while, depending on the size of your user base.
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

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
5.2.3 Domain

The Domain is used when setting HTTP authentication cookies in a user's browser. If this attribute is not 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

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

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...

3.7 Managing an Application’s Session
4.04 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

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

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.

```xml
<ehcache>
  <diskStore path="java.io.tmpdir"/>

  <defaultCache
    maxElementsInMemory="50000"
    eternal="false"
    overflowToDisk="false"
    timeToIdleSeconds="300"
    timeToLiveSeconds="300"
    diskPersistent="false"
    diskExpireThreadIntervalSeconds="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 Jun 12, 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

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
5.4 Backing Up and Restoring Data

You can back up your Crowd data by exporting it to an XML file. The 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 back up your data regularly, especially after any significant configuration changes. It is also recommended that you perform regular backups of your database.

To back up your Crowd data,

1. Log in to the Crowd Administration Console.
2. Click the 'Backup & Restore' link in the top navigation bar.
3. Click the 'Backup' tab.
4. Select the 'Reset Domain' checkbox if the backup file will be restored onto a different server.
   Selecting 'Reset Domain' will reset the domain to blank. (After you restore the data, you can change the domain as described in 5.2.3 Domain.)
5. Type an appropriate 'Backup File Path', including the name of the XML file.
6. 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. Log in 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.
  5. Click the 'Submit' button.

Screenshot 1: 'Backup'

![Screenshot 1: 'Backup'](image1)

Screenshot 2: 'Restore'

![Screenshot 2: 'Restore'](image2)
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
    - 5.2.6.1 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

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.

File Location

The Crowd logging file will by default be available here:

`atlassian-crowd-1.1.X\atlassian-crowd.log`

Change the Logging Level

1. With a text editor, open `crowd-webapp/WEB-INF/classes/log4j.properties`.
   - If you wish to adjust the logging levels of the CrowdID OpenID server, you will need to modify the `crowd-openidserver-webapp/WEB-INF/classes/log4j.properties` file.
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
    - Maven 2 Integration
  - SOAP API
    - Axis Client Stub Generation
    - Microsoft .NET Client
- Creating a Custom Directory Connector
Creating a Crowd Client for your Custom Application

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
  - Maven 2 Integration
- SOAP API
  - Axis Client Stub Generation
  - Microsoft .NET Client

Next Steps:

After creating your Crowd Client, please see 3.2.10 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
    - Maven 2 Integration
  - SOAP API
    - Axis Client Stub Generation
    - Microsoft .NET Client
- Creating a Custom Directory Connector

Crowd Documentation
Application Integration Overview

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**
  - Maven 2 Integration
- **SOAP API**
  - Axis Client Stub Generation
  - [Microsoft .NET Client](#)

[Crowd Documentation](#)
Sample Application ('demo')

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, and is (optionally) configured when you run the Setup Wizard.

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.

Related Topics

- Application Integration Overview
  - Sample Application ('demo')
- Java Integration Libraries
  - Maven 2 Integration
- SOAP API
  - Axis Client Stub Generation
  - Microsoft .NET Client

Crowd Documentation
Java Integration Libraries

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 reads in the crowd.properties configuration file from your application's class path, setting client specific details such as the Crowd server URL and SSO integration details. When the client is loaded into memory, it will then authenticate the the client application with the Crowd security server for future SOAP requests.

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


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.logoff(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:

```xml
<filter>
<filter-name>VerifyTokenFilter</filter-name>
```
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:

```java
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 to the login page
    return SUCCESS;
}
```

**Related Topics**

- Application Integration Overview
  - Sample Application ('demo')
- Java Integration Libraries
  - Maven 2 Integration
- SOAP API
  - Axis Client Stub Generation
  - Microsoft .NET Client

[Link to Crowd Documentation]
Maven 2 Integration

To integrate Crowd with your Maven 2 project, you will need to include the following dependency in your pom.xml:

```xml
<dependency>
  <groupId>com.atlassian.crowd</groupId>
  <artifactId>crowd-client-libraries</artifactId>
  <version>1.2</version>
  <type>pom</type>
</dependency>
```

Because the Crowd libraries are not published to the standard Maven repository, you will need to add Atlassian's public repository:

```xml
<repositories>
  <repository>
    <id>central</id>
    <url>https://m2proxy.atlassian.com/repository/public</url>
    <snapshots>
      <enabled>true</enabled>
      <updatePolicy>always</updatePolicy>
    </snapshots>
    <releases>
      <enabled>true</enabled>
    </releases>
  </repository>
</repositories>
```
This page provides sample code for creating a Crowd Client using the SOAP API.

⚠️ The Crowd API has been tested with: Axis 1/2, Microsoft .NET and XFire.

The SOAP WSDL is available on the following URL for the Crowd Standalone version (after you have downloaded and installed Crowd Standalone):


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


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](https://docs.atlassian.com/crowd/current/com/atlassian/crowd/integration/service/soap/server/SecurityServer.html) 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: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:Body>
    <authenticateApplicationResponse xmlns="urn:SecurityServer">
      <out>
        <name xmlns="http://authentication.integration.crowd.atlassian.com">jira</name>
        <token xmlns="http://authentication.integration.crowd.atlassian.com">9vN5haoWY+wGBs3XitqAIg==</token>
      </out>
    </authenticateApplicationResponse>
  </soap:Body>
</soap:Envelope>
```
authenticatePrincipal - Authenticating a Principal

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

```xml
  <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+xGBs3XitqAIg==</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 with the token for the now authenticated user:

```xml
  <soap:Body>
    <authenticatePrincipalResponse xmlns="urn:SecurityServer">
      <out>o7MSozJJbKQttOLvC4hN2w==</out>
    </authenticatePrincipalResponse>
  </soap:Body>
</soap:Envelope>
```

An invalid authentication attempt will look like the following:

```xml
  <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 look up a user by a token or their name. The example below shows looking up a principal by their authenticated token:

```xml
  <soap:Body>
    <findPrincipalByName xmlns="urn:SecurityServer">
      <name xmlns="http://authentication.integration.crowd.atlassian.com">jstepka</name>
    </findPrincipalByName>
  </soap:Body>
</soap:Envelope>
```
The server lookup response for the principal token:

```xml
  <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">116940408520</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>
        <description xmlns="http://soap.integration.crowd.atlassian.com" xsi:nil="true"/>
      </out>
    </findPrincipalByNameResponse>
  </soap:Body>
</soap:Envelope>
```
Related Topics

- Application Integration Overview
  - Sample Application ('demo')
- Java Integration Libraries
  - Maven 2 Integration
- 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
```

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

```
-rw-r--r-- 1 jstepka jstepka 204 Apr 19 16:56 SecurityServer_pkg
-rw-r--r-- 1 jstepka jstepka 102 Apr 19 16:55 com
-rw-r--r-- 1 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:

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

1. Open a Microsoft Visual Studio .NET Command Prompt.
2. Run the following command to generate a proxy class (change the location of the WSDL according to your installation):
   
   ```
   
   (Note: Ignore any schema validation warnings returned here.)
   ```
3. Compile the generated class with the following references:
   
   ```
   
   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
  - Maven 2 Integration
- SOAP API
  - Axis Client Stub Generation
  - Microsoft .NET Client

Crowd Documentation
Creating a Custom Directory Connector

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.


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
  - Maven 2 Integration
- SOAP API
  - Axis Client Stub Generation
  - Microsoft .NET Client
- Creating a Custom Directory Connector
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 Aug 13, 2007 by rosie@atlassian.com.

⚠️ Crowd 1.2 has now been released — see the Crowd 1.2 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 1.2 Release Notes
- Crowd 1.1.2 Release Notes
- Crowd 1.1.1 Release Notes
- Crowd 1.1.0 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
Crowd 0.2 Beta Release Notes

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

Crowd 1.2 has now been released — see the Crowd 1.2 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 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 1.2 has now been released — see the Crowd 1.2 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 1.2 has now been released — see the Crowd 1.2 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

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;
```

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 1.2 has now been released — see the Crowd 1.2 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 1.2 has now been released — see the Crowd 1.2 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 1.2 has now been released — see the Crowd 1.2 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 1.2 has now been released — see the Crowd 1.2 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 1.2 has now been released — see the Crowd 1.2 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 1.2 has now been released — see the Crowd 1.2 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.

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

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 1.2 has now been released — see the Crowd 1.2 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

<table>
<thead>
<tr>
<th>Key</th>
<th>Summary</th>
<th>Pr</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>CWD-190</td>
<td>XFireFault exception: &quot;No write method for property&quot;.</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CWD-189</td>
<td>Create new group/role broken using OpenLDAP</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CWD-82</td>
<td>Single sign on Seraph authentication fails when the host on a domain is not the same.</td>
<td></td>
<td>Closed</td>
</tr>
</tbody>
</table>

Cheers,

The Atlassian Crowd Development Team
Crowd 1.2 has now been released — see the Crowd 1.2 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](http://www.atlassian.com/Crowd)

<table>
<thead>
<tr>
<th>Key</th>
<th>Summary</th>
<th>Pr</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>CWD-199</td>
<td>Missing libraries from the Crowd distribution</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CWD-198</td>
<td>I renamed the docs from &quot;Documentation&quot; to &quot;Crowd Documentation&quot; (sorry). Can you please fix the &quot;Help link?</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CWD-197</td>
<td>XFire service input and output logging.</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CWD-196</td>
<td>Improve the ability to configure the internal cache's used by the Crowd client and the Crowd console</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CWD-195</td>
<td>Implement SSO for Jive Forums</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CWD-193</td>
<td>Download archive is missing wsdl4j-1.5.2.jar</td>
<td></td>
<td>Closed</td>
</tr>
</tbody>
</table>

Cheers,
The Atlassian Crowd Development Team
Crowd 1.0.3 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

<table>
<thead>
<tr>
<th>Key</th>
<th>Summary</th>
<th>Pr</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>CWD-163</td>
<td>Administration Console allows login of unauthorized users</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CWD-218</td>
<td>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</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CWD-216</td>
<td>Crowd session token should be unique for each user, directory, machine</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CWD-214</td>
<td>Login should logout any previous logged in users before a new login</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CWD-179</td>
<td>Paged results control option for LDAP connectors.</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CWD-177</td>
<td>Fisheye connector logs unnecessary exception.</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CWD-175</td>
<td>Computers show up in the Principal list within Crowd from MSAD</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CWD-169</td>
<td>NullPointerException on add OpenLDAP directory</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CWD-121</td>
<td>Setting a &quot;Remember Me&quot; flag in Confluence, JIRA or Bamboo does not work, since the Token Reaper 'reaps' all session when the timeout is reached</td>
<td></td>
<td>Closed</td>
</tr>
</tbody>
</table>

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 1.2 has now been released — see the Crowd 1.2 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

<table>
<thead>
<tr>
<th>Key</th>
<th>Summary</th>
<th>Pr</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>CWD-221</td>
<td>Add documentation (marketing) section for Apache Directory Server</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CWD-220</td>
<td>Implement RemoteDirectory updatePrincipal(RemotePrincipal) method for LDAP servers using InetOrgPerson as the Principal object.</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>CWD-225</td>
<td>Import and export of Crowd fails when the database is Oracle</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CWD-213</td>
<td>The Sitemesh and Webwork cleanup filters are being wrapped around the XFire requests.</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>CWD-206</td>
<td>JIRA User Import Doesn't Set Groups on Principals</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>CWD-172</td>
<td>Remove this error: SEVERE: No Store configured, persistence disabled</td>
<td></td>
<td>Resolved</td>
</tr>
</tbody>
</table>

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 1.2 has now been released — see the Crowd 1.2 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

<table>
<thead>
<tr>
<th>Key</th>
<th>Summary</th>
<th>Pr</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>CWD-252</td>
<td>Active Directory filter does not exclude accounts which are no sAMAccountName type.</td>
<td>📌</td>
<td>Closed</td>
</tr>
<tr>
<td>CWD-244</td>
<td>Set compile flags with maven build scripts to be vs. 1.4</td>
<td>📌</td>
<td>Resolved</td>
</tr>
<tr>
<td>CWD-259</td>
<td>Username is not displayed in Confluence (2.4.X) when first logging in.</td>
<td>📌</td>
<td>Closed</td>
</tr>
<tr>
<td>CWD-258</td>
<td>Domain for multihost single sign-on is not setting the cookie correctly.</td>
<td>📌</td>
<td>Closed</td>
</tr>
<tr>
<td>CWD-257</td>
<td>VerifyTokenFilter missing from the Demo application.</td>
<td>📌</td>
<td>Closed</td>
</tr>
<tr>
<td>CWD-256</td>
<td>Importer success screens display success even on an exception.</td>
<td>📌</td>
<td>Resolved</td>
</tr>
<tr>
<td>CWD-254</td>
<td>Review Installation documentation</td>
<td>📌</td>
<td>Resolved</td>
</tr>
<tr>
<td>CWD-248</td>
<td>CLONE - The Sitemesh and Webwork cleanup filters are being wrapped around the XFire requests.</td>
<td>📌</td>
<td>Closed</td>
</tr>
<tr>
<td>CWD-243</td>
<td>Document how you can not delete the Crowd console.</td>
<td>📌</td>
<td>Resolved</td>
</tr>
<tr>
<td>CWD-242</td>
<td>You can delete the integrated Crowd application</td>
<td>📌</td>
<td>Resolved</td>
</tr>
<tr>
<td>CWD-235</td>
<td>System error when no directory is selected when adding a group</td>
<td>📌</td>
<td>Resolved</td>
</tr>
<tr>
<td>CWD-234</td>
<td>Add WebSphere installation notes for Crowd.</td>
<td>📌</td>
<td>Resolved</td>
</tr>
<tr>
<td>CWD-229</td>
<td>Transactions wrapping transactions. The transaction manager is not aware about the wrapping transaction.</td>
<td>📌</td>
<td>Resolved</td>
</tr>
<tr>
<td>CWD-222</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Issue</td>
<td>Description</td>
<td>Resolution</td>
<td></td>
</tr>
<tr>
<td>---------</td>
<td>-----------------------------------------------------------------------------</td>
<td>------------</td>
<td></td>
</tr>
<tr>
<td>CWD-226</td>
<td>Crowd is not handling latin1 characters correctly</td>
<td>Resolved</td>
<td></td>
</tr>
<tr>
<td></td>
<td>browser window title should say 'View Application'</td>
<td>Resolved</td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Key</th>
<th>Summary</th>
<th>Pr</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>CWD-265</td>
<td>Confluence displays the users fullname instead of email when integrated with Crowd</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>CWD-263</td>
<td>Fails with exception on Search</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>CWD-262</td>
<td>Improve the management of the Crowd domain during setup and in the Console.</td>
<td></td>
<td>Resolved</td>
</tr>
</tbody>
</table>

Cheers,
The Atlassian Crowd Development Team
Crowd 1.0.7 Release Notes

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:

<table>
<thead>
<tr>
<th>Key</th>
<th>Summary</th>
<th>Pr</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>CWD-296</td>
<td>LDAP update password implementation.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CWD-316</td>
<td>Active Directory principals can sign in with a blank password</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CWD-181</td>
<td>Continually asked to re-auth with Apache</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CWD-287</td>
<td>Reset password option for the Console</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CWD-233</td>
<td>javadoc SecurityServer</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Resolved

Cheers,
The Atlassian Crowd Development Team
Crowd 1.1.0 Release Notes

The Atlassian Crowd team is proud to announce the release of Crowd 1.1.

This release contains a whole host of new features targeted at implementing OpenID, along with core updates to the Crowd Administration Console.

OpenID-enable your organisation with CrowdID

OpenID enables you to use a centralised identity to login to any website that supports OpenID. It opens up the possibilities of massive scale cross-domain SSO.

Think about all the accounts you have online: blogs, wikis, to-do lists, photo galleries. The list is endless. Even simple tasks such as leaving comments on someone else's blog may require you to register an account with that particular blogging system. This leaves you, as an end user, to set up and manage numerous accounts on each of these sites. With OpenID, rather than managing all these disparate accounts individually, users can manage their identity in one place via an authentication server.

With the ever-increasing adoption of this open authentication framework, including names such as Microsoft, AOL, Sun, Verisign and Firefox, expect to see many applications enabled for OpenID authentication.

CrowdID offers OpenID to an organisation's user base, allowing users to manage their online identity. Everything from configuring different profiles, managing trusted sites to reviewing authentication activity, is accessible from CrowdID. Administrators can set up whitelists/blacklists so that only trusted hosts can request authentication and can set up secure communication via SSL. All of the users can be managed via Crowd's security server, utilizing LDAP services from products such as Microsoft Active Directory.

Included with CrowdID is a sample OpenID client application, providing a working example of an OpenID enabled application. This will help developers kick start OpenID-enabling their applications.

Using OpenID

Rather than registering and typing in your username and password on each site that you visit, OpenID allows you to type a URL similar to 'openid.mycompany.com/users/jstepka':

My OpenID

https://openid.atlassian.com/users/jstepka

Use this URL to log in to websites that support OpenID.

The OpenID website that you are logging in to will redirect you to CrowdID, which will ask you if you would like to allow authentication with the requesting site.

You can even choose to 'Always' allow authentication with particular OpenID sites, which allows pass-through authentication if you are already logged into your CrowdID server. If you do this, then when you visit the site later, simply provide your URL (e.g. 'openid.mycompany.com/users/jstepka') and you are in.

Think of it as 'Remember Me' for the whole internet!
'Blacklist' and 'Whitelist'

'Blacklists' and 'whitelists' allow administrators to lock down CrowdID their server so that, if necessary, it can only communicate with trusted hosts with which you have established relationships.

A blacklist will prevent specific hosts from communicating with the OpenID server. A whitelist will allow only specific hosts to communicate with the OpenID server.

OpenID Advanced Options

Some external sites implement security better than others. With CrowdID, you can pick how tough you want to be on OpenID sites that communicate with your Crowd OpenID server.
Crowd Console and Server Updates

Choose Your Encryption Type

Every administrator has their own password policies. When using a Crowd Internal Directory you can now select the level of encryption you need.

![Encryption Options]

Import Your JIRA and Confluence Passwords

Migration can be a pain. To ease your switch from existing Atlassian products, Crowd can now import your existing passwords!

![Password Import]

Faster Web-Services

Crowd web-services now support GZip compression, improving the performance when downloading large amounts of data such as the all the members of a large group or when performing large search.

Improved Apache and Subversion Integration

The Apache and Subversion library performance has been improved with the implementation of client-side caching of approved authentication requests.

Jive Forums 5.5 Support

The Jive Forums centralised authentication connector has been updated to support the new 5.5 major release of Jive Forums.

LDAP Configuration Tester

When setting up a Crowd LDAP connection you can now verify that your configuration connects as expected.
### JIRA Issue Tracker

<table>
<thead>
<tr>
<th>Key</th>
<th>Summary</th>
<th>Pr</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>CWD-376</td>
<td>Export fails when an application does not have a description.</td>
<td><img src="https://example.com" alt=" " /></td>
<td><img src="https://example.com" alt=" " /> Resolved</td>
</tr>
<tr>
<td>CWD-359</td>
<td>'Blacklist' and 'Whitelist' options display intermittently in IE</td>
<td><img src="https://example.com" alt=" " /></td>
<td><img src="https://example.com" alt=" " /> Resolved</td>
</tr>
<tr>
<td>CWD-271</td>
<td>Login and Logoff for OpenID Server application.</td>
<td><img src="https://example.com" alt=" " /></td>
<td><img src="https://example.com" alt=" " /> Closed</td>
</tr>
<tr>
<td>CWD-245</td>
<td>Jive Forums 5.5 Support</td>
<td><img src="https://example.com" alt=" " /></td>
<td><img src="https://example.com" alt=" " /> Resolved</td>
</tr>
<tr>
<td>CWD-379</td>
<td>Change Password link on openid.atlassian.com throws 'No Action' error page</td>
<td><img src="https://example.com" alt=" " /></td>
<td><img src="https://example.com" alt=" " /> Resolved</td>
</tr>
<tr>
<td>CWD-377</td>
<td>Updating an Application will update the password for an application, even when you do not type in a new password</td>
<td><img src="https://example.com" alt=" " /></td>
<td><img src="https://example.com" alt=" " /> Closed</td>
</tr>
<tr>
<td>CWD-360</td>
<td>ORA-01000: maximum open cursors exceeded</td>
<td><img src="https://example.com" alt=" " /></td>
<td><img src="https://example.com" alt=" " /> Resolved</td>
</tr>
<tr>
<td>CWD-354</td>
<td>suggestions for the OpenID login page</td>
<td><img src="https://example.com" alt=" " /></td>
<td><img src="https://example.com" alt=" " /> Resolved</td>
</tr>
<tr>
<td>CWD-351</td>
<td>When logging out of Bamboo and anonymous mode is turned off, users still have the ability to create plans etc.</td>
<td><img src="https://example.com" alt=" " /></td>
<td><img src="https://example.com" alt=" " /> Resolved</td>
</tr>
<tr>
<td>CWD-343</td>
<td>Atlassian-user integration - get display name attribute from attributes if there rather than building display name adhoc.</td>
<td><img src="https://example.com" alt=" " /></td>
<td><img src="https://example.com" alt=" " /> Resolved</td>
</tr>
</tbody>
</table>
CWD-332 Test configuration buttons when creating an LDAP directory connector. Resolved

CWD-323 Test connection utility for LDAP servers. Resolved

CWD-320 Improve the importing of users from Confluence and JIRA so these users do not need to reset their passwords Resolved

CWD-319 The export function of Crowd needs to have a flag to say don't export domain. Resolved

CWD-318 ApacheDS crowd integration does not currently support the adding of groups Resolved

CWD-313 The Apache module needs some kind of cache implemented similar to our other 'clients', to help improve performance around apache integration Resolved

CWD-305 Add optional GZIP compression support for XFire SOAP services and client. Resolved

CWD-304 Auto configure openid server as part of the setup process. Resolved

CWD-302 Skin the OpenID Server Closed

CWD-301 OpenID Client - Dummy Mode Resolved

CWD-300 OpenID Server - dummy mode Resolved

CWD-299 OpenID Client - Check Immediate Resolved

CWD-298 OpenID Server - Check Immediate Resolved

CWD-294 Test OpenIDClient Form Redirection Resolved

CWD-292 OpenID Server Implementation Resolved

CWD-291 Auto configure openid server as part of the setup process. Closed

CWD-290 Upgrade webwork from 2.2.4 to 2.2.5 Resolved

CWD-288 Change application titles - not footers Resolved

CWD-286 Skin Demo RP application Resolved

CWD-285 Display attributes in the demo application upon login (store in session for display) Resolved

CWD-284 Login and Logoff for OpenID demo relying party application. Resolved

CWD-283 Configure request attributes for demo app Resolved

CWD-280 Document OpenID server configuration Closed
<table>
<thead>
<tr>
<th>CWD-279</th>
<th>Attribute/Profile Management</th>
<th></th>
<th></th>
<th>Resolved</th>
</tr>
</thead>
<tbody>
<tr>
<td>CWD-278</td>
<td>Authentication redirect from relying party.</td>
<td></td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>CWD-277</td>
<td>Skin Server</td>
<td></td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>CWD-276</td>
<td>Profile authentication history</td>
<td></td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>CWD-275</td>
<td>Enable/disable localhost relying parties.</td>
<td></td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>CWD-274</td>
<td>Whitelist and Blacklist Editor</td>
<td></td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>CWD-273</td>
<td>Force Association</td>
<td></td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>CWD-272</td>
<td>Reset password option.</td>
<td></td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>CWD-269</td>
<td>document the management of the Crowd domain during setup and in the Console</td>
<td></td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>CWD-246</td>
<td>Update documentation with new information about installing connector for 5.5.X version of JIVE, add 'SecurityServerClient'</td>
<td></td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>CWD-232</td>
<td>add 'SecurityServerClient'</td>
<td></td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>CWD-154</td>
<td>Apache DS connector</td>
<td></td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>CWD-144</td>
<td>Add 'green' success message to 'update' actions on Console.</td>
<td></td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>CWD-65</td>
<td>Explore OpenID support</td>
<td></td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CWD-368</td>
<td>Stray backslash on Groups administration screen</td>
<td></td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>CWD-365</td>
<td>Typo in hint for Password Encryption during initial directory setup</td>
<td></td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>CWD-325</td>
<td>Directory details tab shows empty pink error box</td>
<td></td>
<td></td>
<td>Resolved</td>
</tr>
</tbody>
</table>

Cheers,

The Atlassian Crowd Development Team
Crowd 1.1.1 Release Notes

This page last changed on Jul 26, 2007 by smaddox.
The Crowd development team has released Crowd 1.1.1.

This release is a highly recommended upgrade from Crowd 1.1.0 since it provides a security fix to XWork, the technology underlying the web framework WebWork which is used by Crowd.

This release also contains a new CSV importer as well as fixes for some issues found in 1.1.0.

Importing Users and Groups from a CSV File

You can now copy users from an external directory or user base into Crowd via a CSV (comma-separated values) file.

The new CSV Importer allows you to specify a file containing user information, and optionally another file containing the groups to which the users belong. You can then map the CSV fields to the Crowd directory fields. After performing the import, Crowd sums up the results. Screenshot: 'CSV Importer - Configuration'.

Other Fixes in Crowd 1.1.1

<table>
<thead>
<tr>
<th>Key</th>
<th>Summary</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>CWD-445</td>
<td>Internal Directory search for Group by name is failing to aggregate the correct members</td>
<td>Closed</td>
</tr>
<tr>
<td>CWD-418</td>
<td>Chained directories are returning multiple groups/roles rather than aggregating group names</td>
<td>Resolved</td>
</tr>
<tr>
<td>CWD-388</td>
<td>Paging principal sessions links are incorrect and do not function</td>
<td>Resolved</td>
</tr>
<tr>
<td>CWD-438</td>
<td>Users shown twice in JIRA</td>
<td>Closed</td>
</tr>
<tr>
<td>CWD-437</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CWD-435</td>
<td>Exception using Seraph single-sign-on in Bamboo</td>
<td>Resolved</td>
</tr>
<tr>
<td>---------</td>
<td>-------------------------------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>CWD-434</td>
<td>Searching for a group spanning multiple directories by its name will not amalgamate the principals</td>
<td>Resolved</td>
</tr>
<tr>
<td>CWD-425</td>
<td>Trim the application address when adding a valid application remote address.</td>
<td>Resolved</td>
</tr>
<tr>
<td>CWD-419</td>
<td>displayName attribute is not used with the JIRA connector</td>
<td>Resolved</td>
</tr>
<tr>
<td>CWD-414</td>
<td>The CSV Importer needs to display user results for duplicate entries i.e. users that have been ignored since they already exist in Crowd.</td>
<td>Resolved</td>
</tr>
<tr>
<td>CWD-400</td>
<td>JIRA attach screenshot does not write file to the filesystem when Crowdified.</td>
<td>Resolved</td>
</tr>
<tr>
<td>CWD-397</td>
<td>Document the CSV importer</td>
<td>Resolved</td>
</tr>
<tr>
<td>CWD-385</td>
<td>Generated tokens have non-HTML escaped characters.</td>
<td>Resolved</td>
</tr>
<tr>
<td>CWD-382</td>
<td>Create custom add successful page does not display directort connector page.</td>
<td>Resolved</td>
</tr>
<tr>
<td>CWD-352</td>
<td>Configure the number of paged results for an LDAP connector</td>
<td>Resolved</td>
</tr>
<tr>
<td>CWD-290</td>
<td>Upgrade webwork from 2.2.4 to 2.2.5</td>
<td>Resolved</td>
</tr>
<tr>
<td>CWD-53</td>
<td>CSV importer</td>
<td>Resolved</td>
</tr>
<tr>
<td>CWD-428</td>
<td>Change wording on the Atlassian importer</td>
<td>Resolved</td>
</tr>
<tr>
<td>CWD-407</td>
<td>Textual changes to new CSV-importer screens</td>
<td>Resolved</td>
</tr>
</tbody>
</table>

Cheers,
The Atlassian Crowd Development Team
Crowd 1.1.2 Release Notes

This page last changed on Sep 03, 2007 by justen.stepka@atlassian.com.

The Crowd development team has released Crowd 1.1.2.

Crowd 1.1.2 is a recommended upgrade from Crowd 1.1.1 since it provides improved integration with JIRA and Confluence, and tidier functionality for multiple directories.

For cross product compatibility, you must upgrade to the following versions of each product:

- Crowd 1.1.2 or later
- Bamboo 1.2.2 or later
- Confluence 2.5.6 or later
- JIRA 3.7.4 or later

Integration with JIRA user management

With Crowd 1.1.2, you can now turn external user management off in JIRA. This means that you can allow signup via JIRA, and you can manage your users within JIRA. Changes will flow through to Crowd.

JIRA has an automatic group membership feature. This means that any new user added through JIRA will automatically be a member of all groups which have the JIRA Users permission. In this way, you can ensure that a new user is automatically added to several groups when they sign up with JIRA.

RSS feeds

Crowd 1.1.2 fixes the problem experienced accessing RSS feeds from JIRA including retrieving JIRA issues via Confluence macros (e.g. the JIRA portlet macro).

Improved LDAP Performance

Crowd 1.1.2 now queries LDAP repositories in a more efficient manner that will give a dramatic performance increase for those with large numbers of LDAP groups.

Other Fixes in Crowd 1.1.1

<table>
<thead>
<tr>
<th>Key</th>
<th>Summary</th>
<th>Pr</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>CWD-314</td>
<td>Not able to Retrieve Issues (RSS) if JIRA is Integrated with Crowd</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>CWD-297</td>
<td>JIRA performance improvements</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>CWD-472</td>
<td>OpenID not working with LiveJournal</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>CWD-540</td>
<td>CrowdID Install</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>CWD-503</td>
<td>Cannot modify user profile when using Crowd authentication, fails with NullPointer.onRemotePrincipal.getEmail()</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>CWD-497</td>
<td>Crowd integration of Extranet JIRA has authentication problems</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>CWD-496</td>
<td>requiresPasswordChange gets reset to false during login for an InternalDirectory</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>Issue Number</td>
<td>Description</td>
<td>Status</td>
<td></td>
</tr>
<tr>
<td>--------------</td>
<td>-------------</td>
<td>--------</td>
<td></td>
</tr>
<tr>
<td>CWD-495</td>
<td>Principals are being added with whitespace in their usernames</td>
<td>Resolved</td>
<td></td>
</tr>
<tr>
<td>CWD-492</td>
<td>Concurrent modification exception in JIRAAuthenticator logout code</td>
<td>Resolved</td>
<td></td>
</tr>
<tr>
<td>CWD-487</td>
<td>The upgrade manager should run after setup is complete</td>
<td>Resolved</td>
<td></td>
</tr>
<tr>
<td>CWD-484</td>
<td>When Confluence 2.6 releases we need to move the code from the bamboo-integration module back into the atlassian-user module.</td>
<td>Resolved</td>
<td></td>
</tr>
<tr>
<td>CWD-478</td>
<td>Update Confluence Integration Doc</td>
<td>Resolved</td>
<td></td>
</tr>
<tr>
<td>CWD-462</td>
<td>Implement add user method of OSUser for JIRA</td>
<td>Resolved</td>
<td></td>
</tr>
<tr>
<td>CWD-452</td>
<td>JIRA user management should allow admins to update Crowd users</td>
<td>Resolved</td>
<td></td>
</tr>
<tr>
<td>CWD-448</td>
<td>Remote application's calls to removePrincipal(name) only removes the first principal it finds</td>
<td>Resolved</td>
<td></td>
</tr>
<tr>
<td>CWD-447</td>
<td>Remote application's calls to removeRole(name) only removes the first role it finds</td>
<td>Resolved</td>
<td></td>
</tr>
<tr>
<td>CWD-446</td>
<td>Remote application's calls to removeGroup(name) only removes the first group it finds</td>
<td>Resolved</td>
<td></td>
</tr>
<tr>
<td>CWD-421</td>
<td>Client JARs in client/lib are incomplete</td>
<td>Resolved</td>
<td></td>
</tr>
<tr>
<td>CWD-394</td>
<td>Full Name Search always returns all users</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>CWD-132</td>
<td>Windows service registration feature.</td>
<td>Resolved</td>
<td></td>
</tr>
<tr>
<td>CWD-390</td>
<td>Browser cookies cause NullPointerException when integrated with Confluence</td>
<td>Resolved</td>
<td></td>
</tr>
</tbody>
</table>

Cheers,
The Atlassian Crowd Development Team
The Atlassian Crowd team is delighted to present Crowd 1.2.

Crowd 1.2 is a major release that focuses on enhanced integration, security and usability. Crowd's directory permissions now allow finer-grained control, so that you can define the permissions per application. The Group and Role Browsers now display group/role membership. We have enhanced group management in the existing Jive Forums and Apache/Subversion connectors. Our NTLM plugin offers SSO (single sign-on) for JIRA and Confluence via NTLM desktop authentication. A new connector lets you integrate your Acegi security solution with Crowd. And you can import your Bamboo users directly into a Crowd directory.

We'd like to say a special thank you to CustomWare for their assistance with deployment and testing of the NTLM plugin.

### Highlights of this release:
- Directory Permissions per Application
- Group and Role Membership Browser
- Improved Browser for OpenID Login History
- NTLM Support
- Improved Integration with Jive Forums
- Acegi Application Connector
- Group-Based Authorisation Added for Subversion
- New Importer for Bamboo Users
- Plus Over 70 Improvements and Bug-Fixes

### Responding to your feedback:
- 🌟 8 new feature requests implemented
- 🌟 68 votes satisfied

Your votes and issues help us keep improving our products, and are much appreciated.

---

### Upgrading to Crowd 1.2

You can download Crowd from the Atlassian website. If upgrading from a previous version, please read the Upgrade Notes.

---

### Highlights of Crowd 1.2

#### Directory Permissions per Application

- Directory permissions determine whether groups, principals and roles can be added, modified or deleted.
- Before this release, permissions were set at directory level only. Permissions therefore applied across all applications associated with the directory.
- With Crowd 1.2, directory permissions can be set for each application. For example, you could enable the 'Add Principal' permission on the 'Employees' directory for JIRA but disable the permission for Confluence.
- See the screenshot below, and take a look at an example.
Group and Role Membership Browser

- A new 'Principals' tab in the Group Browser shows all principals belonging to a group.
- You can view membership in the Role Browser too.
- Read the documentation.

Improved Browser for OpenID Login History

- Instead of showing all login history on a single page, the Login History screen now divides the history into pages, for easier viewing.
- To move between pages, click 'Next', 'Prev' or a specific page number.
- In the 'Action' column, a new item '(Auto) Allow Always' tells you which logins were allowed automatically because of a previous 'Allow Always' instruction.

NTLM Support

- **NTLM** is a Microsoft authentication protocol that allows you to access a website using your desktop login. The protocol utilises an integration between Microsoft Internet Explorer and Active Directory. When using this feature, users will only need to log in to their desktop to access NTLM-integrated applications.
<table>
<thead>
<tr>
<th>5</th>
<th>Improved Integration with Jive Forums</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• Crowd 1.2 provides support for group management in Jive Forums.</td>
</tr>
<tr>
<td></td>
<td>• Groups and group memberships are now pulled from Crowd.</td>
</tr>
<tr>
<td></td>
<td>• You can use the Jive Forums admin console to define application permissions associated with groups.</td>
</tr>
<tr>
<td></td>
<td>• This allows Crowd to manage Jive Forums groups and memberships and Jive Forums to handle the permissions associated with the groups.</td>
</tr>
<tr>
<td></td>
<td>• Read the documentation.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>6</th>
<th>Acegi Application Connector</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• Crowd 1.2 provides a built-in application connector for Acegi, a security solution with a particular emphasis on Spring Java/JEE applications.</td>
</tr>
<tr>
<td></td>
<td>• Read the documentation.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>7</th>
<th>Group-Based Authorisation Added for Subversion</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• Crowd allows you to password-protect your SVN repository running under Apache.</td>
</tr>
<tr>
<td></td>
<td>• You can now also configure fine-grained access by group as well as by user.</td>
</tr>
<tr>
<td></td>
<td>• Read more about the Crowd Subversion connector.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>8</th>
<th>New Importer for Bamboo Users</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• Our new Bamboo importer allows you to copy your Bamboo users into a Crowd directory.</td>
</tr>
<tr>
<td></td>
<td>• Read the documentation.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>9</th>
<th>Plus Over 70 Improvements and Bug-Fixes</th>
</tr>
</thead>
</table>

• JIRA and Confluence NTLM connectors are now supported with Crowd 1.2.
• Read the instructions on setting up Confluence and JIRA NTLM support in Crowd.
<table>
<thead>
<tr>
<th>Key</th>
<th>Summary</th>
<th>Priority</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>CWD-580</td>
<td>Events, EventType and Event Listeners are not being exported as part of the XML backup</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>CWD-526</td>
<td>Editing groups in Crowd has no effect in Bamboo</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>CWD-398</td>
<td>;jsessionid added to all Crowd links</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>CWD-314</td>
<td>Not able to Retrieve Issues (RSS) if JIRA is Integrated with Crowd</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>CWD-297</td>
<td>JIRA performance improvements</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>CWD-249</td>
<td>Adjust build process to publish maven2 client poms.</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>CWD-534</td>
<td>Upgrade Crowd to Spring Framework 2.0.6 from 1.2.x</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>CWD-388</td>
<td>Paging principal sessions links are incorrect and do not function.</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>CWD-336</td>
<td>No date sent in email headers for messages sent by Crowd</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CWD-625</td>
<td>If an allowed Principal Attribute is null it is not possible to</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>Issue</td>
<td>Description</td>
<td></td>
<td></td>
</tr>
<tr>
<td>---------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CWD-599</td>
<td>When creating and viewing an LDAP connector, we have been displaying the password as clear text, this should at least be a password field</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CWD-597</td>
<td>License user-limit check event should not execute for unlimited licenses</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CWD-593</td>
<td>Upgrade to Atlassian-Extras 1.9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CWD-588</td>
<td>Jive Forums remote authentication is not working</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CWD-582</td>
<td>If the two core event listeners do not exist add them via an upgrade task.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CWD-579</td>
<td>Role Tab shows the correct number of roles however they all show up as the principal name</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CWD-578</td>
<td>Allow a crowd administrator to recalculate the user total for a Crowd install</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CWD-577</td>
<td>Remove Group link on View Principal does not</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CWD-576</td>
<td>Document Crowd installation on JBoss contain a valid directory ID</td>
<td>Resolved</td>
<td></td>
</tr>
<tr>
<td>---------</td>
<td>-------------------------------------------------</td>
<td>----------</td>
<td></td>
</tr>
<tr>
<td>CWD-575</td>
<td>Document the 'config test' tab</td>
<td>Resolved</td>
<td></td>
</tr>
<tr>
<td>CWD-573</td>
<td>Multiple cookies are wrote back to the browser during an authentication.</td>
<td>Resolved</td>
<td></td>
</tr>
<tr>
<td>CWD-567</td>
<td>HSQL context path storage issues when not using start_crowd.bat/sh</td>
<td>Resolved</td>
<td></td>
</tr>
<tr>
<td>CWD-556</td>
<td>Atlassian applications hang and can not start when integrated with Crowd under the same VM.</td>
<td>Resolved</td>
<td></td>
</tr>
<tr>
<td>CWD-552</td>
<td>Data imports fail when no application-group associations are in place.</td>
<td>Resolved</td>
<td></td>
</tr>
<tr>
<td>CWD-540</td>
<td>CrowdID Install Documentation Mistake Need and EAR/WAR download to use other application servers</td>
<td>Resolved</td>
<td></td>
</tr>
<tr>
<td>CWD-539</td>
<td>Full Name attribute (displayName/firstName +surname) used</td>
<td>Resolved</td>
<td></td>
</tr>
<tr>
<td>CWD-537</td>
<td>Method to create a token for a principal without performing an authentication.</td>
<td>Resolved</td>
<td></td>
</tr>
<tr>
<td>CWD-524</td>
<td>CrowdID Install Documentation Mistake Need and EAR/WAR download to use other application servers</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>Issue</td>
<td>Description</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CWD-517</td>
<td>Documentation update for 'Upgrading Crowd' as per customer's comment</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CWD-516</td>
<td>JIRA breaks with retrieveUserMetaProperties NPE after adding user in Crowd</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CWD-514</td>
<td>Move Crowd to use Webwork 2.2.6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CWD-513</td>
<td>Move Crowd to use Seraph 0.9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CWD-508</td>
<td>Release Crowd EAR/WAR edition</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CWD-504</td>
<td>Crowd should be offered as a EAR/WAR package in addition to standalone</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CWD-503</td>
<td>Cannot modify user profile when using Crowd authentication, fails with NullPointerException on RemotePrincipal.getEmail()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CWD-502</td>
<td>Unauthenticated user causes session nuking in Crowdified JIRA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CWD-501</td>
<td>OpenID history browser Directory CRUD permissions on an Application-by-Appliction basis.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CWD-497</td>
<td>Crowd integration of Extranet</td>
<td></td>
<td></td>
</tr>
<tr>
<td>JIRA Issue</td>
<td>Description</td>
<td></td>
<td></td>
</tr>
<tr>
<td>------------</td>
<td>-------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CWD-496</td>
<td>JIRA has authentication problems. requiresPasswordChange gets reset to false during login for an InternalDirectory.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CWD-495</td>
<td>Principals are being added with whitespace in their usernames.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CWD-492</td>
<td>Concurrent modification exception in JIRAAuthenticator logout code.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CWD-489</td>
<td>Change the Crowd Upgrade Guide to only copy the password from the crowd.properties files, not copy the entire file.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CWD-488</td>
<td>The build.properties file and Ant associated ant task should not overwrite the password attribute in the crowd.properties file.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CWD-487</td>
<td>The upgrade manager should run after setup is complete.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| CWD-484    | When Confluence 2.6 releases we need to move the code from the bamboo-integration module back into the atlassian-
<p>| CWD-465 | Improve the current Jive integration to provide support for Group management | Resolved |
| CWD-464 | Email address validation is not RFC-2822 compliant | Closed |
| CWD-462 | Implement add user method of OSUser for JIRA | Resolved |
| CWD-459 | Update the SecurityServer SOAP API to enable editing/updating groups | Resolved |
| CWD-452 | JIRA user management should allow admins to update Crowd users | Resolved |
| CWD-435 | Exception using Seraph single-sign-on in Bamboo | Resolved |
| CWD-430 | CrowdID Not Signing User Attributes Like Nickname or Email | Resolved |
| CWD-425 | Trim the application address when adding a valid application remote address. | Resolved |
| CWD-421 | Client JARs in client/lib are incomplete | Resolved |
| CWD-419 | displayName attribute is not used with | Resolved |
| CWD-417   | the JIRA connector Libraries in client directory are not enough | Resolved |
| CWD-414   | The CSV Importer needs to display user results for duplicate entries i.e. users that have been ignored since they already exist in Crowd. | Resolved |
| CWD-392   | No group integration into Subversion | Resolved |
| CWD-373   | Improve the build process for source releases | Resolved |
| CWD-349   | Create a Bamboo to Crowd Principal and Group importer. | Resolved |
| CWD-348   | When switching from internat authentication to Crowd authentication (using seraph?), exception is throw during login. | Resolved |
| CWD-185   | The import/export is confined to a given instance, we need to make it so the XML file can be used on any Crowd deployment. | Resolved |
| CWD-135   | Support NTLM | Closed |
| CWD-19    | Acegi Connector | Resolved |</p>
<table>
<thead>
<tr>
<th>Ticket</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CWD-618</td>
<td>View Principal is throwing a RemoteException when trying to view the Roles of a Principal</td>
</tr>
<tr>
<td>CWD-617</td>
<td>Browse Principal is not showing an email address for the principal returned.</td>
</tr>
<tr>
<td>CWD-525</td>
<td>Login to Jira with an existing cookie (non-crowd) shows a NullPointerException.</td>
</tr>
<tr>
<td>CWD-442</td>
<td>View members of the group or role.</td>
</tr>
<tr>
<td>CWD-428</td>
<td>Change wording on the Atlassian importer.</td>
</tr>
<tr>
<td>CWD-407</td>
<td>Textual changes to new CSV-importer screens.</td>
</tr>
<tr>
<td>CWD-390</td>
<td>Browser cookies cause NullPointerException when integrated with Confluence.</td>
</tr>
</tbody>
</table>
Installing Crowd

You can download Crowd here.

- **1. System Requirements**
  - 1.1 Setting JAVA_HOME
- **2. Installing Crowd and CrowdID**
  - 2.1 Connecting Crowd to a Database
    - 2.1.1 HSQL DB
    - 2.1.2 MS SQL Server
    - 2.1.3 MySQL
    - 2.1.4 Oracle
    - 2.1.5 PostgreSQL
  - 2.2 Connecting CrowdID to a Database
    - 2.2.1 HSQL DB
    - 2.2.2 MS SQL Server
    - 2.2.3 MySQL
    - 2.2.4 Oracle
    - 2.2.5 PostgreSQL
- **2.3 Installing Crowd and CrowdID WAR Distribution**
  - 2.3.1 Installing Crowd WAR on JBoss
- **3. Running the Setup Wizard**
- **4. Configuring Crowd**
  - 4.1 Important Files
    - The crowd.properties File
  - 4.2 Changing the Port that Crowd uses
- **5. Installing Crowd as a Windows Service**
  - 5.1 Specifying Startup Order of Windows Services
  - 5.2 Changing the User for the Crowd Windows Service
  - 5.3 Removing the Crowd Windows Service
  - 5.4 Troubleshooting Crowd as a Windows Service

Related Topics

- Crowd Release Notes
- Installing Crowd
- Upgrading Crowd
1. System Requirements

This page last changed on Nov 12, 2007 by smaddox.

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 applications and users 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 applications and users in your Crowd installation.

Software Requirements

1. Sun JDK 1.4 (1.5 or higher is preferred). NOTE: Once the JDK is installed, you will need to set the JAVA_HOME environment variable, pointing to the root directory of the JDK. Some JDK installers set this automatically (check by typing ‘echo %JAVA_HOME%’ in a DOS prompt, or ‘echo $JAVA_HOME’ in a shell). If it is not set, please see 1.1 Setting JAVA_HOME.

2. J2EE 1.4 application server or a Servlet 2.3 web container. NOTE: Crowd ships with Apache Tomcat (5.5.x).

3. JDBC-compliant database that is supported by Hibernate. NOTE: Crowd ships with a built-in HSQL database, which is fine for evaluation purposes. For production environments we recommend configuring Crowd to use an external database.

Supported Databases

The following database servers are supported by Hibernate:

- HypersonicSQL
- PostgreSQL
- Microsoft SQL Server
- MySQL
- Oracle 10g (tested on 10.2.0.1)

Of these, the following databases have been tested and are supported by Atlassian:

- 2.1.1 HSQL DB
- 2.1.2 MS SQL Server
- 2.1.3 MySQL
- 2.1.4 Oracle
- 2.1.5 PostgreSQL

Supported J2EE Servers

The following J2EE servers are supported:

- JBoss (4.2.2 GA)
- Resin (3.0.x) - tested on 3.0.23
- Tomcat (5.5.x) - tested on 5.5.20

Next Step

2. Installing Crowd and CrowdID
Related Topics

• 1. System Requirements
  • 1.1 Setting JAVA_HOME
• 2. Installing Crowd and CrowdID
  • 2.1 Connecting Crowd to a Database
    • 2.1.1 HSQL DB
    • 2.1.2 MS SQL Server
    • 2.1.3 MySQL
    • 2.1.4 Oracle
    • 2.1.5 PostgreSQL
  • 2.2 Connecting CrowdID to a Database
    • 2.2.1 HSQL DB
    • 2.2.2 MS SQL Server
    • 2.2.3 MySQL
    • 2.2.4 Oracle
    • 2.2.5 PostgreSQL
• 2.3 Installing Crowd and CrowdID WAR Distribution
  • 2.3.1 Installing Crowd WAR on JBoss
• 3. Running the Setup Wizard
• 4. Configuring Crowd
  • 4.1 Important Files
    • The crowd.properties File
  • 4.2 Changing the Port that Crowd uses
• 5. Installing Crowd as a Windows Service
  • 5.1 Specifying Startup Order of Windows Services
  • 5.2 Changing the User for the Crowd Windows Service
  • 5.3 Removing the Crowd Windows Service
  • 5.4 Troubleshooting Crowd as a Windows Service
1.1 Setting JAVA_HOME

Once you have installed the JDK (see 1. System Requirements), you need to set the JAVA_HOME environment variable.

To set the JAVA_HOME environment variable on Windows

1. Right click on the 'My Computer' icon on your desktop and select 'Properties'.
2. Click the 'Advanced' tab.
3. Click the 'Environment Variables' button.
4. Click 'New'.
5. In the 'Variable name' field, enter 'JAVA_HOME'.
6. In the 'Variable value' field, enter the directory (including its full path) where you installed the JDK.
7. Restart the computer.

To set the JAVA_HOME environment variable on 'nix based systems

There are many ways you can do it on 'nix based systems (including Mac OS X). Here are two:

For your current user,

1. Open up a shell / terminal window
2. vi ~/.profile (replace vi with your favourite text editor)
3. Add export JAVA_HOME=/path/to/java/home/dir on its own line at the end of the file
4. Add export PATH=$JAVA_HOME/bin:$PATH on its own line immediately after
5. Save, and restart your shell
6. Running java -version should give you the desired results

For all users in the system,

1. Open up a shell / terminal window
2. vi /etc/profile (replace vi with your favourite text editor)
3. Add export JAVA_HOME=/path/to/java/home/dir on its own line at the end of the file
4. Add export PATH=$JAVA_HOME/bin:$PATH on its own line immediately after
5. Save, and restart your shell
6. Running java -version should give you the desired results

If you are using a GUI, you may not need to open up the shell. Instead, you might be able to open the file directly in a graphical text editor.

Related Topics

- 1. System Requirements
  - 1.1 Setting JAVA_HOME
- 2. Installing Crowd and CrowdID
  - 2.1 Connecting Crowd to a Database
    - 2.1.1 HSQL DB
    - 2.1.2 MS SQL Server
    - 2.1.3 MySQL
    - 2.1.4 Oracle
    - 2.1.5 PostgreSQL
  - 2.2 Connecting CrowdID to a Database
    - 2.2.1 HSQL DB
    - 2.2.2 MS SQL Server
    - 2.2.3 MySQL
    - 2.2.4 Oracle
    - 2.2.5 PostgreSQL
  - 2.3 Installing Crowd and CrowdID WAR Distribution
    - 2.3.1 Installing Crowd WAR on JBoss
- 3. Running the Setup Wizard
- 4. Configuring Crowd
  - 4.1 Important Files
- The crowd.properties File
  - 4.2 Changing the Port that Crowd uses
- 5. Installing Crowd as a Windows Service
  - 5.1 Specifying Startup Order of Windows Services
  - 5.2 Changing the User for the Crowd Windows Service
  - 5.3 Removing the Crowd Windows Service
  - 5.4 Troubleshooting Crowd as a Windows Service
2. Installing Crowd and CrowdID

The instructions below tell you how to install the standalone distribution of Crowd, which includes Apache Tomcat. You may wish to deploy a WAR distribution of Crowd or CrowdID on your own existing application server instead.

Note: Crowd versions 1.1 and later include CrowdID. Installing Crowd, as described below, will also install CrowdID.

Installing Crowd (Standalone Distribution)

1. Download Crowd.
2. Unzip the download archive into a directory of your choice (note: do not specify directory names that contain spaces).
   This directory will be referred to as `{CROWD_INSTALL}`.
3. (Optional) If you wish to setup Crowd and/or CrowdID with an external database, see:
   • 2.1 Connecting Crowd to a Database
   • 2.2 Connecting CrowdID to a Database
4. Run the start-up script:
   • `start_crowd.bat` for Windows;
   or:
   • `start_crowd.sh` for Unix environment.
5. Point a web browser at `http://localhost:8095/` where you will see the Setup Wizard.

Related Topics

• 1. System Requirements
  ° 1.1 Setting JAVA_HOME
• 2. Installing Crowd and CrowdID
  ° 2.1 Connecting Crowd to a Database
    - 2.1.1 HSQL DB
    - 2.1.2 MS SQL Server
    - 2.1.3 MySQL
    - 2.1.4 Oracle
    - 2.1.5 PostgreSQL
  ° 2.2 Connecting CrowdID to a Database
    - 2.2.1 HSQL DB
    - 2.2.2 MS SQL Server
    - 2.2.3 MySQL
    - 2.2.4 Oracle
    - 2.2.5 PostgreSQL
  ° 2.3 Installing Crowd and CrowdID WAR Distribution
    - 2.3.1 Installing Crowd WAR on JBoss
• 3. Running the Setup Wizard
• 4. Configuring Crowd
  ° 4.1 Important Files
    - The `crowd.properties` File
  ° 4.2 Changing the Port that Crowd uses
• 5. Installing Crowd as a Windows Service
  ° 5.1 Specifying Startup Order of Windows Services
  ° 5.2 Changing the User for the Crowd Windows Service
  ° 5.3 Removing the Crowd Windows Service
  ° 5.4 Troubleshooting Crowd as a Windows Service
2.1 Connecting Crowd to a Database

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:

- 2.1.1 HSQL DB
- 2.1.2 MS SQL Server
- 2.1.3 MySQL
- 2.1.4 Oracle
- 2.1.5 PostgreSQL

Database Overview

The Crowd Standalone 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 4.1 Important Files. 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:

<table>
<thead>
<tr>
<th>RDBMS</th>
<th>Hibernate SQL Dialect</th>
</tr>
</thead>
<tbody>
<tr>
<td>HypersonicSQL</td>
<td>org.hibernate.dialect.HSQLDialect</td>
</tr>
<tr>
<td>Microsoft SQL Server</td>
<td>org.hibernate.dialect.SQLServerDialect</td>
</tr>
<tr>
<td>MySQL</td>
<td>org.hibernate.dialect.MySQLDialect</td>
</tr>
<tr>
<td>MySQL with InnoDB</td>
<td>org.hibernate.dialect.MySQLInnoDBDialect</td>
</tr>
<tr>
<td>MySQL with MyISAM</td>
<td>org.hibernate.dialect.MySQLMyISAMDDialect</td>
</tr>
<tr>
<td>Oracle</td>
<td>org.hibernate.dialect.OracleDialect</td>
</tr>
<tr>
<td>PostgreSQL</td>
<td>org.hibernate.dialect.PostgreSQLDialect</td>
</tr>
</tbody>
</table>

**hibernate.transaction.factory_class**

You will need to edit the hibernate.transaction.factory_class property to correspond to whichever application server you are using:

<table>
<thead>
<tr>
<th>J2EE Server</th>
<th>Dialect</th>
</tr>
</thead>
<tbody>
<tr>
<td>Borland ES</td>
<td>org.hibernate.transaction.BESTransactionManagerLookup</td>
</tr>
<tr>
<td>JBoss</td>
<td>org.hibernate.transaction.JBossTransactionManagerLookup</td>
</tr>
<tr>
<td>JOnAS</td>
<td>org.hibernate.transaction.JOnASTransactionManagerLookup</td>
</tr>
<tr>
<td>-----------------------</td>
<td>--------------------------------------------------------</td>
</tr>
<tr>
<td>JOTM</td>
<td>org.hibernate.transaction.JOTMTransactionManagerLookup</td>
</tr>
<tr>
<td>JRun4</td>
<td>org.hibernate.transaction.JRun4TransactionManagerLookup</td>
</tr>
<tr>
<td>Orion</td>
<td>org.hibernate.transaction.OrionTransactionManagerLookup</td>
</tr>
<tr>
<td>Resin</td>
<td>org.hibernate.transaction.ResinTransactionManagerLookup</td>
</tr>
<tr>
<td>Weblogic</td>
<td>org.hibernate.transaction.WeblogicTransactionManagerLookup</td>
</tr>
<tr>
<td>WebSphere</td>
<td>org.hibernate.transaction.WebSphereTransactionManagerLookup</td>
</tr>
</tbody>
</table>

Related Topics

- 1. System Requirements
  - 1.1 Setting JAVA_HOME
- 2. Installing Crowd and CrowdID
  - 2.1 Connecting Crowd to a Database
    - 2.1.1 HSQL DB
    - 2.1.2 MS SQL Server
    - 2.1.3 MySQL
    - 2.1.4 Oracle
    - 2.1.5 PostgreSQL
  - 2.2 Connecting CrowdID to a Database
    - 2.2.1 HSQL DB
    - 2.2.2 MS SQL Server
    - 2.2.3 MySQL
    - 2.2.4 Oracle
    - 2.2.5 PostgreSQL
  - 2.3 Installing Crowd and CrowdID WAR Distribution
    - 2.3.1 Installing Crowd WAR on JBoss
- 3. Running the Setup Wizard
- 4. Configuring Crowd
  - 4.1 Important Files
  - 4.2 Changing the Port that Crowd uses
- 5. Installing Crowd as a Windows Service
  - 5.1 Specifying Startup Order of Windows Services
  - 5.2 Changing the User for the Crowd Windows Service
  - 5.3 Removing the Crowd Windows Service
  - 5.4 Troubleshooting Crowd as a Windows Service
2.1.1 HSQL DB

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 crowd.db.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.

HSQLDB should not be used as a production database. It is included for evaluation purposes only.

Related Topics

- 1. System Requirements
  - 1.1 Setting JAVA_HOME
- 2. Installing Crowd and CrowdID
  - 2.1 Connecting Crowd to a Database
    - 2.1.1 HSQL DB
    - 2.1.2 MS SQL Server
    - 2.1.3 MySQL
    - 2.1.4 Oracle
    - 2.1.5 PostgreSQL
  - 2.2 Connecting CrowdID to a Database
    - 2.2.1 HSQL DB
    - 2.2.2 MS SQL Server
    - 2.2.3 MySQL
    - 2.2.4 Oracle
    - 2.2.5 PostgreSQL
  - 2.3 Installing Crowd and CrowdID WAR Distribution
    - 2.3.1 Installing Crowd WAR on JBoss
- 3. Running the Setup Wizard
- 4. Configuring Crowd
  - 4.1 Important Files
    - The crowd.properties File
  - 4.2 Changing the Port that Crowd uses
- 5. Installing Crowd as a Windows Service
  - 5.1 Specifying Startup Order of Windows Services
  - 5.2 Changing the User for the Crowd Windows Service
  - 5.3 Removing the Crowd Windows Service
  - 5.4 Troubleshooting Crowd as a Windows Service
2.1.2 MS SQL Server

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), 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 file apache-tomcat-X.X.XX/conf/Catalina/localhost/crowd.xml and customise the username, password, driverClassName and url parameters for the Datasource.

      ```xml
      <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 distribution) and modify the hibernate.dialect to the following:

      ```properties
      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:

```properties
# - 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**
  - 1.1 Setting JAVA_HOME
- **2. Installing Crowd and CrowdID**
  - 2.1 Connecting Crowd to a Database
    - 2.1.1 HSQL DB
    - 2.1.2 MS SQL Server
    - 2.1.3 MySQL
    - 2.1.4 Oracle
    - 2.1.5 PostgreSQL
  - 2.2 Connecting CrowdID to a Database
    - 2.2.1 HSQL DB
    - 2.2.2 MS SQL Server
    - 2.2.3 MySQL
    - 2.2.4 Oracle
    - 2.2.5 PostgreSQL
  - 2.3 Installing Crowd and CrowdID WAR Distribution
    - 2.3.1 Installing Crowd WAR on JBoss
- **3. Running the Setup Wizard**
- **4. Configuring Crowd**
  - 4.1 Important Files
    - The crowd.properties File
  - 4.2 Changing the Port that Crowd uses
- **5. Installing Crowd as a Windows Service**
  - 5.1 Specifying Startup Order of Windows Services
  - 5.2 Changing the User for the Crowd Windows Service
  - 5.3 Removing the Crowd Windows Service
  - 5.4 Troubleshooting Crowd as a Windows Service
2.1.3 MySQL

To connect Crowd to MySQL (5.0.37 and later),

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](https://dev.mysql.com/downloads/connector/j/).
   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](https://issues.atlassian.com/browse/JRA-8674)).

3. **Configure your application server to connect to MySQL**

   1. Edit the file `apache-tomcat-X.X.XX/conf/Catalina/localhost/crowd.xml` and customise the username, password, driverClassName and url parameters for the Datasource.

   ```xml
   <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&useUnicode=true&characterEncoding=latin1"
         [ delete the minEvictableIdleTimeMillis, timeBetweenEvictionRunsMillis and maxActive params here ]
      />
   <Manager className= "org.apache.catalina.session.PersistentManager" saveOnRestart= "false"/>
   </Context>
   
   The URL above assumes a LATIN-1 database - i.e. created with `create database crowddb character set latin1;`.

   **⚠️** MySQL closes idle connections 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 distribution) and modify the `hibernate.dialect` to the following. Please choose only one of the 3 available options depending on how you have configured your database server.

   ```properties
   *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.MySQLMyISAMDialect
   
   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:

```properties
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**
  - 1.1 Setting JAVA_HOME
- **2. Installing Crowd and CrowdID**
  - 2.1 Connecting Crowd to a Database
    - 2.1.1 HSQL DB
    - 2.1.2 MS SQL Server
    - 2.1.3 MySQL
    - 2.1.4 Oracle
    - 2.1.5 PostgreSQL
  - 2.2 Connecting CrowdID to a Database
    - 2.2.1 HSQL DB
    - 2.2.2 MS SQL Server
    - 2.2.3 MySQL
    - 2.2.4 Oracle
    - 2.2.5 PostgreSQL
- **2.3 Installing Crowd and CrowdID WAR Distribution**
  - 2.3.1 Installing Crowd WAR on JBoss
- **3. Running the Setup Wizard**
- **4. Configuring Crowd**
  - 4.1 Important Files
    - The crowd.properties File
  - 4.2 Changing the Port that Crowd uses
- **5. Installing Crowd as a Windows Service**
  - 5.1 Specifying Startup Order of Windows Services
  - 5.2 Changing the User for the Crowd Windows Service
  - 5.3 Removing the Crowd Windows Service
  - 5.4 Troubleshooting Crowd as a Windows Service
2.1.4 Oracle

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

   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 apache-tomcat-X.X.XX/conf/catalina/localhost/crowd.xml and customise the username, password, driverClassName and url parameters for the Datasource.
   
   ```xml
   <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:crowdb"
            [ 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

   ```properties
   hibernate.dialect=org.hibernate.dialect.OracleDialect
   ...
   ```

   2. Then run ./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:

```properties
hibernate.connection.datasource=java:comp/env/jdbc/CrowdDS
hibernate.dialect=org.hibernate.dialect.OracleDialect
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**
  - 1.1 Setting JAVA_HOME

- **2. Installing Crowd and CrowdID**
  - 2.1 Connecting Crowd to a Database
    - 2.1.1 HSQL DB
    - 2.1.2 MS SQL Server
    - 2.1.3 MySQL
    - 2.1.4 Oracle
    - 2.1.5 PostgreSQL
  - 2.2 Connecting CrowdID to a Database
    - 2.2.1 HSQL DB
    - 2.2.2 MS SQL Server
    - 2.2.3 MySQL
    - 2.2.4 Oracle
    - 2.2.5 PostgreSQL
  - 2.3 Installing Crowd and CrowdID WAR Distribution
    - 2.3.1 Installing Crowd WAR on JBoss

- **3. Running the Setup Wizard**

- **4. Configuring Crowd**
  - 4.1 Important Files
    - The crowd.properties File
  - 4.2 Changing the Port that Crowd uses

- **5. Installing Crowd as a Windows Service**
  - 5.1 Specifying Startup Order of Windows Services
  - 5.2 Changing the User for the Crowd Windows Service
  - 5.3 Removing the Crowd Windows Service
  - 5.4 Troubleshooting Crowd as a Windows Service
2.1.5 PostgreSQL

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
   2. Add the PostgreSQL JDBC driver jar to the common/lib directory.

3. Configure your application server to connect to PostgreSQL
   1. Edit the file apache-tomcat-X.X.XX/conf/Catalina/localhost/crowd.xml and customise the username, password, driverClassName and url parameters for the Datasource.

   ```xml
   <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" />
      <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:

```properties
# - 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**
  - 1.1 Setting JAVA_HOME
- **2. Installing Crowd and CrowdID**
  - 2.1 Connecting Crowd to a Database
    - 2.1.1 HSQL DB
    - 2.1.2 MS SQL Server
    - 2.1.3 MySQL
    - 2.1.4 Oracle
    - 2.1.5 PostgreSQL
  - 2.2 Connecting CrowdID to a Database
    - 2.2.1 HSQL DB
    - 2.2.2 MS SQL Server
    - 2.2.3 MySQL
    - 2.2.4 Oracle
    - 2.2.5 PostgreSQL
  - 2.3 Installing Crowd and CrowdID WAR Distribution
    - 2.3.1 Installing Crowd WAR on JBoss
- **3. Running the Setup Wizard**
- **4. Configuring Crowd**
  - 4.1 Important Files
    - The crowd.properties File
  - 4.2 Changing the Port that Crowd uses
- **5. Installing Crowd as a Windows Service**
  - 5.1 Specifying Startup Order of Windows Services
  - 5.2 Changing the User for the Crowd Windows Service
  - 5.3 Removing the Crowd Windows Service
  - 5.4 Troubleshooting Crowd as a Windows Service
2.2 Connecting CrowdID to a Database

CrowdID is a free add-on that ships with Crowd versions 1.1 and later.

By default, CrowdID in the Crowd 'Standalone' distribution is shipped preconfigured with HSQL. This is fine for evaluation purposes, but for production installations, you should connect CrowdID 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 CrowdID to an external database:

- 2.2.1 HSQL DB
- 2.2.2 MS SQL Server
- 2.2.3 MySQL
- 2.2.4 Oracle
- 2.2.5 PostgreSQL

Database Overview

CrowdID in the Crowd 'Standalone' distribution includes the Apache Tomcat application server and an in-memory HSQL database engine. This JNDI reference (CrowdIDDS) 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 4.1 Important Files. 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:

<table>
<thead>
<tr>
<th>RDBMS</th>
<th>Hibernate SQL Dialect</th>
</tr>
</thead>
<tbody>
<tr>
<td>HypersonicSQL</td>
<td>org.hibernate.dialect.HSQLDialect</td>
</tr>
<tr>
<td>Microsoft SQL Server</td>
<td>org.hibernate.dialect.SQLServerDialect</td>
</tr>
<tr>
<td>MySQL</td>
<td>org.hibernate.dialect.MySQLDialect</td>
</tr>
<tr>
<td>MySQL with InnoDB</td>
<td>org.hibernate.dialect.MySQLInnoDBDialect</td>
</tr>
<tr>
<td>MySQL with MyISAM</td>
<td>org.hibernate.dialect.MySQLMyISAMDDialect</td>
</tr>
<tr>
<td>Oracle</td>
<td>org.hibernate.dialect.OracleDialect</td>
</tr>
<tr>
<td>PostgreSQL</td>
<td>org.hibernate.dialect.PostgreSQLDialect</td>
</tr>
</tbody>
</table>

**hibernate.transaction.factory_class**

You will need to edit the hibernate.transaction.factory_class property to correspond to whichever application server you are using:

<table>
<thead>
<tr>
<th>J2EE Server</th>
<th>Dialect</th>
</tr>
</thead>
<tbody>
<tr>
<td>Borland ES</td>
<td>org.hibernate.transaction.BESTransactionManagerLookup</td>
</tr>
<tr>
<td>----------------------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td>JBoss</td>
<td>org.hibernate.transaction.JBossTransactionManagerLook</td>
</tr>
<tr>
<td>JOnAS</td>
<td>org.hibernate.transaction.JOnASTransactionManagerLook</td>
</tr>
<tr>
<td>JOTM</td>
<td>org.hibernate.transaction.JOTMTransactionManagerLook</td>
</tr>
<tr>
<td>JRun4</td>
<td>org.hibernate.transaction.JRun4TransactionManagerLook</td>
</tr>
<tr>
<td>Orion</td>
<td>org.hibernate.transaction.OrionTransactionManagerLook</td>
</tr>
<tr>
<td>Resin</td>
<td>org.hibernate.transaction.ResinTransactionManagerLook</td>
</tr>
<tr>
<td>Weblogic</td>
<td>org.hibernate.transaction.WeblogicTransactionManager</td>
</tr>
<tr>
<td>WebSphere</td>
<td>org.hibernate.transaction.WebSphereTransactionManager</td>
</tr>
</tbody>
</table>

Related Topics

- **1. System Requirements**
  - 1.1 Setting JAVA_HOME
- **2. Installing Crowd and CrowdID**
  - 2.1 Connecting Crowd to a Database
    - 2.1.1 HSQL DB
    - 2.1.2 MS SQL Server
    - 2.1.3 MySQL
    - 2.1.4 Oracle
    - 2.1.5 PostgreSQL
  - 2.2 Connecting CrowdID to a Database
    - 2.2.1 HSQL DB
    - 2.2.2 MS SQL Server
    - 2.2.3 MySQL
    - 2.2.4 Oracle
    - 2.2.5 PostgreSQL
  - 2.3 Installing Crowd and CrowdID WAR Distribution
    - 2.3.1 Installing Crowd WAR on JBoss
- **3. Running the Setup Wizard**
- **4. Configuring Crowd**
  - 4.1 Important Files
    - The crowd.properties File
  - 4.2 Changing the Port that Crowd uses
- **5. Installing Crowd as a Windows Service**
  - 5.1 Specifying Startup Order of Windows Services
  - 5.2 Changing the User for the Crowd Windows Service
  - 5.3 Removing the Crowd Windows Service
  - 5.4 Troubleshooting Crowd as a Windows Service
2.2.1 HSQL DB

The default version of CrowdID 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 CrowdID in this period, data can be lost, and typically all configuration data for your CrowdID server will be lost.

HSQLDB should not be used as a production database. It is included for evaluation purposes only.

Related Topics

- 1. System Requirements
  - 1.1 Setting JAVA_HOME
- 2. Installing Crowd and CrowdID
  - 2.1 Connecting Crowd to a Database
    - 2.1.1 HSQL DB
    - 2.1.2 MS SQL Server
    - 2.1.3 MySQL
    - 2.1.4 Oracle
    - 2.1.5 PostgreSQL
  - 2.2 Connecting CrowdID to a Database
    - 2.2.1 HSQL DB
    - 2.2.2 MS SQL Server
    - 2.2.3 MySQL
    - 2.2.4 Oracle
    - 2.2.5 PostgreSQL
  - 2.3 Installing Crowd and CrowdID WAR Distribution
    - 2.3.1 Installing Crowd WAR on JBoss
- 3. Running the Setup Wizard
- 4. Configuring Crowd
  - 4.1 Important Files
    - The `crowd.properties` File
  - 4.2 Changing the Port that Crowd uses
- 5. Installing Crowd as a Windows Service
  - 5.1 Specifying Startup Order of Windows Services
  - 5.2 Changing the User for the Crowd Windows Service
  - 5.3 Removing the Crowd Windows Service
  - 5.4 Troubleshooting Crowd as a Windows Service
2.2.2 MS SQL Server

To connect CrowdID to MS SQL Server,

1. **Configure SQL Server**
   
   1. Create a database user which CrowdID 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 CrowdID to store data in (e.g. crowdiddb). This must be a different database to the one used by Crowd.
   
   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.
      
      ```xml
      <Context path="/openidserver" docBase="../../crowd-openidserver-webapp" debug= "0">
      <Resource name= "jdbc/CrowdIDDS" 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/crowdiddb"
      [ delete the minEvictableIdleTimeMillis, timeBetweenEvictionRunsMillis and maxActive params here ]
      />
      </Context>
      
      <Manager className="org.apache.catalina.session.PersistentManager" saveOnRestart="false"/>
      
      2. Delete the minEvictableIdleTimeMillis, timeBetweenEvictionRunsMillis and maxActive attributes (which are only needed for HSQL, and degrade performance otherwise).

4. **Configure CrowdID to use MS SQL Server**

   1. Edit the build.properties file (located in the root of the Standalone distribution) and modify the hibernate.dialect to the following:
      
      ```java
      hibernate.dialect=org.hibernate.dialect.SQLServerDialect
      ```
   
   2. Then run the ./build.sh or build.bat. This will configure CrowdID 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-openidserver-webapp \WEB-INF\classes\jdbc.properties; modify the file to the following:
# - Crowd Configuration Options

```java
hibernate.connection.datasource=java\:comp/env/jdbc/CrowdIDDS
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 CrowdID configured to use the correct database. Now start up CrowdID and watch the logs for any errors.

**Related Topics**

- **1. System Requirements**
  - 1.1 Setting JAVA_HOME
- **2. Installing Crowd and CrowdID**
  - 2.1 Connecting Crowd to a Database
    - 2.1.1 HSQL DB
    - 2.1.2 MS SQL Server
    - 2.1.3 MySQL
    - 2.1.4 Oracle
    - 2.1.5 PostgreSQL
  - 2.2 Connecting CrowdID to a Database
    - 2.2.1 HSQL DB
    - 2.2.2 MS SQL Server
    - 2.2.3 MySQL
    - 2.2.4 Oracle
    - 2.2.5 PostgreSQL
  - 2.3 Installing Crowd and CrowdID WAR Distribution
    - 2.3.1 Installing Crowd WAR on JBoss
- **3. Running the Setup Wizard**
- **4. Configuring Crowd**
  - 4.1 Important Files
    - The crowd.properties File
  - 4.2 Changing the Port that Crowd uses
- **5. Installing Crowd as a Windows Service**
  - 5.1 Specifying Startup Order of Windows Services
  - 5.2 Changing the User for the Crowd Windows Service
  - 5.3 Removing the Crowd Windows Service
  - 5.4 Troubleshooting Crowd as a Windows Service
2.2.3 MySQL

To connect CrowdID 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. crowdiddb). This must be a different database to the one used by Crowd.
   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 file apache-tomcat-X.X.XX/conf/Catalina/localhost/openidserver.xml and customise the username, password, driverClassName and url parameters for the Datasource.

```
<Context path="/openidserver" docBase="../crowd-openidserver-webapp" debug="0">
  <Resource name="jdbc/CrowdIDDS" 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/crowdiddb?autoReconnect=true&amp;useUnicode=true&amp;characterEncoding=latin1" />
</Context>
```

   The URL above assumes a LATIN-1 database - i.e. created with `create database crowdb character set latin1`.

   MySQL closes idle connections 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 CrowdID to use MySQL

   1. Edit the build.properties file (located in the root of the Standalone distribution) and modify the hibernate.dialect to the following. Please choose only one of the 3 available options depending on how you have configured your database server.

```
hibernate.dialect=org.hibernate.dialect.MySQLDialect
```

   2. Then run ./build.sh or build.bat. This will configure CrowdID 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-openidserver-webapp\WEB-INF\classes\jdbc.properties`. Modify the file to the following:

```java
# - Crowd Configuration Options

hibernate.connection.datasource=java:comp/env/jdbc/CrowdIDDS
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 CrowdID configured to use the correct database. Now start up CrowdID and watch the logs for any errors.

Related Topics

- **1. System Requirements**
  - 1.1 Setting JAVA_HOME
- **2. Installing Crowd and CrowdID**
  - 2.1 Connecting Crowd to a Database
    - 2.1.1 HSQL DB
    - 2.1.2 MS SQL Server
    - 2.1.3 MySQL
    - 2.1.4 Oracle
    - 2.1.5 PostgreSQL
  - 2.2 Connecting CrowdID to a Database
    - 2.2.1 HSQL DB
    - 2.2.2 MS SQL Server
    - 2.2.3 MySQL
    - 2.2.4 Oracle
    - 2.2.5 PostgreSQL
  - 2.3 Installing Crowd and CrowdID WAR Distribution
    - 2.3.1 Installing Crowd WAR on JBoss
- **3. Running the Setup Wizard**
- **4. Configuring Crowd**
  - 4.1 Important Files
    - The crowd.properties File
  - 4.2 Changing the Port that Crowd uses
- **5. Installing Crowd as a Windows Service**
  - 5.1 Specifying Startup Order of Windows Services
  - 5.2 Changing the User for the Crowd Windows Service
  - 5.3 Removing the Crowd Windows Service
  - 5.4 Troubleshooting Crowd as a Windows Service
2.2.4 Oracle

To connect CrowdID 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. crowdiddb). This must be a different database to the one used by Crowd.
   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
   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 `apache-tomcat-X.X.XX/conf/Catalina/localhost/openidserver.xml` and customise the username, password, driverClassName and url parameters for the Datasource.

   ```xml
   <Context path= "openidserver" docBase="../../../crowd-openidserver-webapp" debug="0">
     <Resource name="jdbc/CrowdIDDS" 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:crowdiddb"
               />
   </Context>
   ``
   2. Delete the minEvictableIdleTimeMillis, timeBetweenEvictionRunsMillis and maxActive attributes (which are only needed for HSQL, and degrade performance otherwise).

4. Configure CrowdID 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 `./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-openidserver-webapp/WEB-INF/classes/jdbc.properties`. Modify the file to the following:

```
# - Crowd Configuration Options

hibernate.connection.datasource=java\:comp\:env\:jdbc\:CrowdIDDS
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 CrowdID configured to use the correct database. Now start up CrowdID and watch the logs for any errors.

Related Topics

- 1. System Requirements
  - 1.1 Setting JAVA_HOME
- 2. Installing Crowd and CrowdID
  - 2.1 Connecting Crowd to a Database
    - 2.1.1 HSQL DB
    - 2.1.2 MS SQL Server
    - 2.1.3 MySQL
    - 2.1.4 Oracle
    - 2.1.5 PostgreSQL
  - 2.2 Connecting CrowdID to a Database
    - 2.2.1 HSQL DB
    - 2.2.2 MS SQL Server
    - 2.2.3 MySQL
    - 2.2.4 Oracle
    - 2.2.5 PostgreSQL
  - 2.3 Installing Crowd and CrowdID WAR Distribution
    - 2.3.1 Installing Crowd WAR on JBoss
- 3. Running the Setup Wizard
- 4. Configuring Crowd
  - 4.1 Important Files
    - The crowd.properties File
  - 4.2 Changing the Port that Crowd uses
- 5. Installing Crowd as a Windows Service
  - 5.1 Specifying Startup Order of Windows Services
  - 5.2 Changing the User for the Crowd Windows Service
  - 5.3 Removing the Crowd Windows Service
  - 5.4 Troubleshooting Crowd as a Windows Service
2.2.5 PostgreSQL

To connect CrowdID to PostgreSQL,

1. **Configure PostgreSQL**

   1. Create a database user which CrowdID will connect as (e.g. crowduser).
   2. Create a database for CrowdID to store data in (e.g. crowdiddb). This must be a different database to the one used by Crowd.
   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**

   2. Add the PostgreSQL JDBC driver jar to the common/lib directory.

3. **Configure your application server to connect to PostgreSQL**

   1. Edit the file `apache-tomcat-X.X.XX/conf/Catalina/localhost/openidserver.xml` and customise the username, password, driverClassName and url parameters for the Datasource.

   ```xml
   <Context path="/openidserver" docBase="/../../crowd-openidserver-webapp" debug="0">
   
   <Resource name="jdbc/CrowdIDDS" 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/crowdiddb" [ 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 CrowdID 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

   ```properties
   hibernate.dialect=org.hibernate.dialect.PostgreSQLDialect
   ```

   2. Then run `./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-openidserver-webapp\WEB-INF\classes\jdbc.properties`. Modify the file to the following:

```properties
# - Crowd Configuration Options

hibernate.connection.datasource=java:comp/env/jdbc/CrowdIDDS
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 CrowdID configured to use the correct database. Now start up CrowdID and watch the logs for any errors.

Related Topics

- **1. System Requirements**
  - 1.1 Setting JAVA_HOME
- **2. Installing Crowd and CrowdID**
  - 2.1 Connecting Crowd to a Database
    - 2.1.1 HSQL DB
    - 2.1.2 MS SQL Server
    - 2.1.3 MySQL
    - 2.1.4 Oracle
    - 2.1.5 PostgreSQL
  - 2.2 Connecting CrowdID to a Database
    - 2.2.1 HSQL DB
    - 2.2.2 MS SQL Server
    - 2.2.3 MySQL
    - 2.2.4 Oracle
    - 2.2.5 PostgreSQL
  - 2.3 Installing Crowd and CrowdID WAR Distribution
    - 2.3.1 Installing Crowd WAR on JBoss
- **3. Running the Setup Wizard**
- **4. Configuring Crowd**
  - 4.1 Important Files
    - The crowd.properties File
  - 4.2 Changing the Port that Crowd uses
- **5. Installing Crowd as a Windows Service**
  - 5.1 Specifying Startup Order of Windows Services
  - 5.2 Changing the User for the Crowd Windows Service
  - 5.3 Removing the Crowd Windows Service
  - 5.4 Troubleshooting Crowd as a Windows Service
2.3 Installing Crowd and CrowdID WAR Distribution

The Crowd WAR distribution is intended for deployment into an existing J2EE application server. It is assumed that you already know how to deploy a web application on the application server of choice. If not, please contact your system administrator to assist you, or consider installing the Crowd 'Standalone' distribution instead.

The standard Crowd installation guide tells you how to install the 'Standalone' distribution of Crowd, which includes Apache Tomcat. Instead, you may wish to deploy Crowd or CrowdID onto your own existing application server. For this purpose, we provide WAR (Webapp ARchive) distributions of the Crowd and CrowdID server applications.

Crowd supports all the application servers listed in System Requirements.

Below is a generic overview of the steps required. You will need to perform specific configuration steps, depending upon your application server. As an example of the installation steps, you can also refer to the following specific instructions:

- **2.3.1 Installing Crowd WAR on JBoss**

Overview of the WAR Installation Steps

1. Download the WAR distribution from the Crowd download centre.
   
   You will find the WAR archives for the Crowd and the CrowdID applications. You will need to deploy each application separately. For the rest of these instructions, we assume you are deploying Crowd WAR.

2. Unzip the download archive into a directory of your choice. We'll call it CROWD in the rest of these instructions.

3. Create a database in your chosen database server and add the required datasource definition file to your application server.

4. Modify file CROWD/WEB-INF/classes/jdbc.properties to use your chosen database dialect.

5. Modify file CROWD/WEB-INF/classes/crowd.properties to point to the port of your application server (8080 default):

   crowd.server.url=http://localhost:8080/crowd/services/
   application.login.url=http://localhost:8080/crowd/console/

6. Depending upon your application server, you may need to zip up the WAR file again before deploying it. Place the CROWD directory or the WAR file into your application server's deployment directory. Please consult the server-specific documentation on how to do this.

7. Restart your application server.

8. Point a web browser at the IP address and port that your application server is running on (typically http://localhost:8080). The Crowd Setup Wizard will start.

Related Topics

- **1. System Requirements**
  - 1.1 Setting JAVA_HOME
- **2. Installing Crowd and CrowdID**
  - 2.1 Connecting Crowd to a Database
    - 2.1.1 HSQL DB
    - 2.1.2 MS SQL Server
    - 2.1.3 MySQL
    - 2.1.4 Oracle
    - 2.1.5 PostgreSQL
  - 2.2 Connecting CrowdID to a Database
    - 2.2.1 HSQL DB
    - 2.2.2 MS SQL Server
    - 2.2.3 MySQL
    - 2.2.4 Oracle
    - 2.2.5 PostgreSQL
  - 2.3 Installing Crowd and CrowdID WAR Distribution
2.3.1 Installing Crowd WAR on JBoss

3. Running the Setup Wizard

4. Configuring Crowd
   - 4.1 Important Files
     - The crowd.properties File
   - 4.2 Changing the Port that Crowd uses

5. Installing Crowd as a Windows Service
   - 5.1 Specifying Startup Order of Windows Services
   - 5.2 Changing the User for the Crowd Windows Service
   - 5.3 Removing the Crowd Windows Service
   - 5.4 Troubleshooting Crowd as a Windows Service
2.3.1 Installing Crowd WAR on JBoss

This page last changed on Nov 26, 2007 by smaddox.

The standard Crowd installation guide tells you how to install the 'Standalone' distribution of Crowd, which includes Apache Tomcat. You may wish to deploy Crowd on your own existing application server instead. For this purpose, we provide WAR (Webapp ARchive) distributions of the Crowd and CrowdID server applications.

Follow the steps below to install Crowd on JBoss 4.2.2 GA using a PostgreSQL database:

1. Download the WAR distribution from the Crowd download centre. You will find the WAR archives for the Crowd and the CrowdID applications. You will need to deploy each application separately. For the rest of these instructions, we assume you are deploying Crowd WAR.

2. Unzip the download archive into a directory of your choice. We'll call it server/default/deploy/crowd.war in the rest of these instructions.

3. Add file server/default/deploy/crowd.war/WEB-INF/jboss-web.xml, with the following contents:

   ```xml
   <jboss-web>
   <resource-ref>
   <res-ref-name> jdbc/CrowdDS </res-ref-name>
   <res-type> javax.sql.DataSource </res-type>
   <jndi-name> java:CrowdDS </jndi-name>
   </resource-ref>
   </jboss-web>
   ```

4. Create database crowd_db in PostgreSQL.

5. Add a datasource definition file server/default/deploy/postgres-ds.xml:

   ```xml
   <datasources>
   <local-tx-datasource>
   <jndi-name>CrowdDS</jndi-name>
   <connection-url>jdbc:postgresql://localhost:5432/crowd_db</connection-url>
   <driver-class>org.postgresql.Driver</driver-class>
   <user-name>postgres</user-name>
   <password>postgres</password>
   </local-tx-datasource>
   </datasources>
   ```

6. Modify file server/default/deploy/crowd.war/WEB-INF/classes/jdbc.properties to use the PostgreSQL dialect:

   ```properties
   hibernate.dialect=org.hibernate.dialect.PostgreSQLDialect
   ```

7. Modify file server/default/deploy/crowd.war/WEB-INF/classes/crowd.properties to point to the port of the JBoss server (8080 default):

   ```properties
   crowd.server.url=http://localhost:8080/crowd/services/
   application.login.url=http://localhost:8080/crowd/console/
   ```

8. Start JBoss with run.sh.

9. Point a web browser at http://localhost:8080/ where you will see the Crowd Setup Wizard.

Related Topics

- 1. System Requirements
  - 1.1 Setting JAVA_HOME
- 2. Installing Crowd and CrowdID
2.1 Connecting Crowd to a Database
   - 2.1.1 HSQL DB
   - 2.1.2 MS SQL Server
   - 2.1.3 MySQL
   - 2.1.4 Oracle
   - 2.1.5 PostgreSQL

2.2 Connecting CrowdID to a Database
   - 2.2.1 HSQL DB
   - 2.2.2 MS SQL Server
   - 2.2.3 MySQL
   - 2.2.4 Oracle
   - 2.2.5 PostgreSQL

2.3 Installing Crowd and CrowdID WAR Distribution
   - 2.3.1 Installing Crowd WAR on JBoss

3. Running the Setup Wizard

4. Configuring Crowd
   - 4.1 Important Files
     - The crowd.properties File
   - 4.2 Changing the Port that Crowd uses

5. Installing Crowd as a Windows Service
   - 5.1 Specifying Startup Order of Windows Services
   - 5.2 Changing the User for the Crowd Windows Service
   - 5.3 Removing the Crowd Windows Service
   - 5.4 Troubleshooting Crowd as a Windows Service
3. Running the Setup Wizard

To access the Crowd Administration Console and run the Setup Wizard, go to the URL http://localhost:8095/crowd or http://localhost:8095/crowd/console.

Welcome to the Setup Wizard

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 log in to 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 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'
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: 'Default Administrator'

6. Integrated Applications

You have the option to auto-configure two applications. It is recommended that you select 'True' for both:

- OpenID Server — This application (CrowdID) allows you to provide OpenID services for your end-users. For details please see the CrowdID Administration Guide and the CrowdID User Guide.
- Demo Application — The 'demo' application highlights best practices for using the Crowd framework, and is provided to assist you with quickly setting up and configuring Crowd. 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: 'Integrated Applications'

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**
  - 1.1 Setting JAVA_HOME

- **2. Installing Crowd and CrowdID**
  - 2.1 Connecting Crowd to a Database
    - 2.1.1 HSQL DB
    - 2.1.2 MS SQL Server
    - 2.1.3 MySQL
    - 2.1.4 Oracle
    - 2.1.5 PostgreSQL
  - 2.2 Connecting CrowdID to a Database
    - 2.2.1 HSQL DB
    - 2.2.2 MS SQL Server
    - 2.2.3 MySQL
    - 2.2.4 Oracle
    - 2.2.5 PostgreSQL
  - 2.3 Installing Crowd and CrowdID WAR Distribution
    - 2.3.1 Installing Crowd WAR on JBoss

- **3. Running the Setup Wizard**

- **4. Configuring Crowd**
  - 4.1 Important Files
    - The crowd.properties File
  - 4.2 Changing the Port that Crowd uses

- **5. Installing Crowd as a Windows Service**
  - 5.1 Specifying Startup Order of Windows Services
  - 5.2 Changing the User for the Crowd Windows Service
  - 5.3 Removing the Crowd Windows Service
  - 5.4 Troubleshooting Crowd as a Windows Service
4. Configuring Crowd

Configuring Crowd

You can configure Crowd to suit your environment, as described in:

- 4.1 Important Files
- 4.2 Changing the Port that Crowd uses

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).

⚠ When configuring an application to work with Crowd, you will be interested in the crowd.properties file.

For more details please see 4.1 Important Files.

Related Topics

- Configuring an SSL Certificate for Microsoft Active Directory
- 1. System Requirements
  - 1.1 Setting JAVA_HOME
- 2. Installing Crowd and CrowdID
  - 2.1 Connecting Crowd to a Database
    - 2.1.1 HSQL DB
    - 2.1.2 MS SQL Server
    - 2.1.3 MySQL
    - 2.1.4 Oracle
    - 2.1.5 PostgreSQL
  - 2.2 Connecting CrowdID to a Database
    - 2.2.1 HSQL DB
    - 2.2.2 MS SQL Server
    - 2.2.3 MySQL
    - 2.2.4 Oracle
    - 2.2.5 PostgreSQL
  - 2.3 Installing Crowd and CrowdID WAR Distribution
    - 2.3.1 Installing Crowd WAR on JBoss
- 3. Running the Setup Wizard
- 4. Configuring Crowd
  - 4.1 Important Files
  - The crowd.properties File
  - 4.2 Changing the Port that Crowd uses
- 5. Installing Crowd as a Windows Service
  - 5.1 Specifying Startup Order of Windows Services
  - 5.2 Changing the User for the Crowd Windows Service
  - 5.3 Removing the Crowd Windows Service
  - 5.4 Troubleshooting Crowd as a Windows Service
4.1 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`).

When configuring an application to work with Crowd, you will be interested in the `crowd.properties` file.

**build.properties**

The default `build.properties` file will look similar to the following:

```properties
# Modify the attributes of this file to quickly adjust the deployment values of Crowd.

# The Hibernate database dialect to use.
hibernate.dialect=org.hibernate.dialect.HSQLDialect

# The Hibernate transaction factory to use.
hibernate.transaction.factory_class=org.hibernate.transaction.JDBCTransactionFactory

# The http port you wish to run crowd from, ie: http://localhost:8095/crowd
crowd.tomcat.connector.port=8095

crowd.tomcat.shutdown.port=8020

crowd.url=http://localhost:8095/crowd
demo.url=http://localhost:8095/demo

# OpenID server context root
openidserver.url=http://localhost:8095/openidserver
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hibernate.dialect</td>
<td>This parameter controls the database dialect the Hibernate persistence system will use when executing commands versus your database server.</td>
</tr>
<tr>
<td>hibernate.transaction.factory_class</td>
<td>This parameter controls the transaction factory to use when executing transactions at runtime: Hibernate provides two generic options, additional application server specific options are available:</td>
</tr>
<tr>
<td></td>
<td>org.hibernate.transaction.JDBCTransactionFactory delegates to database</td>
</tr>
</tbody>
</table>
(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 root of the Crowd demo web-application
openidserver.url  The path and port for the root of the CrowdID 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:

```
shamid@mocha:/atlassian-crowd-1.1.0$ ./build.sh
Buildfile: build.xml
init:
  assistant:
  Changing Tomcat's connector port to 8095
  Changing Tomcat's shutdown port to 8020
  Configuring the Crowd Console
  Copying crowd.properties to: crowd-webapp/WEB-INF/classes
  Copying 1 file to /home/shamid/atlassian-crowd-1.1.0/crowd-webapp/WEB-INF/classes
  Configuring the Crowd hibernate configuration
  Updating the HibernateDialect and TransactionFactory in crowd-webapp/WEB-INF/classes/jdbc.properties
  Updating property file: /home/shamid/atlassian-crowd-1.1.0/crowd-webapp/WEB-INF/classes/jdbc.properties
  Configuring the demo application
  Renaming and copying demo.properties to: demo-webapp/WEB-INF/classes/crowd.properties
  Copying 1 file to /home/shamid/atlassian-crowd-1.1.0/demo-webapp/WEB-INF/classes
  Configuring the OpenID server application
  Renaming and copying openidserver.properties to: crowd-openidserver-webapp/WEB-INF/classes/crowd.properties
  Copying 1 file to /home/shamid/atlassian-crowd-1.1.0/crowd-openidserver-webapp/WEB-INF/classes
  Configuring the OpenID hibernate configuration
  Updating the HibernateDialect and TransactionFactory in crowd-openidserver-webapp/WEB-INF/classes/jdbc.properties
  Updating property file: /home/shamid/atlassian-crowd-1.1.0/crowd-openidserver-webapp/WEB-INF/classes/jdbc.properties

BUILD SUCCESSFUL
Total time: 2 seconds
```

**Related Topics**

- [1. System Requirements](#)
  - [1.1 Setting JAVA_HOME](#)
• 2. Installing Crowd and CrowdID
  ° 2.1 Connecting Crowd to a Database
    - 2.1.1 HSQL DB
    - 2.1.2 MS SQL Server
    - 2.1.3 MySQL
    - 2.1.4 Oracle
    - 2.1.5 PostgreSQL
  ° 2.2 Connecting CrowdID to a Database
    - 2.2.1 HSQL DB
    - 2.2.2 MS SQL Server
    - 2.2.3 MySQL
    - 2.2.4 Oracle
    - 2.2.5 PostgreSQL
  ° 2.3 Installing Crowd and CrowdID WAR Distribution
    - 2.3.1 Installing Crowd WAR on JBoss
• 3. Running the Setup Wizard
• 4. Configuring Crowd
  ° 4.1 Important Files
    - The crowd.properties File
  ° 4.2 Changing the Port that Crowd uses
• 5. Installing Crowd as a Windows Service
  ° 5.1 Specifying Startup Order of Windows Services
  ° 5.2 Changing the User for the Crowd Windows Service
  ° 5.3 Removing the Crowd Windows Service
  ° 5.4 Troubleshooting Crowd as a Windows Service
## The crowd.properties File

This page last changed on Nov 08, 2007 by smaddox.

The attributes of the `crowd.properties` file are as follows:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>application.name</td>
<td>The name that the application will use when authenticating with the Crowd server. This needs to match the name you specified in <a href="#">3.2 Adding an Application</a>.</td>
</tr>
<tr>
<td>application.password</td>
<td>The password that the application will use when authenticating with the Crowd server. This needs to match the password you specified in <a href="#">3.2 Adding an Application</a>.</td>
</tr>
<tr>
<td>application.login.url</td>
<td>The URL to which to redirect the principal should their authentication token expire or be invalid due to security restrictions.</td>
</tr>
<tr>
<td>crowd.server.url</td>
<td>The URL to use when connecting with the integration libraries to communicate with the Crowd server.</td>
</tr>
<tr>
<td>session.isauthenticated</td>
<td>The session key to use when storing a <code>Boolean</code> value indicating whether the principal is authenticated or not.</td>
</tr>
<tr>
<td>session.tokenkey</td>
<td>The session key to use when storing a <code>String</code> value of the principal's authentication token.</td>
</tr>
<tr>
<td>session.validationinterval</td>
<td>The session key to use when storing an <code>Integer</code> value of the number of minutes between authentication validation. If this value is set to 0, each HTTP request will be authenticated.</td>
</tr>
<tr>
<td>session.lastvalidation</td>
<td>The session key to use when storing a <code>Date</code> value of the principal's last authentication.</td>
</tr>
</tbody>
</table>
4.2 Changing the Port that Crowd uses

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 4.1 Important Files.
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. Change the `openidserver.url` property to the new port on which the CrowdID Server will be accessed.
5. Run the `build.xml` script, as described in 4.1 Important Files.

Related Topics

- 1. System Requirements
  - 1.1 Setting JAVA_HOME
- 2. Installing Crowd and CrowdID
  - 2.1 Connecting Crowd to a Database
    - 2.1.1 HSQL DB
    - 2.1.2 MS SQL Server
    - 2.1.3 MySQL
    - 2.1.4 Oracle
    - 2.1.5 PostgreSQL
  - 2.2 Connecting CrowdID to a Database
    - 2.2.1 HSQL DB
    - 2.2.2 MS SQL Server
    - 2.2.3 MySQL
    - 2.2.4 Oracle
    - 2.2.5 PostgreSQL
  - 2.3 Installing Crowd and CrowdID WAR Distribution
    - 2.3.1 Installing Crowd WAR on JBoss
- 3. Running the Setup Wizard
- 4. Configuring Crowd
  - 4.1 Important Files
    - The `crowd.properties` File
  - 4.2 Changing the Port that Crowd uses
- 5. Installing Crowd as a Windows Service
  - 5.1 Specifying Startup Order of Windows Services
  - 5.2 Changing the User for the Crowd Windows Service
  - 5.3 Removing the Crowd Windows Service
  - 5.4 Troubleshooting Crowd as a Windows Service
5. Installing Crowd as a Windows Service

This page last changed on Oct 09, 2007 by smaddox.

For long-term use, you should configure Crowd to restart automatically when the operating system restarts. For Windows servers, this means configuring Crowd to run as a Windows service.

Running Crowd as a Windows service has other advantages. When Crowd is started manually, a console window opens - there is a risk that someone may accidentally shut down Crowd by closing the window. Also, the Crowd logs are properly managed by the Windows service (reliably found in \atlassian-crowd.log in the root Crowd directory, and rotated by file size).

Installing Crowd as a Windows Service

1. Open a DOS prompt.
2. ‘cd’ to your Crowd directory, and then the Tomcat bin subdirectory, e.g. {CROWD_INSTALL}\apache-tomcat-5.5.20\bin
3. If a directory in the path has spaces (e.g. C:\Program Files\..), please convert it to its eight-character equivalent (e.g. c:\Progra~1\..).
4. Ensure the JAVA_HOME variable is set to the JDK base directory. Use echo %JAVA_HOME% to confirm this.
5. Run the following command:
   service.bat install Crowd

   Screenshot: Installing Crowd as a Windows Service

Crowd should now have been installed as a service, and will be visible in the Windows Services console.

Screenshot: Windows Services Console

6. Run the following command, to have the Crowd service start automatically when the server starts:
   tomcat5 //US//Crowd --Startup auto

   The Crowd service will automatically start up the next time the server reboots.

   • You can manually start the Crowd service with the command net start Crowd, and stop it with net stop Crowd.
Additional Crowd Setup Options (Optional)

- To see what parameters the Crowd service is starting with, go to Start -> Run and run `regedt32.exe`. There should be an entry at HKEY_LOCAL_MACHINE -> SOFTWARE -> Apache Software Foundation -> Procrun 2.0 -> Crowd.

**Related Topics**

- **5.1 Specifying Startup Order of Windows Services**
- **5.2 Changing the User for the Crowd Windows Service**
- **5.3 Removing the Crowd Windows Service**
- **5.4 Troubleshooting Crowd as a Windows Service**
5.1 Specifying Startup Order of Windows Services

This page is relevant if you have **installed Crowd as a Windows service**.

If you have multiple Windows services that depend on each other, it is important that they are started in the correct order. For example, if you are running both JIRA and Crowd, it is important to start Crowd first, so that Crowd is running before people try to login to JIRA.

For information about specifying the startup order for multiple services, please refer to [support.microsoft.com/kb/193888](http://support.microsoft.com/kb/193888).

Related Topics

- 5.1 Specifying Startup Order of Windows Services
- 5.2 Changing the User for the Crowd Windows Service
- 5.3 Removing the Crowd Windows Service
- 5.4 Troubleshooting Crowd as a Windows Service

- 5, Installing Crowd as a Windows Service
5.2 Changing the User for the Crowd Windows Service

This page was last changed on Aug 28, 2007 by smaddox.

This page is relevant if you have installed Crowd as a Windows service. You may want to change the user under which the Crowd Windows service is running, for security reasons.

Changing the Windows User for the Crowd Service

2. Locate the 'Apache Tomcat Crowd' service, right-click and view the 'Properties'.
3. Go to the 'Log On' tab and change the user as desired.

Screenshot: Changing the User for the Windows Service

Related Topics

- 5.1 Specifying Startup Order of Windows Services
- 5.2 Changing the User for the Crowd Windows Service
- 5.3 Removing the Crowd Windows Service
- 5.4 Troubleshooting Crowd as a Windows Service
- 5. Installing Crowd as a Windows Service
5.3 Removing the Crowd Windows Service

This page is relevant if you have installed Crowd as a Windows service.

To remove the Crowd Windows service:

1. Open a DOS prompt.
2. 'cd' to your Crowd directory, and then the Tomcat bin subdirectory, e.g. \{CROWD_INSTALL\\apache-tomcat-5.5.20\\bin
3. Run one of the following commands:
   • Either:
     
     service.bat remove Crowd
   • Or if the above does not work, use

     tomcat5 /DS://Crowd

Related Topics

- 5.1 Specifying Startup Order of Windows Services
- 5.2 Changing the User for the Crowd Windows Service
- 5.3 Removing the Crowd Windows Service
- 5.4 Troubleshooting Crowd as a Windows Service
- 5. Installing Crowd as a Windows Service
5.4 Troubleshooting Crowd as a Windows Service

This page is relevant if you have installed Crowd as a Windows service.

Problems may occur when trying to set up Crowd to run as a Windows service with JDK 1.6. The problem is caused by a failure to locate MSVCR71.DLL, which can be found in your %JAVA_HOME%/bin. There are two options to resolve this problem:

- Add %JAVA_HOME/bin to PATH, then restart the server.
- Or copy MSVCR71.DLL to system path: either C:\WINDOWS\SYSTEM32 or C:\WINNT\SYSTEM32

Related Topics

- 5.1 Specifying Startup Order of Windows Services
- 5.2 Changing the User for the Crowd Windows Service
- 5.3 Removing the Crowd Windows Service
- 5.4 Troubleshooting Crowd as a Windows Service
- 5. Installing Crowd as a Windows Service
Upgrading Crowd

This page last changed on Nov 26, 2007 by smaddox.

Step 1. Preparation

Please read the Release Notes for the version you are upgrading to, and the Upgrade Notes for any versions you are skipping:

- Crowd 1.2 Upgrade Notes
- Crowd 1.1 Upgrade Notes
- Crowd 1.0 Upgrade Notes

Step 2. Shut down Crowd and all integrated applications

You need to shut down Crowd and all Crowd-connected applications before backing up or performing the upgrade.

Step 3. Back up your Crowd files

Back up the following files:

- {CROWD_INSTALL}/database (back up the entire directory) (Note: if you are using an external database, this is not required)
- {CROWD_INSTALL}/crowd-webapp/WEBINF/classes/crowd.properties
- {CROWD_INSTALL}/crowd-webapp/WEBINF/classes/jdbc.properties
- {CROWD_INSTALL}/crowd-openidserver-webapp/WEBINF/classes/crowd.properties (Note: if you are upgrading from Crowd 1.0, this file may not exist)
- {CROWD_INSTALL}/crowd-openidserver-webapp/WEBINF/classes/jdbc.properties (Note: if you are upgrading from Crowd 1.0, this file may not exist)
- If you are using the 'Standalone' distribution of Crowd, back up the following files as well:
  - {CROWD_INSTALL}/apache-tomcat-X.X.XX/conf/Catalina/localhost/crowd.xml
  - {CROWD_INSTALL}/apache-tomcat-X.X.XX/conf/Catalina/localhost/openidserver.xml (Note: if you are upgrading from Crowd 1.0, this file may not exist)
  - {CROWD_INSTALL}/apache-tomcat-X.X.XX/conf/server.xml (Required if you are not running on port 8095)
  - Your database driver, e.g. {CROWD_INSTALL}/apache-tomcat-X.X.XX/common/lib/mysql*.jar (Note: if you are using an external database, this is not required)
- If you have installed Crowd on a separate application server, you need to back up your customised configuration files.

Step 4. Re-install Crowd

1. Download Crowd.
2. Unzip the download archive into a directory of your choice (note: do not specify directory names that contain spaces). This directory will be referred to as {CROWD_INSTALL}.

⚠️ Please make sure that your new {CROWD_INSTALL} directory has a different name from your old {CROWD_INSTALL} directory (e.g. you may need to rename your old {CROWD_INSTALL} directory before you begin), as legacy files may cause problems.

Step 5. Restore your Crowd files

Copy the files from Step 3 to your new {CROWD_INSTALL} directory.

Step 6. Start Crowd

Run the start-up script:

- start_crowd.bat for Windows;
The upgrade process will be performed when Crowd starts up. You will not see the Setup Wizard.

Monitor the atlassian-crowd.log to ensure that the upgrade process has completed successfully.

**Step 7. Update your integrated applications**

Copy the new `client/crowd-XXXX-X.X.X.jar` files to each Crowd-integrated application’s WEB-INF/lib folder, replacing the existing `crowd-XXXX-X.X.X.jar` files.

For details please see the configuration instructions for each application:

- [3.2.01 Integrating Crowd with Apache](#)
- [3.2.02 Integrating Crowd with Subversion](#)
- [3.2.03 Integrating Crowd with Atlassian Confluence](#)
- [3.2.04 Integrating Crowd with Atlassian CrowdID](#)
- [3.2.05 Integrating Crowd with Atlassian FishEye](#)
- [3.2.06 Integrating Crowd with Atlassian JIRA](#)
- [3.2.07 Integrating Crowd with Jive Forums](#)
- [3.2.08 Integrating Crowd with Atlassian Bamboo](#)
- [3.2.09 Integrating Crowd with Acegi Security](#)
- [3.2.10 Integrating Crowd with a Custom Application](#)

If you have installed Crowd on a new server, or changed Crowd's URL or port number, you will also need to edit the `crowd.properties` file in each integrated application accordingly.

**Troubleshooting**

If you have any problems during upgrade, please raise a support request at [https://support.atlassian.com/](https://support.atlassian.com/) and attach your `atlassian-crowd.log` so we can help you find out what's gone wrong.

**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 1.1 Upgrade Notes**

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

To upgrade to Crowd 1.1.x from 1.0.x or earlier,

- Follow the usual steps for upgrading crowd.
- Configure two additional web applications, as described below.

**Configuring OpenID Server and OpenID Demo Client applications**

In Crowd 1.1, two new web applications have been added to Crowd, along with the Crowd Administration Console and the Demo Application. The new applications are:

<table>
<thead>
<tr>
<th>Application</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenID Server</td>
<td>Note: Logically, the OpenID Server is a client application of the Crowd Server, and must be configured as such. The OpenID Server requires a database. By default, a HSQL database is used.</td>
</tr>
<tr>
<td>OpenID Demo Client</td>
<td>A simple web application which can be used as a starting point to develop OpenID-enabled Java applications. This application is lightweight. It has no persistence store and does not talk to the Crowd Security Server.</td>
</tr>
</tbody>
</table>

Perform the following steps to finish the upgrade:

1. Create a database to house the data specific to the OpenID Server.
2. Point the application context to the new database. The application context for the OpenID Server is in `atlassian-crowd-1.1.0/apache-tomcat-5.5.20/conf/catalina/localhost/openidserver.xml`. More information on how to modify this file for your particular database can be found in [2.2 Connecting CrowdID to a Database](#).
3. Update the `atlassian-crowd-1.1.0/crowd-openidserver-webapp/WEB-INF/classes/jdbc.properties` to reflect the dialect of your database.
4. Update `atlassian-crowd-1.1.0/crowd-openidserver-webapp/WEB-INF/classes/crowd.properties` to use a secure password for the OpenID Server application.
5. Add the application via the Crowd Administration Console. The default name of the application is crowd-openid-server and the password is whatever you specified in `crowd.properties` in the previous step. For more information on how to add an application, see [3.2 Adding an Application](#).
6. Restart the server. This should set up the OpenID Server in Crowd.
Crowd 1.2 Upgrade Notes

Upgrade Procedure

To upgrade to Crowd 1.2.x from 1.1.x or earlier,

• Follow the instructions on upgrading Crowd.

Upgrade Notes

Application Directory Permissions

With Crowd 1.2, directory permissions can now be set at application level. When you upgrade to Crowd 1.2:

• The upgrade procedure will set all application-level permissions equal to your existing directory-level permissions. This means that, for a particular directory, all applications will have the same permissions immediately after the upgrade i.e. the permissions which were set at directory level before the upgrade.
• You can alter the permissions for each application after the upgrade is complete, if you wish.

Developer Notes

SOAP Service API

There are changes to the Crowd API, including new SOAP methods (see CWD-459 and CWD-537), so you should re-generate your WSDL bindings to the Crowd server.
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
- Recovering your Console application password
- Resetting the Domain Cookie Value
- Self Signed Certificate

Integration FAQ
- All Integrations
  - If I delete a principal from Crowd, how will this affect integrated applications?
- Atlassian Product Integration
  - Application Caching
  - JIRA integration
  - Public Signup Setup
- IBM Websphere Integration
Deployment FAQ

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

- Recovering your Console application password
- Resetting the Domain Cookie Value
- Self Signed Certificate
Recovering your Console application password

The Crowd console itself must authenticate to the Crowd framework to perform authentication and authorisation calls.

Like an integrated application, if you have an improper password in the crowd.properties configuration file, the following exception will be thrown when the application attempts to connect to Crowd SOAP services:

  at sun.reflect.NativeConstructorAccessorImpl.newInstance0(Native Method)
  at sun.reflect.NativeConstructorAccessorImpl.newInstance(NativeConstructorAccessorImpl.java:39)
  at sun.reflect.DelegatingConstructorAccessorImpl.newInstance(DelegatingConstructorAccessorImpl.java:27)
  at java.lang.reflect.Constructor.newInstance(Constructor.java:494)
  at org.codehaus.xfire.aegis.type.basic.BeanType.createFromFault(BeanType.java:235)
  at org.codehaus.xfire.aegis.type.basic.BeanType.readObject(BeanType.java:105)
  at org.codehaus.xfire.aegis.AegisBindingProvider.readParameter(AegisBindingProvider.java:49)
  at org.codehaus.xfire.client.ClientFaultConverter.processFaultDetail(ClientFaultConverter.java:51)
  at org.codehaus.xfire.client.ClientFaultConverter.invoke(ClientFaultConverter.java:32)
  at org.codehaus.xfire.handler.HandlerPipeline.invoke(HandlerPipeline.java:131)
  at org.codehaus.xfire.handler.HandlerPipeline.invoke(HandlerPipeline.java:131)
  at org.codehaus.xfire.client.HttpChannel.sendViaClient(HttpChannel.java:139)
  at org.codehaus.xfire.client.HttpChannel.send(HttpChannel.java:48)
  at org.codehaus.xfire.handler.OutMessageSender.invoke(OutMessageSender.java:26)
  at org.codehaus.xfire.handler.HandlerPipeline.invoke(HandlerPipeline.java:131)
  at org.codehaus.xfire.client.Invocation.invoke(Invocation.java:79)
  at org.codehaus.xfire.client.Invocation.invoke(Invocation.java:114)
  at org.codehaus.xfire.client.Invocation.invoke(Invocation.java:336)
  at org.codehaus.xfire.client.XFireProxy.handleRequest(XFireProxy.java:77)
  at org.codehaus.xfire.client.XFireProxy.invoke(XFireProxy.java:57)
  at $Proxy8.authenticateApplication(Unknown Source)
  at com.atlassian.crowd.integration.service.soap.client.GenericClient.authenticate(GenericClient.java:263)
  ... 73 more
  at org.codehaus.xfire.fault.Soap11FaultSerializer.readMessage(Soap11FaultSerializer.java:31)
  at org.codehaus.xfire.fault.SoapFaultSerializer.readMessage(SoapFaultSerializer.java:28)
  at org.codehaus.xfire.soap.handler.ReadHeadersHandler.checkForFault(ReadHeadersHandler.java:111)
  at org.codehaus.xfire.soap.handler.ReadHeadersHandler.invoke(ReadHeadersHandler.java:67)
  at org.codehaus.xfire.handler.HandlerPipeline.invoke(HandlerPipeline.java:131)
  at org.codehaus.xfire.client.onReceive(Client.java:406)
  ... 84 more

If the password for the Crowd console is lost, the only method of recovery is to reset the password in the crowd.properties configuration file to a known application password. To do this you will need to have access to the Crowd database server and run the following commands:

1. Get a list of the applications integrated with Crowd:

```
mysql> select id, name from application;
+--------+---------------------+
| id     | name                |
+--------+---------------------+
|  98305 | crowd               |
|  98306 | demo                |
|  98307 | crowd-openid-server |
| 655361 | jira                |
| 753665 | jiveforums          |
+--------+---------------------+
```

2. Choose an application for which you have the password, and where you're happy to use the same password for the Crowd application. Let's call your application 'X'. Use application X's ID to query the database and retrieve X's credentials:
3. Now query the database for the **ID** of the Crowd application and set Crowd’s application credentials to the **credential** of your application X:

```sql
mysql> update applicationcredentials set credential = 'sQnzu7wkTrgkQZF+0G1hi5AI3Qmzvv0bXgc5THBqi7mAstdd4X1127ASbRt9fEyavWi6m0QP9B81Thf+rDKy8hg==' where applicationid = 98305;
Query OK, 0 rows affected (0.00 sec)
Rows matched: 1  Changed: 0  Warnings: 0
```

4. **Update your** atlassian-crowd-1.1.2/crowd-webapp/WEB-INF/classes/crowd.properties **application.password** value to the value of X’s password.

5. **You may now start Crowd.**

**Further information**

- If you have installed only Crowd and no other integrated applications, you'll need to clear all the database tables (if you've already hooked up to a database server) and re-install Crowd. This should not cause you to lose much data, since no other applications have yet been defined.
- The issue is that the password for the crowd application is being changed during the setup process for crowd. This problem will be resolved with Crowd 1.2 - see CWD-488.
- You may be tempted to try changing the password back to 'password'. Alas, this won't work, because the passwords are encrypted using SHA1.
Resetting the Domain Cookie Value

To reset the SSO (single sign-on) cookie domain, run the following SQL command on the Crowd database:

```
update serverproperty set value = '' where name = 7;
```

Once you have done this you will need to restart Crowd and then log in. This will reset any domain SSO token misconfiguration.
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
Integration FAQ

This page last changed on Sep 17, 2007 by smaddox.

- **All Integrations**
  - If I delete a principal from Crowd, how will this affect integrated applications?
- **Atlassian Product Integration**
  - Application Caching
  - JIRA integration
  - Public Signup Setup
- **IBM Websphere Integration**
All Integrations

This page last changed on Oct 02, 2007 by smaddox.

• If I delete a principal from Crowd, how will this affect integrated applications?
If I delete a principal from Crowd, how will this affect integrated applications?

We recommend that you deactivate a principal rather than deleting them, in case some applications contain historical data, e.g. documents that the user has created.

For example, a principal may be a participant in a JIRA issue. If you remove the principal from the directory managed by Crowd, JIRA will not be able to find the user details when referencing the issue. If you do need to remove the principal from Crowd, you must first remove the user’s involvement in any JIRA issues, as described in the JIRA documentation.

Read more about deleting or deactivating principals in Crowd.
Atlassian Product Integration

This section covers general questions around Crowd's integration with other Atlassian products.

General Integration Questions

Why don't my Groups and Principals show up in Bamboo, Confluence, Fisheye or JIRA?

I want to allow public signups, but don't want 'public' users in my company LDAP repository. How should I configure Crowd?

Confluence Integration

JIRA Integration

What is the difference between JIRA's direct LDAP integration & Crowd's JIRA integration?

If I delete a principal from Crowd, how will this affect JIRA?

Bamboo Integration

Fisheye Integration
Application Caching

When Crowd is deployed into Bamboo, Confluence, Fisheye or JIRA, the Crowd client may be using caching. If you notice that changes made in Crowd do not appear in one of Crowd's configured applications, this will most likely mean that the changes have not yet propagated into the client caches.

For more information, refer to:

- [Caching](#) – information on turning the cache on or off on the Crowd server.
- [Configuring Caching for an Application](#) – information on fine tuning the caching properties of the client application.
What is the difference between JIRA's LDAP integration and Crowd's JIRA integration?

**JIRA's LDAP integration** only delegates authentication to LDAP. This means that you still need to create groups and users in JIRA, and those users must have usernames that match your users in LDAP.

When you use **Crowd's JIRA integration**, all user and group management is delegated to Crowd. This means that you no longer have to create users and groups in JIRA. Crowd gives you access to all these users and groups in your underlying LDAP directories.
Public Signup Setup

This page last changed on Sep 17, 2007 by smaddox.

This tip applies if you:

- Have public-facing JIRA, Confluence and Bamboo servers and private LDAP repositories.
- Allow public signup via JIRA, Confluence and/or Bamboo.
- Want to partition where users are created via the public signup functionality.

Crowd allows for multiple directories to be assigned to an application. Follow these steps to direct all public signups into your chosen Crowd directory:

1. Define two directories:
   a. An internal directory for 'public' users.
   b. An LDAP directory for staff and contractors.
2. Assign both these directories to the 'JIRA' application in Crowd.
3. Use the 'ordering' arrows to move the internal 'public' directory into the first position.

Using this configuration, when Crowd receives a request from JIRA to create a user, Crowd will create the user in the 'public' internal directory only.

In future releases of Crowd, this functionality will be refined so that directory permissions can be set for each application. You will then be able to nominate the directories where users are created on a per-application basis.
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.

----

Some users have also reported errors like the following:

java.lang.VerifyError:
{class: org/codehaus/xfire/aegis/type/basic/ObjectType, method: writeSchema signature: (Lorg/jdom/Element;)V} Incompatible argument to method

This is related to the following XFire issue the suggested fix for this is to upgrade the version of JDOM that is shipped with Websphere to something greater than 1.0 (Websphere ships with JDOM Beta 6).

If you add a later version of JDOM to the WebSphere\AppServer\lib directory and remove the old version, this should fix the above problem.
CrowdID Administration Guide

This page last changed on Aug 09, 2007 by rosie@atlassian.com.

CrowdID is a free add-on that ships with Crowd versions 1.1 and later. It gives administrators a secure way to provide OpenID accounts for their users.

The CrowdID Administration Guide is for people who have CrowdID administration rights. For instructions on using CrowdID to access OpenID-enabled websites, please see the CrowdID User Guide.

Table of Contents

- 1. About CrowdID
  - 1.1 How CrowdID works with Crowd
    - 1.1.1 Determining the name of the CrowdID application
    - 1.1.2 Locating the Crowd Server that CrowdID is using
  - 1.1 How OpenID sites interact with CrowdID
- 2. Allowing users to access CrowdID
  - 2.1 Granting CrowdID access rights to a user
  - 2.2 Granting CrowdID administration rights to a user
- 3. Specifying the sites to which users can login
  - 3.1 Allowing all hosts
  - 3.2 Allowing all except specified hosts ('Blacklist')
  - 3.3 Allowing specified hosts only ('Whitelist')
- 4. Configuring CrowdID system settings
  - 4.1 Specifying the CrowdID URL
  - 4.2 Enabling localhost authentication
  - 4.3 Enabling immediate authentication requests
  - 4.4 Enabling communication with stateless clients
1. About CrowdID

(*CrowdID* is a free add-on that ships with Crowd versions 1.1 and later. It gives administrators a secure way to provide [OpenID](http://openid.net/) accounts for their users.

Crowd is a middleware application that connects web applications (such as CrowdID, JIRA and Confluence) to specified directories (e.g. Microsoft Active Directory, OpenLDAP). For details please see 1.1 Concepts in the Crowd Administration Guide.

- 1.1 How CrowdID works with Crowd
  - 1.1.1 Determining the name of the CrowdID application
  - 1.1.2 Locating the Crowd Server that CrowdID is using
- 1.1 How OpenID sites interact with CrowdID

To access CrowdID, go to [http://localhost:8095/openidserver](http://localhost:8095/openidserver).
1.1 How CrowdID works with Crowd

CrowdID is a free add-on that ships with Crowd versions 1.1 and later. It gives administrators a secure way to provide OpenID accounts for their users.

Crowd is a middleware application that connects web applications (such as CrowdID, JIRA and Confluence) to specified directories (e.g. Microsoft Active Directory, OpenLDAP). For details please see 1.1 Concepts in the Crowd Administration Guide.

This means that:

• CrowdID is a Crowd-connected application.
• CrowdID users are authenticated against Crowd-connected directories.
• If a user has already logged into any other Crowd-connected application (and single sign-on is enabled), they will not be prompted for any further login once they have entered their OpenID URL at an OpenID-enabled website.
• Multiple CrowdID instances can use one Crowd instance. Large organisations often find this useful.

CrowdID is automatically installed when you install Crowd. When you start Crowd for the first time and run the Setup Wizard, you will be offered the option of configuring CrowdID. If you choose not to setup CrowdID at that time, you can always set it up later as described in 4. Configuring CrowdID system settings. Note that you will also need to define the CrowdID application in Crowd, and map it to an appropriate directory — for details please see the Crowd Administration Guide.

To access CrowdID, go to http://localhost:8095/openidserver.

RELATED TOPICS

• 1.1 How CrowdID works with Crowd
  ° 1.1.1 Determining the name of the CrowdID application
  ° 1.1.2 Locating the Crowd Server that CrowdID is using
• 1.1 How OpenID sites interact with CrowdID

Crowd Documentation
1.1.1 Determining the name of the CrowdID application

CrowdID is a Crowd-connected application (for more information please see 3. Managing Applications in the Crowd Administration Guide).

To change the details or users of your CrowdID application within Crowd, you will need to know the name by which your Crowd application is defined in your Crowd server.

To see the name of your CrowdID application,

1. Login to CrowdID.
2. Click the 'Administration' link in the top navigation bar.
3. Click the 'Crowd Server' link in the left navigation column.
4. This will display the 'Crowd Server' details.
   The 'Application Name' field contains the name by which your CrowdID application is known to your Crowd server.

Screenshot: 'Application Name'

RELATED TOPICS

- 1.1 How CrowdID works with Crowd
  - 1.1.1 Determining the name of the CrowdID application
  - 1.1.2 Locating the Crowd Server that CrowdID is using
- 1.1 How OpenID sites interact with CrowdID

Crowd Documentation
1.1.2 Locating the Crowd Server that CrowdID is using

To change the details or users of your CrowdID application within Crowd, you will need to login to your Crowd server.

To determine the location of your Crowd server,

1. Login to CrowdID.
2. Click the 'Administration' link in the top navigation bar.
3. Click the 'Crowd Server' link in the left navigation column.
4. This will display the 'Crowd Server' details.
   The 'Crowd Services' field contains the URL of your Crowd server. Go to this URL to login to Crowd.

Screenshot: 'Crowd Server'

RELATED TOPICS

- 1.1 How CrowdID works with Crowd
  - 1.1.1 Determining the name of the CrowdID application
  - 1.1.2 Locating the Crowd Server that CrowdID is using
- 1.1 How OpenID sites interact with CrowdID

Crowd Documentation
1.1 How OpenID sites interact with CrowdID

This diagram shows how an OpenID-enabled website (known as a 'Relying Party') interacts with CrowdID (an 'OpenID Provider') to validate an end-user's login attempt.

For more information about the OpenID protocol please see http://openid.net.

RELATED TOPICS

- 1.1 How CrowdID works with Crowd
  - 1.1.1 Determining the name of the CrowdID application
  - 1.1.2 Locating the Crowd Server that CrowdID is using
- 1.1 How OpenID sites interact with CrowdID

Crowd Documentation
2. Allowing users to access CrowdID

Granting access to CrowdID is done through Crowd. You can grant people rights to:

- **use CrowdID** — Granting CrowdID access rights to a user allows them to use CrowdID to access OpenID websites and perform all the actions described in the [CrowdID User Guide](#).
- **administer CrowdID** — Granting administration rights to a user allows them to use the 'Administration' menu within CrowdID, which enables them to perform the actions described in the [CrowdID Administration Guide](#).
2.1 Granting CrowdID access rights to a user

Granting CrowdID access rights to a user allows them to use CrowdID to access OpenID websites and perform all the actions described in the CrowdID User Guide. Access to CrowdID is managed via Crowd. A user can only access CrowdID if they belong to a directory that is mapped to the CrowdID application within Crowd.

To grant CrowdID access rights to a particular user,

1. Login to your Crowd server¹.
2. View your CrowdID application² as described in 3.1 Using the Application Browser in the Crowd Administration Guide.
3. Click the 'Directories' tab to see a list of directories that are mapped to your CrowdID application. You will need to add the user to one of these directories.
4. If your directory capabilities permit, add the user to the directory via Crowd as described in 4.02 Adding a Principal in the Crowd Administration Guide. (Otherwise you may need to use your specific directory-management tool, instead of Crowd, to add the user to the directory.)

To grant CrowdID access rights to all the users in a particular directory,

1. Login to your Crowd server¹.
2. Map the directory to your CrowdID application² as described in 3.3 Mapping a Directory to an Application in the Crowd Administration Guide.

To grant CrowdID access rights to a particular group of users within a directory,

1. Login to your Crowd server¹.
2. Map the group to your CrowdID application² as described in 3.4 Specifying which Groups can access an Application in the Crowd Administration Guide.

¹ To find your Crowd server's URL, see 1.1.2 Locating the Crowd Server that CrowdID is using.
² To identify the name by which your CrowdID application is known within Crowd, see 1.1.1 Determining the name of the CrowdID application.

RELATED TOPICS

• 2.1 Granting CrowdID access rights to a user
• 2.2 Granting CrowdID administration rights to a user

Crowd Documentation

RELATED TOPICS

• 2.1 Granting CrowdID access rights to a user
• 2.2 Granting CrowdID administration rights to a user

Crowd Documentation
2.2 Granting CrowdID administration rights to a user

Granting administration rights to a user allows them to use the 'Administration' menu within CrowdID, which enables them to perform the actions described in the CrowdID Administration Guide. CrowdID administration rights are managed via Crowd. To grant administration rights to a user, you need to add them to the 'crowd-administrators' group as described below. Note:

- adding a user to the 'crowd-administrators' group will also confer Crowd administration rights (unless you choose to use a different group to contain Crowd administrators). See 4.08 Granting Crowd Administration Rights to a User in the Crowd Administration Guide.
- the 'crowd-administrators' group always contains CrowdID administrators, regardless of whether or not you are using it to contain Crowd administrators.

To grant administration rights to a user,

1. Login to your Crowd server¹.
2. Click the 'Principals' link in the top navigation bar.
3. This will display the Principal Browser (in Crowd, users are known as 'principals'). Select the directory that contains the principal to whom you wish to grant administration rights.
4. Locate the principal, and click the 'View' link that corresponds to the principal.
5. This will display the 'Principal Details' screen. Click the 'Groups' tab.
6. A list of the principal's current groups (if any) will be displayed. Select the 'crowd-administrators' group from the drop-down box below the list, then click the 'Add' button.

¹ To find your Crowd server's URL, see 1.1.2 Locating the Crowd Server that CrowdID is using.

RELATED TOPICS

- 2.1 Granting CrowdID access rights to a user
- 2.2 Granting CrowdID administration rights to a user

Crowd Documentation
3. Specifying the sites to which users can login

There are three ways to specify which OpenID hosts (i.e. websites or IP addresses) your users can login to using their CrowdID:

- **No restriction** — your CrowdID users can login to any OpenID host
- **Blacklist** — your CrowdID users can login to any OpenID host except the one(s) that you specify
- **Whitelist** — your CrowdID users can login to only those OpenID host(s) that you specify
3.1 Allowing all hosts

There are three ways to specify which OpenID hosts (i.e. websites or IP addresses) your users can login to using their CrowdID:

- **No restriction** — your CrowdID users can login to any OpenID host
- **Blacklist** — your CrowdID users can login to any OpenID host except the one(s) that you specify
- **Whitelist** — your CrowdID users can login to only those OpenID host(s) that you specify

To allow users to login to any OpenID host,

1. Login to CrowdID.
2. Click the 'Administration' link in the top navigation bar.
3. Click the 'Trust Relationships' link in the left navigation column.
4. For 'Restriction Type', select 'None'.

Screenshot: 'Restriction Type — None'

**RELATED TOPICS**

- [3.1 Allowing all hosts](#)
- [3.2 Allowing all except specified hosts (‘Blacklist’)](#)
- [3.3 Allowing specified hosts only (‘Whitelist’)](#)

Crowd Documentation
3.2 Allowing all except specified hosts ('Blacklist')

There are three ways to specify which OpenID hosts (i.e. websites or IP addresses) your users can login to using their CrowdID:

- **No restriction** — your CrowdID users can login to any OpenID host
- **Blacklist** — your CrowdID users can login to any OpenID host except the one(s) that you specify
- **Whitelist** — your CrowdID users can login to only those OpenID host(s) that you specify

To specify an OpenID blacklist,

1. Login to CrowdID.
2. Click the 'Administration' link in the top navigation bar.
3. Click the 'Trust Relationships' link in the left navigation column.
4. For 'Restriction Type', select 'Blacklist'.
5. Wait for a section titled 'Blacklist mode: hosts that can not login' to appear on the screen.
6. For each site to which you want to prevent users logging in,
   a. Type the URL or IP address in the 'Address' field.
   b. Click the 'Add' button.

Screenshot: 'Restriction Type — Blacklist'

---

Related Topics

- [3.1 Allowing all hosts](#)
- [3.2 Allowing all except specified hosts ('Blacklist')](#)
- [3.3 Allowing specified hosts only ('Whitelist')](#)

Crowd Documentation
3.3 Allowing specified hosts only (‘Whitelist’)

There are three ways to specify which OpenID hosts (i.e. websites or IP addresses) your users can login to using their CrowdID:

- **No restriction** — your CrowdID users can login to any OpenID host
- **Blacklist** — your CrowdID users can login to any OpenID host except the one(s) that you specify
- **Whitelist** — your CrowdID users can login to only those OpenID host(s) that you specify

To specify an OpenID whitelist,

1. Login to CrowdID.
2. Click the ‘Administration’ link in the top navigation bar.
3. Click the ‘Trust Relationships’ link in the left navigation column.
4. For ‘Restriction Type’, select ‘Blacklist’.
5. Wait for a section titled ‘Whitelist mode: hosts that can login’ to appear on the screen.
6. For each site to which you want to allow users to login,
   a. Type the URL or IP address in the ‘Address’ field.
   b. Click the ‘Add’ button.

Screenshot: ‘Restriction Type — Whitelist’

### RELATED TOPICS

- [3.1 Allowing all hosts](#)
- [3.2 Allowing all except specified hosts (‘Blacklist’)](#)
- [3.3 Allowing specified hosts only (‘Whitelist’)](#)
4. Configuring CrowdID system settings

This page last changed on Jun 13, 2007 by rosie@atlassian.com.

- 4.1 Specifying the CrowdID URL
- 4.2 Enabling localhost authentication
- 4.3 Enabling immediate authentication requests
- 4.4 Enabling communication with stateless clients
4.1 Specifying the CrowdID URL

The CrowdID URL is the URL that your end-users will type when logging into OpenID-enabled websites. To define the URL of your CrowdID instance,

1. Login to CrowdID.
2. Click the 'Administration' link in the top navigation bar.
3. Click the 'General Configuration' link in the left navigation column.
4. Type the URL into the 'Base URL' field.
5. Click the 'Update' button.

Screenshot: 'General Configuration'

RELATED TOPICS

- 4.1 Specifying the CrowdID URL
- 4.2 Enabling localhost authentication
- 4.3 Enabling immediate authentication requests
- 4.4 Enabling communication with stateless clients

Crowd Documentation
4.2 Enabling localhost authentication

Enabling localhost authentication prevents OpenID-enabled sites from directly accessing your end-users' local machines.

To enable localhost authentication,

1. Login to CrowdID.
2. Click the 'Administration' link in the top navigation bar.
3. Click the 'General Configuration' link in the left navigation column.
4. Select the 'Allow localhost authentications' checkbox.
5. Click the 'Update' button.

Screenshot: 'General Configuration'

**RELATED TOPICS**

- 4.1 Specifying the CrowdID URL
- 4.2 Enabling localhost authentication
- 4.3 Enabling immediate authentication requests
- 4.4 Enabling communication with stateless clients

Crowd Documentation

**RELATED TOPICS**

- 4.1 Specifying the CrowdID URL
- 4.2 Enabling localhost authentication
- 4.3 Enabling immediate authentication requests
- 4.4 Enabling communication with stateless clients

Crowd Documentation
4.3 Enabling immediate authentication requests

Enabling 'Allow immediate authentication requests' allows an OpenID-enabled site to check whether the user is logged in, without actually prompting the user to login. Known as pass-through authentication, this provides greater convenience for end-users, particularly when an end-user visits a site for which they have previously selected 'Allow Always' (see 2.4 Allowing or denying a login in the CrowdID User Guide).

To enable 'Allow immediate authentication requests',

1. Login to CrowdID.
2. Click the 'Administration' link in the top navigation bar.
3. Click the 'General Configuration' link in the left navigation column.
4. Select the 'Allow immediate authentication requests' checkbox.
5. Click the 'Update' button.

Screenshot: 'General Configuration'

RELATED TOPICS

- 4.1 Specifying the CrowdID URL
- 4.2 Enabling localhost authentication
- 4.3 Enabling immediate authentication requests
- 4.4 Enabling communication with stateless clients

Crowd Documentation
4.4 Enabling communication with stateless clients

This page last changed on Jun 20, 2007 by shamid.

Some OpenID-enabled sites do not support pre-shared secrets (associations). Selecting allow stateless clients enables your CrowdID server to communicate with such sites.

To allow stateless clients,

1. Login to CrowdID.
2. Click the 'Administration' link in the top navigation bar.
3. Click the 'General Configuration' link in the left navigation column.
4. Select the 'Allow stateless clients' checkbox.
5. Click the 'Update' button.

Screenshot: 'General Configuration'

RELATED TOPICS

- 4.1 Specifying the CrowdID URL
- 4.2 Enabling localhost authentication
- 4.3 Enabling immediate authentication requests
- 4.4 Enabling communication with stateless clients
With Crowd comes CrowdID, your OpenID provider.

CrowdID is an Atlassian product which allows you to use a single login for all OpenID-enabled websites. This means that you don't have to remember a separate username and password for each different site that you visit. You can just use your OpenID for all of them.

You can use CrowdID if your administrator has installed it for your organisation. For instructions on setting up CrowdID, please see the CrowdID Administration Guide. The CrowdID User Guide tells you how to

• Log in to websites using CrowdID.
• Instruct CrowdID to always allow login to a specific site.
• Set up your own profile(s) within CrowdID.
• Use CrowdID to change your password.

Contents of the CrowdID User Guide

• 1. Getting started with CrowdID
  ° 1.1 What is OpenID?
  ° 1.2 What is CrowdID?
  ° 1.3 What is an OpenID URL or identifier?
  ° 1.4 Viewing the CrowdID page
• 2. Logging in to a website using OpenID
  ° 2.1 Does the website support OpenID?
  ° 2.2 Entering your OpenID URL
  ° 2.3 Logging in to CrowdID
  ° 2.4 Allowing or denying a login
  ° 2.5 Providing additional profile information to a website
• 3. Viewing your always-approved websites
• 4. Viewing your login history
• 5. Updating your profile
• 6. Using more than one profile
  ° 6.1 Adding a profile
  ° 6.2 Choosing a profile for a website
  ° 6.3 Setting a default profile
  ° 6.4 Deleting a profile
• 7. Changing or resetting your password
  ° 7.1 Changing your password
  ° 7.2 Resetting your password
1. Getting started with CrowdID

CrowdID is an Atlassian product which allows you to use a single login for all OpenID-enabled websites. This means that you don't have to remember a separate username and password for each different site that you visit. You can just use your OpenID for all of them.

You can use CrowdID if your administrator has installed it for your organisation.

- 1.1 What is OpenID?
- 1.2 What is CrowdID?
- 1.3 What is an OpenID URL or identifier?
- 1.4 Viewing the CrowdID page
1.1 What is OpenID?

This page last changed on Jun 19, 2007 by smaddox.

The term 'OpenID' has two meanings:

- The OpenID protocol, described below.
- Your own identifier or URL.

OpenID is an open, free protocol which allows you to use a single identifier to login to any OpenID-enabled website. OpenID allows the website to communicate with your OpenID provider (e.g. your organisation's CrowdID server) when attempting to verify your login.

✅ Do you have a zillion usernames and passwords, which you use for logging in to blogs and websites all over the place? OpenID allows you to throw them all away, for all websites that support it. More and more sites are coming on board.

RELATED TOPICS

- 1.1 What is OpenID?
- 1.2 What is CrowdID?
- 1.3 What is an OpenID URL or identifier?
- 1.4 Viewing the CrowdID page

CrowdID User Guide
1.2 What is CrowdID?

CrowdID is an Atlassian product which makes use of the OpenID protocol to allow you to use a single login for a number of websites. To put it another way: CrowdID is an 'OpenID provider'. You can use CrowdID if your administrator has installed it for your organisation.

This means that you can:

- Securely store your username and password on your organisation's server.
- Use your OpenID as a single identifier to log in to all websites which support OpenID.
- Control how you allow or deny login requests from websites.

☑️ Your organisation can use CrowdID to set up an internal OpenID provider. There are also other OpenID providers, where you can get a free OpenID.

RELATED TOPICS

- 1.1 What is OpenID?
- 1.2 What is CrowdID?
- 1.3 What is an OpenID URL or identifier?
- 1.4 Viewing the CrowdID page

CrowdID User Guide
1.3 What is an OpenID URL or identifier?

To log in to an OpenID-enabled website you need an OpenID identifier, also called an OpenID URL or simply an OpenID. Your OpenID is a URL (web address) which points to your organisation's CrowdID server. Here are some examples of what your OpenID may look like:

http://my.server.name/mynname
http://myname.mysite.com

To find your OpenID URL, you can:

- Ask your system administrator, or
- Click the 'My OpenID' link on the 'Home' tab of the CrowdID page.

RELATED TOPICS

- 1.1 What is OpenID?
- 1.2 What is CrowdID?
- 1.3 What is an OpenID URL or identifier?
- 1.4 Viewing the CrowdID page
1.4 Viewing the CrowdID page

This page last changed on Jun 19, 2007 by smaddox.

The CrowdID page allows you to:

- View your OpenID URL.
- Set up your profile(s).
- View your list of always-approved sites.
- View your login history.
- Resume approval of a login. (This option appears only during a login process, if you move away from the ‘OpenID Verification’ page.)
- Change your password.

There are two ways to access the CrowdID page:

- While you are logging in to another site.
- Directly via the CrowdID URL.

To access the CrowdID page while you are logging in to another site,

1. Use your OpenID to log in to the website you want to visit.
2. Log in to CrowdID if prompted.
3. The CrowdID 'OpenID Verification' page will appear, provided that you have not previously added the website to your list of always-approved sites. You can choose any of the CrowdID options on the left-hand navigation panel, even during the login process.
4. When you have finished your tasks in CrowdID, you can resume the login.

To access CrowdID directly via the CrowdID URL,

1. Ask your administrator for the CrowdID address (URL) as configured for your organisation.
2. Type or paste the address into the address or navigation bar of your internet browser.
3. The CrowdID Login page will appear. Type in your username and password.
4. Click the 'Login' button.
5. The CrowdID 'My OpenID' page will appear. The CrowdID options are displayed in the left-hand navigation panel and top menu bar.

Screenshot: CrowdID My OpenID page

RELATED TOPICS

- 1.1 What is OpenID?
- 1.2 What is CrowdID?
- 1.3 What is an OpenID URL or identifier?
- 1.4 Viewing the CrowdID page
2. Logging in to a website using OpenID

CrowdID enables you to log in to a website using your OpenID. The login process depends upon the following:

- Have you logged in to CrowdID already during this browser session?
- Have you previously added the website to your list of always-approved sites?
- Does the website you are visiting require additional profile information?

Steps in the login process:

1. Find the OpenID login page or section on the website you want to visit.
2. Enter your OpenID and click the login button.
3. If prompted, log in to CrowdID. (Required if you have not already logged in during this browser session.)
4. If prompted, instruct CrowdID to allow the website login. (Required if you have not previously added the website to your list of always-approved sites.)
5. If prompted, supply additional profile information. (Required if the website you are visiting wants more information.)

The login process can be very simple: just the first two steps above, provided that you have already logged in to CrowdID this session and have already added the website to your list of always-approved sites.
2.1 Does the website support OpenID?

You can only use your OpenID (also called an OpenID URL or identifier) to log in to a website if the site supports the OpenID protocol. The number of websites that support OpenID is growing rapidly.

To see if a particular website supports OpenID, check the site's login page for one or more of the following:

- The word 'OpenID'.
- The OpenID logo

RELATED TOPICS

- 2.1 Does the website support OpenID?
- 2.2 Entering your OpenID URL
- 2.3 Logging in to CrowdID
- 2.4 Allowing or denying a login
- 2.5 Providing additional profile information to a website

CrowdID User Guide
2.2 Entering your OpenID URL

With CrowdID, you can use your 'OpenID' (also called an OpenID URL or identifier) to log in to a website that supports the OpenID protocol.

To log in to a website which supports OpenID,

1. Go to the login page of the website you want to visit.
2. Look for the OpenID login section.

   Sometimes the OpenID login will be on the same page as the standard login. Other sites will have a separate OpenID login page.
3. Type or paste your OpenID into the login text box.

   Usually, you must enter the full OpenID. In some sites, you can enter the OpenID without 'http://'
4. Click the login button. The button will probably be labelled 'Log in', 'Sign in' or 'Go'.

One of the following things will happen now:

• If you have not already logged in to CrowdID during this browser session, you will see the CrowdID login page.
• If you have already logged in to CrowdID and you have previously instructed CrowdID to allow this website always, then you will be logged straight into the website.
• If you have already logged in to CrowdID but have not previously set this site to "Allow Always", then CrowdID will ask you to approve the login.
• If your administrator has blocked access to this website, CrowdID will display an 'OpenID Verification Error' message.

RELATED TOPICS

• 2.1 Does the website support OpenID?
• 2.2 Entering your OpenID URL
• 2.3 Logging in to CrowdID
• 2.4 Allowing or denying a login
• 2.5 Providing additional profile information to a website

CrowdID User Guide
2.3 Logging in to CrowdID

CrowdID will ask you to log in, if you have not already done so during this browser session or if your session has timed out. The CrowdID login may appear during the process of logging in to another website, or when you are accessing CrowdID directly.

To log in to CrowdID,

1. Type in your username and password.
2. Click the 'Login' button.

You can reset your password, if you have forgotten it.

Screenshot: CrowdID login page

If you are in the process of logging in to another website, CrowdID will now ask you to approve the login.

RELATED TOPICS

- 2.1 Does the website support OpenID?
- 2.2 Entering your OpenID URL
- 2.3 Logging in to CrowdID
- 2.4 Allowing or denying a login
- 2.5 Providing additional profile information to a website

CrowdID User Guide
2.4 Allowing or denying a login

When you use your OpenID to log in to a website, CrowdID will present the 'OpenID Verification' page where you can allow or deny the login.

- If you have previously instructed CrowdID to allow this site always, you will not see this page. You can remove a site from the 'Allow Always' list in CrowdID.

You can instruct CrowdID to:

- **Allow the login** for this session only ('Allow Once').
- **Allow login** to this site every time you use your OpenID ('Allow Always').
- **Refuse login** to this site ('Deny').
- **Use a specific profile**.

If you move away from the 'OpenID Verification' page within CrowdID, you can go back to the page and resume approval.

**Screenshot: OpenID Verification page**

To allow the login for this session only,

1. Click 'Allow Once' on the right of the CrowdID 'OpenID Verification' page.
2. CrowdID will send you back to the original site, passing your profile information as well as the confirmed login. The website you are visiting may ask you to complete your profile information.

To allow login to this site every time you use your OpenID,

1. Click 'Allow Always' on the right of the CrowdID 'OpenID Verification' page.
2. CrowdID will add the website to your list of approved sites and send you back to the original site, passing your profile information as well as the confirmed login. The website you are visiting may ask you to complete your profile information.

To refuse login to this site,
1. Click 'Deny' on the right of the CrowdID 'OpenID Verification' page.
2. CrowdID will send you back to the original site and refuse the login. The original site will probably show a message something like 'Verification cancelled'.

To use a specific profile,

1. If you have defined more than one profile, you can choose a specific profile for the website you are visiting. Select a profile from the dropdown list labelled 'Use this profile' on the CrowdID 'OpenID Verification' page.
2. The profile details will change in the 'Select Profile' section of the page. CrowdID will pass these profile details to the website when you allow the login.

To go back to the 'OpenID Verification' page and resume approval,

1. Click 'Resume Approval' in the left-hand navigation panel. This option will appear if you move away from the 'OpenID Verification' page during the login process.
2. CrowdID will return to the 'OpenID Verification' page, where you can allow the login.

RELATED TOPICS

- 2.1 Does the website support OpenID?
- 2.2 Entering your OpenID URL
- 2.3 Logging in to CrowdID
- 2.4 Allowing or denying a login
- 2.5 Providing additional profile information to a website

CrowdID User Guide
2.5 Providing additional profile information to a website

When you log in to a website using your OpenID, CrowdID passes your profile information to the website. Some websites will then log you in immediately, while other websites may ask you to confirm or complete the profile information.

You are now outside CrowdID. Any dialogue here is between you and the website you are visiting.

To provide additional profile information to a website,

1. Check the profile information displayed, and add extra information as you wish.
2. Click the button or other option supplied by the website to complete the login process.

You can change your profile information and define more than one profile in CrowdID.

RELATED TOPICS

- 2.1 Does the website support OpenID?
- 2.2 Entering your OpenID URL
- 2.3 Logging in to CrowdID
- 2.4 Allowing or denying a login
- 2.5 Providing additional profile information to a website

CrowdID User Guide
3. Viewing your always-approved websites

When logging in to a website, you can instruct CrowdID to allow login to the site every time you use your OpenID ('Allow Always').

The CrowdID 'Approved Sites' page allows you to:

- **View your list** of always-approved sites.
- **Remove a site** from the list.
- **Choose a profile** for use when logging in to a site.

- **Tip**: If you have never instructed CrowdID to 'Allow Always' for any sites, The 'Approved Sites' page will display a message like 'You currently have no approved sites.'
- **Tip**: You can add profiles on the CrowdID 'Profiles' page.

To view your list of always-approved sites,

1. **Access CrowdID**.
2. Click 'Approved Sites' in the left-hand navigation panel.

To remove a site from the list,

1. **Access CrowdID**.
2. Click 'Approved Sites' in the left-hand navigation panel.
3. Your list of always-approved sites will appear. Click the remove button next to the site which you want to remove.
4. Click the 'Apply' button.
5. 'Update Successful' message is displayed.

- **Tip**: If you do not click the 'Apply' button, your changes will be cancelled.

To choose a profile for use when logging in to a site,

1. **Access CrowdID**.
2. Click 'Approved Sites' in the left-hand navigation panel.
3. Your list of always-approved sites will appear. Select the profile you want from the dropdown list next to the applicable site.
4. Click the 'Apply' button.
5. 'Update Successful' message is displayed.

- **Tip**: If you do not click the 'Apply' button, your changes will be cancelled.

**Screenshot: CrowdID Approved Sites page**
RELATED TOPICS

- 1. Getting started with CrowdID
- 2. Logging in to a website using OpenID
- 3. Viewing your always-approved websites
- 4. Viewing your login history
- 5. Updating your profile
- 6. Using more than one profile
- 7. Changing or resetting your password
4. Viewing your login history

The CrowdID 'Login History' page displays a list of the sites you have visited and the type of approval you gave on each visit:

- 'Allow Always' - At the time of this login, you instructed CrowdID to allow login to the site every time you use your OpenID.
- '(Auto) Allow Always' - This login was allowed automatically, because you have previously instructed CrowdID to allow login to the site every time you use your OpenID.
- 'Allow Once' - You instructed CrowdID to allow login to the site at that time only.
- 'Deny' - You instructed CrowdID to refuse the login to the site at that time.

To view your login history,

1. Access CrowdID.
2. Click 'Login History' in the left-hand navigation panel.

If you have used your OpenID many times, the login history items will be shown on more than one page. To move from one page to another, click the page numbers or the 'Next' and 'Prev' links at the bottom of the page.

Screenshot: CrowdID Login History page

RELATED TOPICS

- 1. Getting started with CrowdID
- 2. Logging in to a website using OpenID
- 3. Viewing your always-approved websites
- 4. Viewing your login history
• 5. Updating your profile
• 6. Using more than one profile
• 7. Changing or resetting your password
5. Updating your profile

When you log in to a website using your OpenID, CrowdID will pass some information to the website. The information is copied from your profile on CrowdID. When your profile is first created, CrowdID will auto-fill the information where possible, by copying:

- Country and language from the language information in your browser.
- Name and email address from your organisation's user directory.

You can update your profile information on CrowdID, as described below.

You can also:

- Add a new profile.
- Choose a profile for a website.
- Set a profile as default.
- Delete a profile.

To update your profile,

1. Access CrowdID.
2. Click 'Profiles' in the left-hand navigation panel.
3. Select the required profile from the 'Profile' dropdown list, if you have more than one profile.
4. Update the profile details then click the 'Save' button.
5. 'Profile updated' message is displayed at the top of the page.

Screenshot: CrowdID Profiles page

RELATED TOPICS

- 1. Getting started with CrowdID
- 2. Logging in to a website using OpenID
- 3. Viewing your always-approved websites
• 4. Viewing your login history
• 5. Updating your profile
• 6. Using more than one profile
• 7. Changing or resetting your password
6. Using more than one profile

This page last changed on Jun 18, 2007 by smaddox.

You can create multiple profiles in CrowdID and then allocate specific profiles to specific websites.

- 6.1 Adding a profile
- 6.2 Choosing a profile for a website
- 6.3 Setting a default profile
- 6.4 Deleting a profile
6.1 Adding a profile

When you log in to a website using your OpenID, CrowdID will pass some information to the website. The information is copied from your profile on CrowdID. When your profile is first created, CrowdID will auto-fill the information where possible, by copying:

- Country and language from the language information in your browser.
- Name and email address from your organisation's user directory.

To add a profile,

1. **Access CrowdID**.
2. Click 'Profiles' in the left-hand navigation panel.
3. Select '--- Create New Profile --' from the 'Profile' dropdown list.
4. CrowdID will auto-fill the information where possible. Update the profile details then click the 'Save' button.
5. 'Profile updated' message is displayed at the top of the page.

Screenshot: CrowdID adding a profile

**RELATED TOPICS**

- 6.1 Adding a profile
- 6.2 Choosing a profile for a website
- 6.3 Setting a default profile
- 6.4 Deleting a profile

**CrowdID User Guide**
6.2 Choosing a profile for a website

You can choose a specific profile for use when logging in to a website. There are different ways to choose a profile:

- Choose a profile for a specific login, during the login process. You can do this for sites which you have not set to 'Allow Always'.
- Choose a profile for a specific website, on the CrowdID 'Approved Sites' page. You can do this for sites which you have set to 'Allow Always'.
- Set your default profile on the CrowdID 'Profiles' page.

RELATED TOPICS

- 6.1 Adding a profile
- 6.2 Choosing a profile for a website
- 6.3 Setting a default profile
- 6.4 Deleting a profile

CrowdID User Guide
6.3 Setting a default profile

This page last changed on Jun 20, 2007 by shamid.

If you have more than one profile, you can choose one of them as default.

Effect of the 'default' profile when you are logging in to a website:

- If you have never logged in to the website before or have previously allowed or denied authentication to that site, the default profile will be pre-selected. You can still choose a different profile during the login.
- If you have set the website to 'Always Allow', CrowdID will use the profile selected for the site on the Approved Sites page.

To set a default profile,

1. Access CrowdID.
2. Click 'Profiles' in the left-hand navigation panel.
3. Select the required profile in the 'Profile' dropdown list
4. Click the 'Make Default' link next to the 'Profile' dropdown list.
   - The 'Make Default' link does not appear if the selected profile is already the default.
5. The word '(default)' appears next to the profile name in the dropdown list.

Screenshot: CrowdID setting a default profile

 RELATED TOPICS

- 6.1 Adding a profile
- 6.2 Choosing a profile for a website
- 6.3 Setting a default profile
- 6.4 Deleting a profile

CrowdID User Guide
6.4 Deleting a profile

This page last changed on Jun 19, 2007 by smaddox.

You can delete one of your profiles on CrowdID, provided that it is not your default profile.

To delete a profile,

1. Access CrowdID.
2. Click ‘Profiles’ in the left-hand navigation panel.
3. Select the required profile in the ‘Profile’ dropdown list
4. Click the ‘Delete’ button.
5. ‘Profile deleted’ message is displayed at the top of the page.

If you delete a profile which is linked to one or more of your always-approved websites, CrowdID will remove the affected website(s) from the list.

Screenshot: CrowdID profiles page

RELATED TOPICS

- 6.1 Adding a profile
- 6.2 Choosing a profile for a website
- 6.3 Setting a default profile
- 6.4 Deleting a profile

CrowdID User Guide
7. Changing or resetting your password

If your administrator has allowed it, you can use CrowdID to change your password across all Crowd applications. Note that you will need to be logged in to Crowd before you can do this.

When attempting to log in to Crowd, you can also reset your password, if you have forgotten the old one. A new password will be emailed to you.
7.1 Changing your password

The CrowdID 'Change Your Password' page allows you to change your password across all applications in your organisation, provided that the application is linked to Crowd.

Note:

- Crowd will attempt to change your password in all the user directories linked to Crowd. This will be successful where the directory allows it.
- Your administrator may disable password-change via CrowdID. In that case, you will receive an error message when you apply the change.

To change your password,

1. Access CrowdID.
2. Click 'Change Password' in the top menu bar.
3. The 'Change Your Password' page will appear. Type in your old password once, and the new password twice.
4. Click the 'Update' button.
5. 'Password updated' message is displayed.

If the change is successful, your password may also have changed in other Crowd-connected applications.

Screenshot: CrowdID Change Your Password page

RELATED TOPICS

- 1. Getting started with CrowdID
- 2. Logging in to a website using OpenID
- 3. Viewing your always-approved websites
- 4. Viewing your login history
- 5. Updating your profile
- 6. Using more than one profile
- 7. Changing or resetting your password
7.2 Resetting your password

The CrowdID 'Login' page allows you to reset your password, which is useful when you have forgotten the password.

This will reset your password across all applications in your organisation, provided that the application is linked to Crowd.

To reset your password,

1. Access CrowdID.
2. Click the 'Forgotten your password?' link on the CrowdID Login page.
3. The 'Reset Your Password' page will appear. Type in your Crowd username and click the 'Continue' button.
4. A message will appear: 'Your new password is on the way!'. Click the 'Home' link at the top of the page.
5. You will receive an email message with your new password. Copy the password.
6. Log in to CrowdID using the new password.
7. Change your password to one you can remember easily.

If the change is successful, your password may also have changed in other Crowd-connected applications.

Screenshot: CrowdID Reset Your Password page

 RELATED TOPICS

- 1. Getting started with CrowdID
- 2. Logging in to a website using OpenID
- 3. Viewing your always-approved websites
- 4. Viewing your login history
- 5. Updating your profile
- 6. Using more than one profile
- 7. Changing or resetting your password
Crowd 1.2 has now been released — see the Crowd 1.2 Release Notes.