*A project report on*

# IDENTITY ACCESS MANAGEMENT

*Submitted in partial fulfillment for the award of the degree of*

**Bachelor of Technology (B. Tech)**

*by*

**CHIRAG GAUR**

**18BEC0021**



**SCHOOL OF ELECTRONICS ENGINEERING**

April, 2022

**DECLARATION**

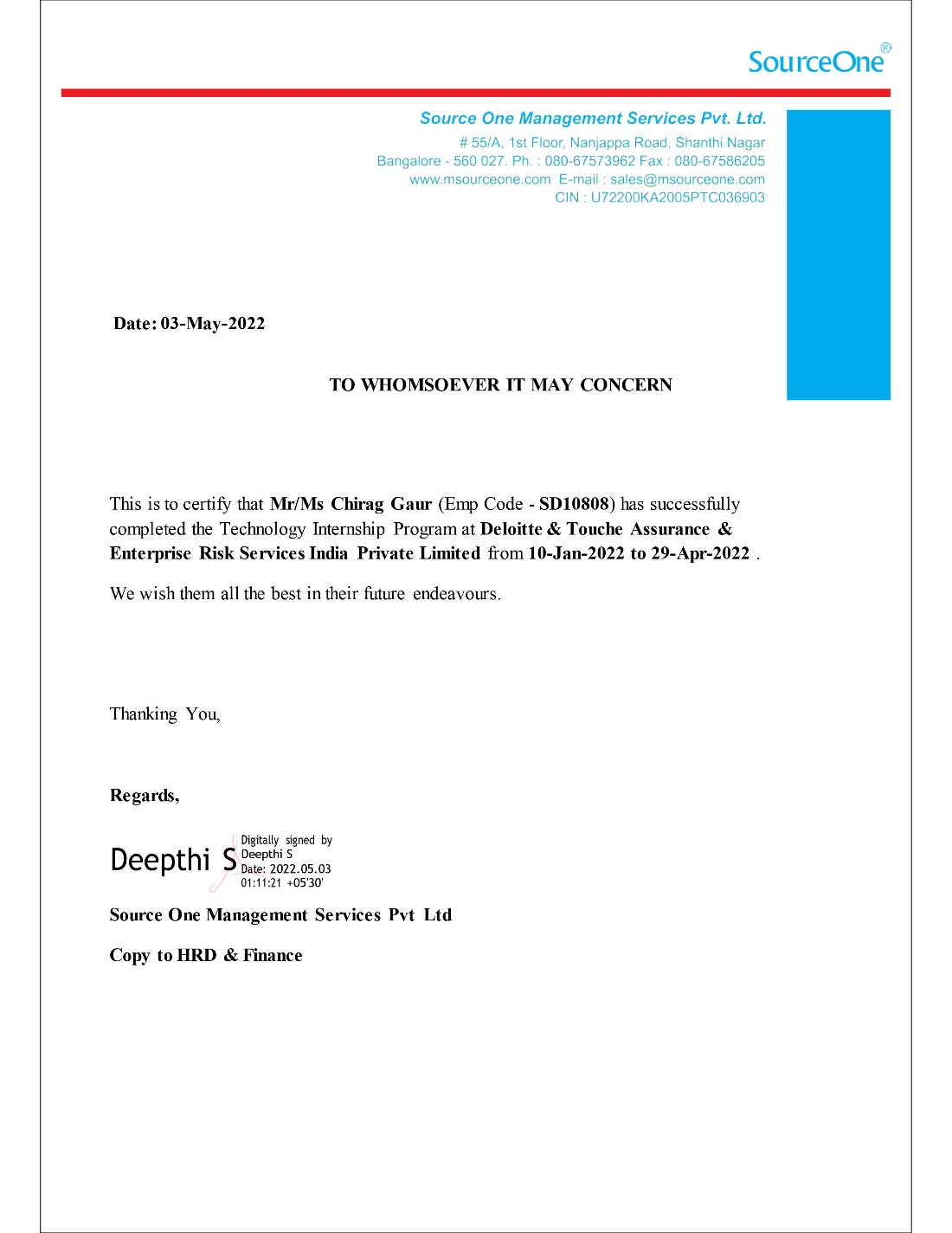
I hereby declare that the thesis entitled “Identity Access Management” submitted by me, for the award of the degree of *Bachelor of Technology in Electronics and Communication Engineering* to VIT is a record of bonafide work carried out by me under the supervision of Placement Office.

I further declare that the work reported in this thesis has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Place: Vellore Chirag Gaur

Date: 22-04-2022 **Signature of the Candidate**

**INTERNSHIP COMPLETION CERTIFICATE**



**ACKNOWLEDGEMENT**

It is my pleasure to express with deep sense of gratitude to PAT Office, Vellore Institute of Technology for their constant guidance and continual encouragement.

I would like to express my gratitude to DEAN of SENSE, Vellore Institute of Technology for providing with an environment to work in and for his inspiration during the tenure of the course.

In jubilant mood I express ingeniously my whole-hearted thanks to HOD SENSE, Dr. Prakasm P, all teaching staff and members working as limbs of our university for their not-self-centered enthusiasm coupled with timely encouragements showered on me with zeal, which prompted the acquirement of the requisite knowledge to finalize my course study successfully. I would like to thank my parents for their support.

It is indeed a pleasure to thank my friends who persuaded and encouraged me to take up and complete this task. At last, but not least, I express my gratitude and appreciation to all those who have helped me directly or indirectly toward the successful completion of this project.

Place: Vellore

Date: 22-04-2022  **Chirag Gaur**

**18BEC0021**

**EXECUTIVE SUMMARY**

The purpose of this thesis is to serve as a record for the work done during my internship at the Hyderabad office of Deloitte USI from **January 10, 2022** to **April 29, 2022**. This internship was done not only as the Capstone Project to complete my bachelor’s degree, but also to gain an insight into how it is to be working in and making contributions to an organization of this stature.

Identity Access Management and Web Access Management are the focal points of this project. My experience at Deloitte has taught me not just about my management and technical duties, but also about working as part of a team, managing time, and coming up with creative solutions to unanticipated challenges.

Main focus of this internship was to understand the concept of Web Access Management and implement PingIdentity products to configure real life use cases using industry standards protocol such as SAML 2.0, OAuth and OpenID for Single Sign-On (SSO) and Federations.

Following tools of PingIdentity were used during my Internship tenure:

1. PingDirectory
2. PingFederate
3. PingAccess

All the above tools and relevant use-cases has been explained in detail in the later sections. The insights gained during the internship have helped me to put my knowledge and skills to work on relevant use cases in the domain of IAM and WAM. Identity management solutions such as SailPoint and PingIdentity are used to demonstrate key aspects of cybersecurity i.e., authentication and authorization. After reviewing several case-studies on cybersecurity incidents, I discovered that lots of enterprises leave vulnerabilities in their systems that can lead to cyberattack. The above mentioned IAM solutions can be used to prevent cyber hacks. Apart from technical skills I have also learnt how to communicate and build relationships with the people I worked with.

## **TABLE OF CONTENTS**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **CONTENTS** | | **Page No.** |
|  | **Internship Completion Certificate** | | 3 |
|  | **Acknowledgement** | | 4 |
|  | **Executive Summary** | | 5 |
|  | **Table of Contents** | | 6-7 |
|  | **List of Figures** | | 8 |
|  | **List of Tables** | | 9 |
|  | **List of Acronyms** | | 10 |
| **1.** | **INTRODUCTION** | |  |
|  | 1.1 | Background | 11 |
|  | 1.2 | Motivation | 11 |
| **2.** | **IDENTITY ACCESS MANAGEMENT** | |  |
|  | 2.1 What is Identity Access Management? | | 12 |
|  | 2.2Terminlogies in IAM | | 12 |
|  | 2.3 IAM Services | | 13-16 |
| **3.** | **SAILPOINT BOOTCAMP** | |  |
|  | 3.1 Introduction to SailPoint | | 17 |
|  | 3.2 Terminologies used in SailPoint | | 17 |
|  | 3.3 Components of SailPoint | | 18 |
|  | 3.4 SailPoint Functional Elements | | 19-21 |
|  | 3.5 Implementation of SailPoint | | 22-31 |
| **4.** | **JAVA BOOTCAMP** | |  |
|  | 4.1 Introduction to JAVA | | 32 |
|  | 4.2 Terminologies in JAVA | | 32 |
|  | 4.3 OOPs Concept in JAVA | | 33-34 |
|  | 4.4 MVC Design Pattern | | 35 |
|  | 4.5 Implementation of JAVA Concepts | | 36-41 |
| **5.** | **PINGIDENTITY** | |  |
|  | 5.1 Introduction to PingIdentity | | 42 |
|  | 5.2 Open Standard Protocols | | 43 |
|  | 5.3 Comparison of Open Standard Protocols | | 44 |
| **6.** | **PINGDIRECTORY** | |  |
|  | 6.1 Introduction to PingDirectory | | 45 |
|  | 6.2 Installation of PingDirectory | | 46 |
|  | 6.3 Configuration of PingDirectory | | 47 |
|  | 6.4 Folder Configuration | | 48 |
|  | 6.5 Supported Standards in PingDirectory | | 49 |
|  | 6.6 Commands in PingDirectory | | 50 |
|  | 6.7 Use Cases | | 51-52 |
| **7.** | **PINGFEDERATE** | |  |
|  | 7.1 Introduction to PingFederate | | 53 |
|  | 7.2 Installation of PingFederate | | 54 |
|  | 7.3 Configuration of PingFederate | | 55 |
|  | 7.4 Logs in PingFederate | | 56-60 |
|  | 7.5 Supported Standards in PingFederate | | 61-62 |
|  | 7.6 Use Cases | | 63-69 |
| **8.** | **PINGACCESS** | |  |
|  | 8.1 Introduction to PingAccess | | 70 |
|  | 8.2 Installation of PingAccess | | 70 |
|  | 8.3 Configuration of PingAccess | | 71 |
|  | 8.4 Why PingAccess? | | 72 |
|  | 8.5 Supported Standards in PingAccess | | 73-74 |
| **9.** | **CONCLUSION** | | 75 |
| **10.** | **REFERENCES** | | 76 |

### **LIST OF FIGURES**

[Figure 1: Steps involved in User life cycle management 13](file:///C:\Users\chigaur\Downloads\Chirag%20(2).docx#_Toc102006666)

[Figure 2: Access Management 13](file:///C:\Users\chigaur\Downloads\Chirag%20(2).docx#_Toc102006667)

[Figure 3: Building blocks of Access Management 14](file:///C:\Users\chigaur\Downloads\Chirag%20(2).docx#_Toc102006668)

[Figure 4: Types of Authentication 14](file:///C:\Users\chigaur\Downloads\Chirag%20(2).docx#_Toc102006669)

[Figure 5: Example of Role based Access Control 16](file:///C:\Users\chigaur\Downloads\Chirag%20(2).docx#_Toc102006670)

[Figure 6: Templates for pre-configured tasks 19](file:///C:\Users\chigaur\Downloads\Chirag%20(2).docx#_Toc102006671)

[Figure 7: IdentityIQ Homepage 20](file:///C:\Users\chigaur\Downloads\Chirag%20(2).docx#_Toc102006672)

[Figure 8: Work Items 20](file:///C:\Users\chigaur\Downloads\Chirag%20(2).docx#_Toc102006673)

[Figure 9: Defining the Financials Application 22](file:///C:\Users\chigaur\Downloads\Chirag%20(2).docx#_Toc102006674)

[Figure 10: Configuring the attributes of accounts 22](file:///C:\Users\chigaur\Downloads\Chirag%20(2).docx#_Toc102006675)

[Figure 11: Correlation mapping of Financial Application 23](file:///C:\Users\chigaur\Downloads\Chirag%20(2).docx#_Toc102006676)

[Figure 12: Account Aggregation Task 24](file:///C:\Users\chigaur\Downloads\Chirag%20(2).docx#_Toc102006677)

[Figure 13: Task Result of Account Aggregation 24](file:///C:\Users\chigaur\Downloads\Chirag%20(2).docx#_Toc102006678)

[Figure 14: Verifying that the user is mapped to Financials Applications account 24](file:///C:\Users\chigaur\Downloads\Chirag%20(2).docx#_Toc102006679)

[Figure 15: Attributes of the Financials Application 25](file:///C:\Users\chigaur\Downloads\Chirag%20(2).docx#_Toc102006680)

[Figure 16: Entitlements of the Financials Application 25](file:///C:\Users\chigaur\Downloads\Chirag%20(2).docx#_Toc102006681)

[Figure 17: HR System Application 26](file:///C:\Users\chigaur\Downloads\Chirag%20(2).docx#_Toc102006682)

[Figure 18: Attributes of HR System Application 27](file:///C:\Users\chigaur\Downloads\Chirag%20(2).docx#_Toc102006683)

[Figure 19: Rule for default password "xyz" 27](file:///C:\Users\chigaur\Downloads\Chirag%20(2).docx#_Toc102006684)

[Figure 20: Contractor Feed Application 28](file:///C:\Users\chigaur\Downloads\Chirag%20(2).docx#_Toc102006685)

[Figure 21: Aggregation of Employee and Contractor Application 29](file:///C:\Users\chigaur\Downloads\Chirag%20(2).docx#_Toc102006686)

[Figure 22: All the standard identity attributes 29](file:///C:\Users\chigaur\Downloads\Chirag%20(2).docx#_Toc102006687)

[Figure 23: Creating 'department' attribute 30](file:///C:\Users\chigaur\Downloads\Chirag%20(2).docx#_Toc102006688)

[Figure 24: Creating ‘Status’ attribute 30](file:///C:\Users\chigaur\Downloads\Chirag%20(2).docx#_Toc102006689)

[Figure 25: Doing manager correlation 31](file:///C:\Users\chigaur\Downloads\Chirag%20(2).docx#_Toc102006690)

[Figure 26: Verifying the attribute-value pair of a user 31](file:///C:\Users\chigaur\Downloads\Chirag%20(2).docx#_Toc102006691)

[Figure 27: OOPs Concepts 33](file:///C:\Users\chigaur\Downloads\Chirag%20(2).docx#_Toc102006692)

[Figure 28: MVC Design 35](file:///C:\Users\chigaur\Downloads\Chirag%20(2).docx#_Toc102006693)

[Figure 29: MVC model of FoodCart Application 36](file:///C:\Users\chigaur\Downloads\Chirag%20(2).docx#_Toc102006694)

[Figure 30: Sequence Diagram of SAML 2.0 43](file:///C:\Users\chigaur\Downloads\Chirag%20(2).docx#_Toc102006695)

[Figure 31: Sequence Diagram of OAuth 2.0 43](file:///C:\Users\chigaur\Downloads\Chirag%20(2).docx#_Toc102006696)

[Figure 32: PingDirectory connectivity to various data stores 45](file:///C:\Users\chigaur\Downloads\Chirag%20(2).docx#_Toc102006697)

[Figure 33: PingDirectory Installation Path 48](file:///C:\Users\chigaur\Downloads\Chirag%20(2).docx#_Toc102006698)

[Figure 34: Example of LDAP Tree 49](file:///C:\Users\chigaur\Downloads\Chirag%20(2).docx#_Toc102006699)

[Figure 35: Protection of application such as SaaS and Web apps through PingFederate. 53](file:///C:\Users\chigaur\Downloads\Chirag%20(2).docx#_Toc102006700)

[Figure 36: Working of OAuth 2.0 73](file:///C:\Users\chigaur\Downloads\Chirag%20(2).docx#_Toc102006701)

[Figure 37: OAuth 2.0 and OpenID Works Together 74](file:///C:\Users\chigaur\Downloads\Chirag%20(2).docx#_Toc102006702)

### **LIST OF TABLES**

[Table 1: IdentityIQ Maintenance tasks 19](#_Toc104557435)

[Table 2: Model component of FoodCart 37](#_Toc104557436)

[Table 3: Controller and View component of 'FoodCart' 37](#_Toc104557437)

[Table 4: Testing the 'FoodCart' Application 38](#_Toc104557438)

[Table 5: Creating JAVA Application to access Derby database 39](#_Toc104557439)

[Table 6: Microservices Implementation 41](#_Toc104557440)

[Table 7: SAML v/s OAuth v/s OIDC 44](#_Toc104557441)

[Table 8: Installation Steps of PingDirectory 46](#_Toc104557442)

[Table 9: Configuration Steps of PingDirectory 47](#_Toc104557443)

[Table 10: LDAP v/s Relational Database 49](#_Toc104557444)

[Table 11: Commands in PingDirectory 50](#_Toc104557445)

[Table 12: Use-case of PingDirectory 52](#_Toc104557446)

[Table 13: Configuration of PingFederate 55](#_Toc104557447)

[Table 14: Types of log file in PingFederate 56](#_Toc104557448)

[Table 15: Audit Log Configuration 56](#_Toc104557449)

[Table 16: Understanding audit.log file 58](#_Toc104557450)

[Table 17: Logging levels 59](#_Toc104557451)

[Table 18: Updating the logging level 60](#_Toc104557452)

[Table 19: PCV and IdP adapter creation 63](#_Toc104557453)

[Table 20: Creating SP connection 65](#_Toc104557454)

[Table 21: Building the connection between SP and IdP 68](#_Toc104557455)

[Table 22: Accessing the URL 69](#_Toc104557456)

[Table 23: Installation of PingAccess 70](#_Toc104557457)

[Table 24: Configuration of PingAccess 71](file:///C:\Users\chigaur\Downloads\CapstoneReport%20v1.5.docx#_Toc104557458)

|  |  |
| --- | --- |
|  |  |
| LDAP | Lightweight Directory Access Protocol |
| SAML | Security Assertion Mark-up Language |
| OAuth | Open Authorization |
| OIDC | OpenID Connect |
| SP | Service Provider |
| IdP | Identity Provider |
| IAM | Identity Access Management |
| WAM | Web Access Management |
| RBA | Role Based Access |
| SSO | Single Sign-on |
| PCV | Password Credentials Validator |
| JDK | Java Development Kit |
| JRE | Java Runtime Environment |
| OOPs | Object Oriented Programming |
| JVM | Java Virtual Machine |
| JDBC | Java Database Connectivity |
| MVC | Model View Controller |
| COTS | Commercial off the Shelf |
| IDaaS | Identity as a Service |

### **LIST OF ACRONYMS**

# Chapter 1

# Introduction

## 1.1 BACKGROUND

As an Advisory Analyst Intern at Deloitte Touche Tohmatsu LLC, I had several opportunities to learn and improve my technical and communication skills throughout the internship. I got a chance to get a deeper understanding of the analysis of extensive data of multiple clients and widening the learning curve while using a combination of different tools, software, and techniques to assist the team in ensuring proper process of the team. The Virtual Technology Internship Program (TIP) at Deloitte had been full of excitement starting from VDI enabled personal laptops to Deloitte-provided laptops, learning through Boot-Camps to applying those learnings on your Projects.

Deloitte Touche Tohmatsu Limited, commonly referred to as Deloitte, is a multinational professional services network with offices in over 150 countries and territories around the world. Deloitte is one of the Big Four accounting organizations and the largest professional services network in the world by revenue and number of professionals, with headquarters in London, England. Deloitte provides audit, consulting, financial advisory, risk advisory, tax, and legal services with approximately 334,800 professionals globally. Deloitte offers services to more than 90% of clients featuring in Fortune 500.

## 1.2 OBJECTIVE

The primary aim of the internship was to understand Identity and Web Access Management. PingIdentity products were used as a tool to demonstrate the concepts of Authentication, Authorization, Single Sign-On (SSO) and Federated identity Management solutions. Some of the use-cases were also implemented with the PingIdentity products.

Following products of PingIdentity were used during the Internship tenure:

1. PingDirectory: It is a fast, scalable directory used to store identity and profile data of the clients.
2. PingFederate: It is a central authentication solution allowing customers, and partners to securely access all of their web applications from any device.
3. PingAccess: It is a centralized access security solution that provides secure access to web applications

# Chapter 2

# Identity Access Management

## 2.1 WHAT IS IDENTITY ACCESS MANAGEMENT?

* Identity and access management (IAM) is the discipline that enables the right individuals to access the right set of resources.
* IAM helps in proving right access to critical systems thus reducing the risk of data loss and privacy of the systems.
* IAM provides services such as User Lifecycle Management, Access Management, SSO, Federation, User Access Management and Certifications.
* IAM provides provisioning i.e., it allows creation of user accounts in trusted application like the HR application, Database, Windows servers, UNIX servers etc., and provides appropriate access to accounts on these applications.

## 2.2 TERMINOLOGIES IN IAM

1. Digital Identity/Identity: The digital representation of a user, including a unique identifier, credentials, core attributes, and context-specific attributes.
2. Authentication: The process of validating identity of a user who is attempting to access a system. Common means of authentication includes username and password, biometrics, smartcard, one-time pins etc.
3. Authorization: The process of determining whether a user is permitted to access an application or data.
4. Provisioning: IAM service responsible for creation of user accounts in information technology (IT) resources and providing appropriate access to those accounts.
5. Self-service: Enabling end user to perform specific activities (e.g., password reset, access request) without the requirements to call service desk.
6. Authoritative Source: An authorised origination points or “source of truth” for a piece of data (e.g., HR for employee) about a user identity.
7. Single Sign-on: It is a user authentication tool that enables users to securely access multiple applications using just one set of credentials.
8. Connector: Component used by IAM System to interface and interact with managed systems.

## 2.3 IAM SERVICES

Some of the key services offered by IAM includes:

1. User Life cycle Management and Provisioning
2. Access Management, SSO and Federation
3. Role Based Access Control
4. Access request and Certification

### 2.3.1 USER LIFE CYCLE MANAGEMENT AND PROVISIONING

User life cycle management is a process which involves the identity creation of the user and managing the access of those identities. This process is also known as identity lifecycle Management.

User life cycle management involves the following steps:

1. User Identity is created in the system as per the business process
2. User is provided access to different system based on the preconfigured business rules
3. User Maintenance includes user data modification, additional role requests, password reset, user Access Review, logging, and reporting
4. Removing user access permissions from all tools /applications and archiving user identity from the system

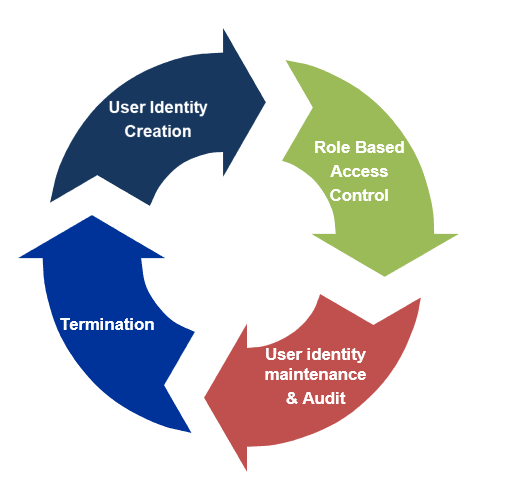


Figure 1: Steps involved in User life cycle management

### 2.3.2 ACCESS MANAGEMENT, SSO AND FEDERATION

2.3.2.1 ACCESS MANAGEMENT

Access Management is a way of managing access of end users. It either allows or denies the user to access a web resource.

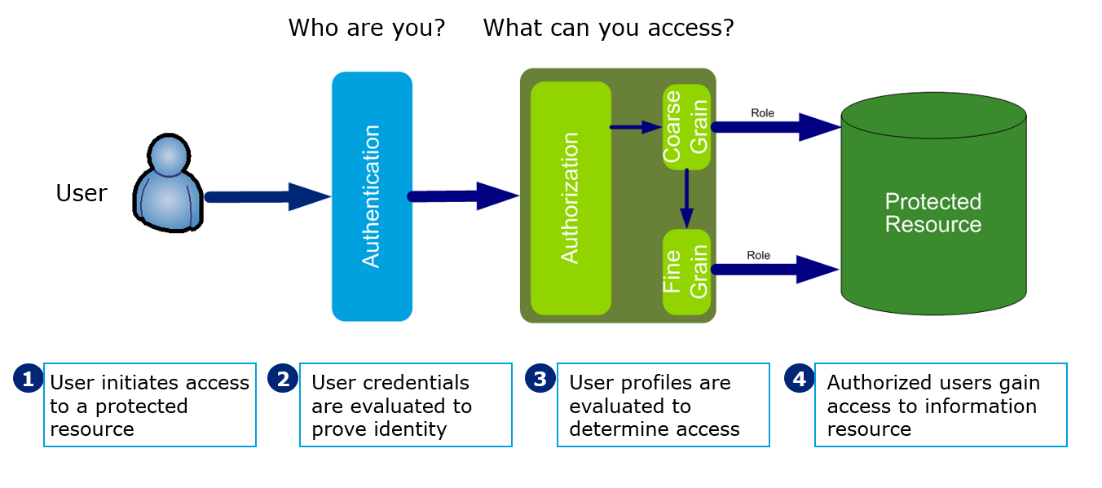


Figure 2: Access Management

There are four building blocks of access management which are as follows:

Figure 3: Building blocks of Access Management

1. Authentication: It is the process of determining the identity of a user that is attempting to access a system. Some of the common methods of authentication includes Basic Username and Password, Tokens, Smartcards and Biometrics



Figure 4: Types of Authentication

1. Authorization: Authorization allows what web resources users can access and what privileges they have such as access to which file directories, hours of access, amount of allocated storage space etc. There are two types of Authorization:
2. Coarse Grained Authorization- It includes protection of web pages, web directories or web applications. Users can access the entire application in this type of authorization method.

Example: User Dev can access only Application X and not Application Y

1. Fine Grained Authorization – It includes granular access control of applications. Access can be restricted to certain part of applications in this type of authorization method.

Example: User Dev can only access function *‘Search User’* in Application X

1. Audit: Auditing is the process of maintaining detailed, secure logs of critical activities in a business environment. For example, the following activities could be audited:

* Login failures
* Unauthorized access to protected resources
* Modification to security policy

1. Access Policies: Access control policies are set of rules and criteria’s that specify how access is restricted to users and who may access information under what circumstances. Access policies provides the following
   * Provide responses that allow or deny access to a resource
   * Redirect the user to other resources
   * Customize the content the user receives based on user attributes
   * Define a user’s session time

2.3.2.2 SINGLE SIGN-ON

Single sign-on (SSO) is a mechanism that allows a user to access different apps integrated with same Access Management solution with just one set of login credentials (for example, access various google services in the same browser session by logging in just once).

2.3.2.3 FEDERATION

Federation is a model based upon trust in which user identities are individually managed and distributed by the service providers or member organizations. Some of the federation standards includes

* SAML
* OAuth
* WS Federation

### 2.3.3 ROLE BASED ACCESS CONTROL

* Role based Access Control (RBAC) is a method which involves assigning access privileges to the user based on a its role in an organization. Access control mechanism which defines, manages, and enforces access control privileges through the use of roles between end user and permission assignments. Users are assigned roles, which in turn provide access to the systems, based on their job responsibilities.
* RBAC is based on principle of Least Privilege & Segregation of Duties
* Principle of Least Privilege: User to be given no more privilege than necessary to perform a job.
* Segregation of Duties: For a particular set of transactions, no single user be allowed to execute all transactions within the set

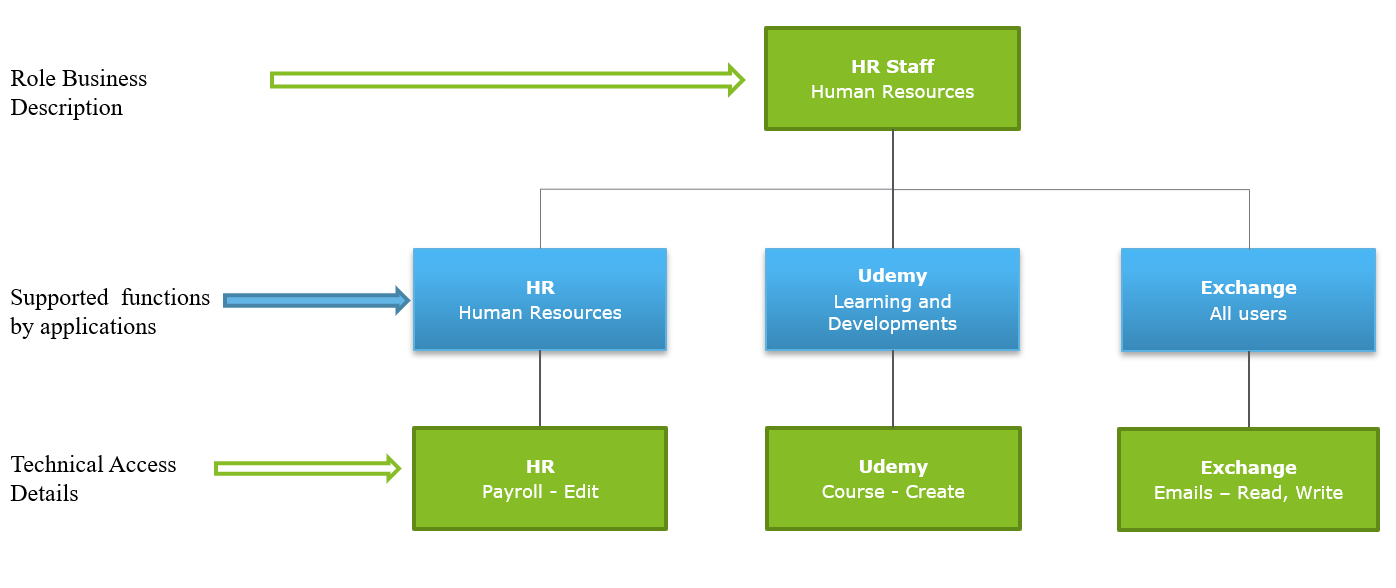


Figure 5: Example of Role based Access Control

### 2.3.4 USER ACCESS CERTIFICATION

* User Access Certification are Periodic access reviews ensure that users are assigned the appropriate minimum access necessary to do their jobs. They are needed to revoke any inappropriate access and enforce policies in areas of separation of duties and least privilege

# Chapter 3

# SailPoint

## 3.1 INTRODUCTION TO SAILPOINT

SailPoint is an Identity Management solution that automates the management and control of access, delivering only the required access to the right identities and technology resources at the right time.

## 3.2 TERMINOLOGIES IN SAILPOINT

i. Identity Cubes: Identity Cubes are a correlated set of accounts and entitlements that represent a single user. Identity Cubes are used by SailPoint IdentityIQ to represent users

ii. Identity Attributes: Identity Attributes are used to describe Identity Cubes and hence describe the real-world user. Identity Attributes are generated by a list of attributes from different sources directly or by using rules or mappings. Name, email, department etc. are all examples of identification attributes.

iii. Connector:  It is a component that imports account data from a variety of targeted networks, applications, and systems.

iv. Provisioning: User provisioning or account provisioning technology creates, modifies, disables, and deletes user accounts and their profiles across IT infrastructure and business applications

v. Account Aggregation: The method of building and upgrading Identity Cubes with an account, attribute, and entitlement data accessed through customized Applications.

vi. Access Certifications: User control rights are reviewed regularly to ensure that they are aligned with the user's job role and follow protocol guidelines.

vii. Policy Enforcements: The collection of preventive and detective controls that ensure the company follows established policies automatically.

viii. Self-service: The method of encouraging users to request resource access through a self-service interface, with the request being forwarded to the required manager(s) for approval using workflow

ix. Password Management: Controlling the setting, resetting, and synchronizing of passwords through networks by automation.

## 3.3 COMPONENTS OF SAILPOINT IDENTITY IQ

SailPoint Identity IQ is made up of four main components:

1. Compliance Manager.
2. Lifecycle Manager.
3. Governance Platform.
4. User Provisioning.

### 3.3.1 COMPLIANCE MANAGER

Compliance Manager combines identity procedures such as Access certification and Policy enforcement and automates auditing, monitoring, and maintenance practices.

Role of Compliance Manager is described as follows:

1. It tracks and prevents unauthorized access and policy breaches in real-time
2. Compliance Manager focuses on the prioritization of the most important compliance tasks and imposes restrictions on the users’ access rights that pose the greatest risk.

### 3.3.2 LIFECYCLE MANAGER

IdentityIQ Lifecycle Manager integrates with authoritative applications such as HR applications to simplify changes to user access arising from a variety of identity lifecycle activities (i.e., new hires, transfers, or terminations).

Role of Lifecycle Manager is described as follows:

1. Enable users to request and handle access on their own.
2. Enable users to update and reset their passwords.
3. Automate identity lifecycle activities will help the user to get control quicker.
4. Offload IT and support desk activities to streamline IT operations.

### 3.3.3 GOVERNANCE PLATFORM

It creates a foundation for effective risk management by centralizing identity data and providing a single place to model roles, policies, risk and business processes.

Governance Platform inside SailPoint can be used :

1. To Assess the risk of each user and resource within the ecosystem to prioritize enforcement and protection efforts.
2. For detective and preventive control, define and use enterprise access policies.

### 3.3.4 USER PROVISIONING

[**User Account Provisioning**](https://www.tools4ever.com/use_case/account-provisioning/) (or user provisioning) ensures user accounts are created, given proper permissions, changed, disabled, and deleted. Inside SailPoint auditors are provided with reports of provisioning changes.

## 3.4 SAILPOINT FUNCTIONAL ELEMENTS

### 3.4.1 TASKS

All data inside IdentityIQ is stored as objects and users run or schedule tasks to act on objects. In simpler words tasks typically used to process data. Many standard tasks are pre-configured, and templates of such task are provided inside SailPoint.

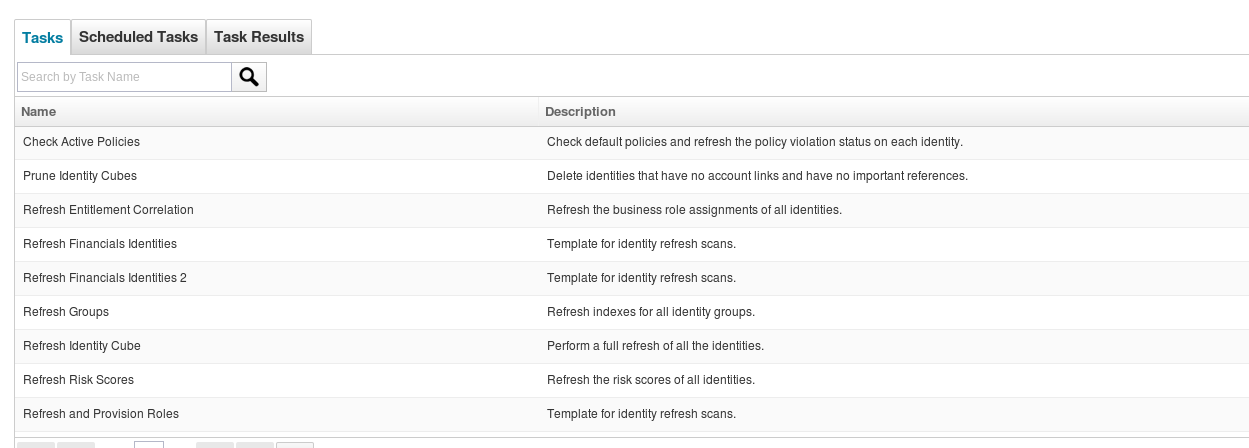


Figure 6: Templates for pre-configured tasks

|  |  |  |
| --- | --- | --- |
| Task | Purpose | Default Schedule |
| Perform maintenance | Keep standards systems moving through their phases | Every 5 minutes |
| Check expired mitigations daily | Scans for policy and certificates exceptions that have expired | Daily |
| Check expired work items | Scans for uncompleted work items that have expired | Daily |
| Perform Identity Request Maintenance | Checks for provisioning | Daily |

Table 1: IdentityIQ Maintenance tasks

### 3.4.2 BUSINESS PROCESSES (WORKFLOW)

A sequence of steps to perform work triggered by system events or a user request. Standard business processes are pre-defined in IdentityIQ.

### 3.4.3 HOME PAGE

IdentityIQ home page comprises of various quick links which is used for accessing frequently used features. Home Page has a header which can be used to access tab such as Identities, Applications, Intelligence and Set-up.

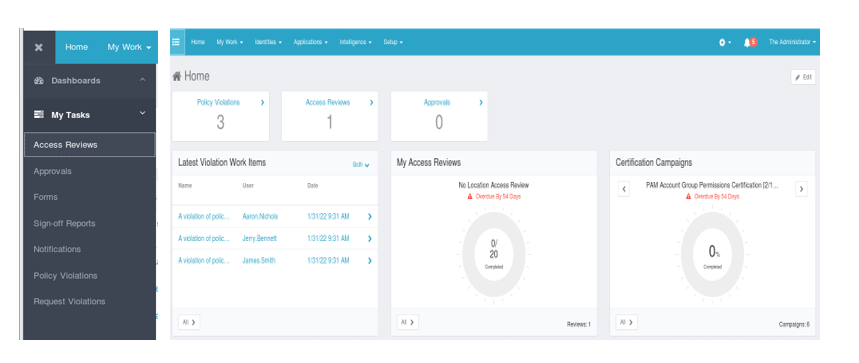


Figure 7: IdentityIQ Homepage

### 3.4.4 WORK ITEMS

Work Items are work that the user needs to complete. User is informed about work items via email Some of the important work items are listed below:

1. Access Request Approval
2. Manual Changes
3. Access Review
4. Policy Violation
5. Delegation

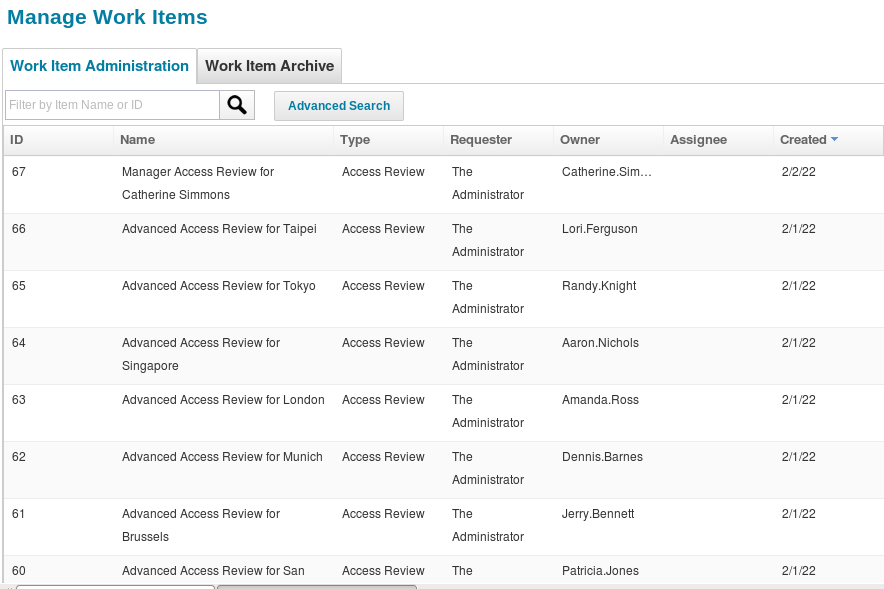


Figure 8: Work Items

### 3.4.5 RULES

Rules are Snippets of user code written to inject business logic

1. Common uses
2. Prepare data for aggregation
3. Customize data during aggregation
4. Define unique business policies
5. Control certification behavior
6. Provide values for drop-down menus

Creation of rules inside SailPoint can be done

1. With IdentityIQ rule editor
2. Create in IDE
3. Import from XML file

## 3.5 IMPLEMENTATION OF SAILPOINT

### USE-CASE I: Loading and Correlating the Financials Application

Objective: In this exercise, we will load and correlate the Financials application. This application is used by people in the Finance department. This application data includes entitlement data as well as account data.

Overview: We will correlate user accounts from this financials application to existing Identity Cubes. We will do this by defining an Account Correlation configuration when we configure the application. Account Correlation can be configured as a simple attribute mapping.

1. Define the Financials Application
   1. Create a new Application definition for the Employee Data
2. Navigate to **Applications Application Definition** and select **Add New Application**
3. Configure the Application as follows:
   * 1. Name: Financials
     2. Owner: spadmin (The Administrator)
     3. Application Type: Delimited File
     4. Authoritative Application: Checked
4. Navigate to **Configuration Settings** and configure as follows:
5. File Path: /home/spadmin/ImplementerTraining/data/Finance-users.csv
6. Delimiter: ,
7. File has column header on first line: Checked
8. Navigate to **Configuration Settings** and configure as follows:
9. Data needs to be Merged: Checked
10. Index Column: dbId
11. Data sorted by the indexColumn(s): Checked
12. Which Columns should be merged: groupmbr

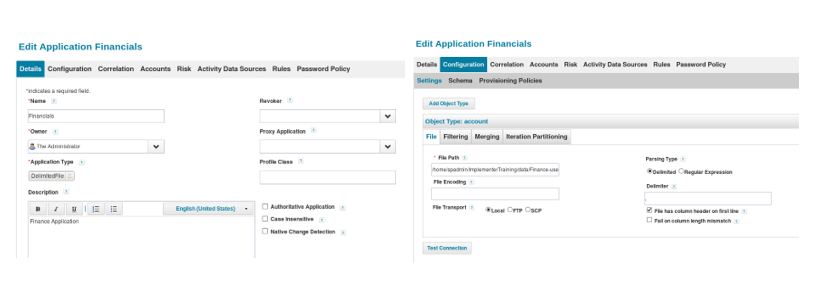


Figure 9: Defining the Financials Application

* 1. We will configure the attributes that we will read from the Financials application.

1. Navigate to **Configuration Schema**
2. For the account schema, select **Discover Schema Attributes**. Configure the schema settings as shown:
3. Native Object Type: account
4. Identity Attribute: dbId
5. Display Attribute: userName
6. Under attributes, for the groupmbr attribute, on the far right, click Edit and check Managed, Entitlement and Multi-Valued

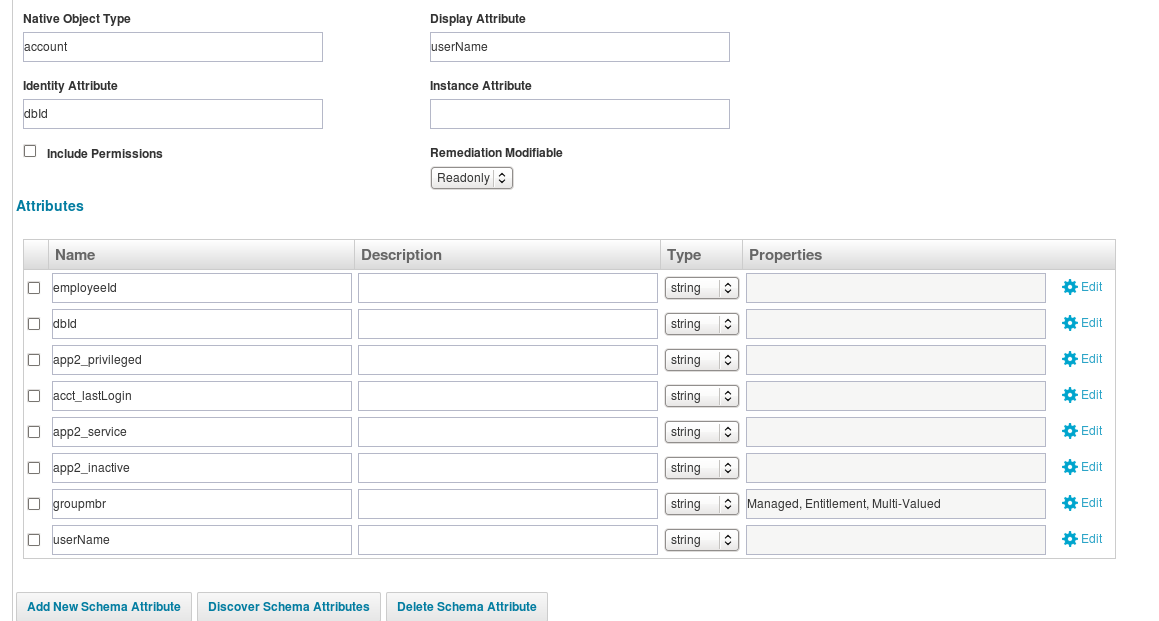


Figure 10: Configuring the attributes of accounts

* 1. Define an Account Correlation configuration to match accounts from this application to existing Identity Cubes

1. Navigate to **Correlation**
2. Click the **New** button
3. Configure the Attribute Based Correlation Assignment as shown, then click **Add**

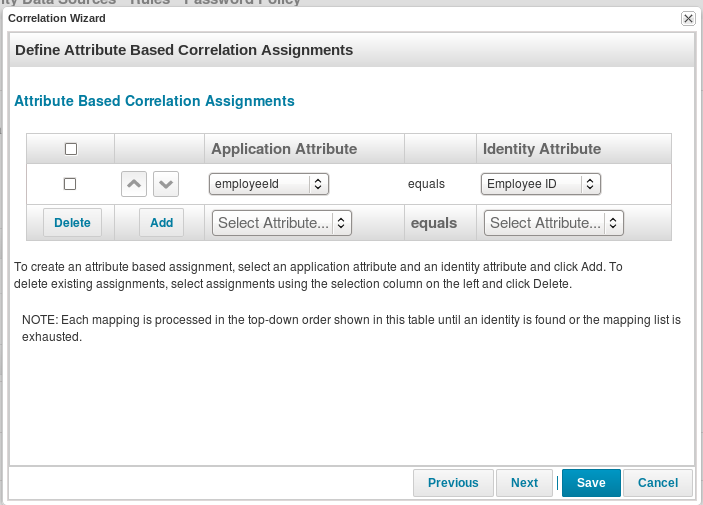


Figure 11: Correlation mapping of Financial Application

1. Aggregate from the financials Application
2. We will now create a task to aggregate accounts from the Financials application.
3. Type: Account Aggregation
4. Name: Aggregate Financial Application
5. Description: Task to aggregate accounts from the Financials application.
6. Select applications to scan: Financials
7. Detect deleted accounts: Checked
8. Disable optimization of unchanged accounts: Checked
9. Promote managed attributes: Checked

Note: Task named ‘Account Aggregation’ is being created to aggregate accounts from the Financials Application.

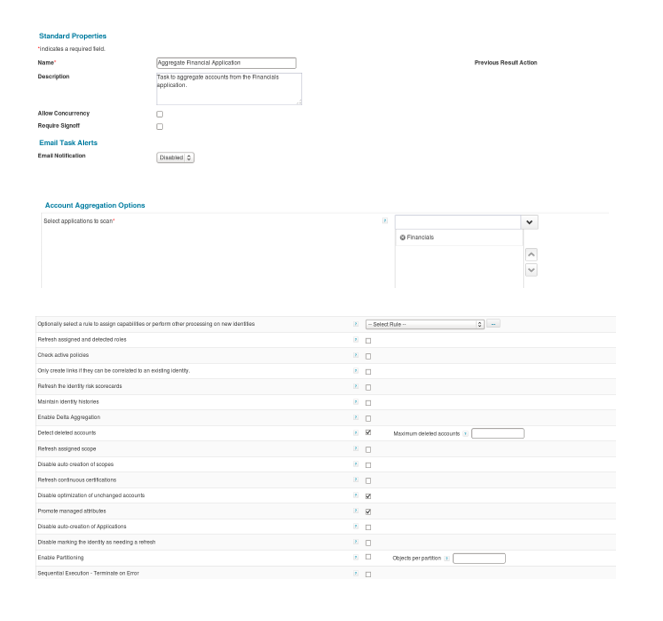


Figure 12: Account Aggregation Task

1. Go to Task Results and confirm that the results are shown as below:

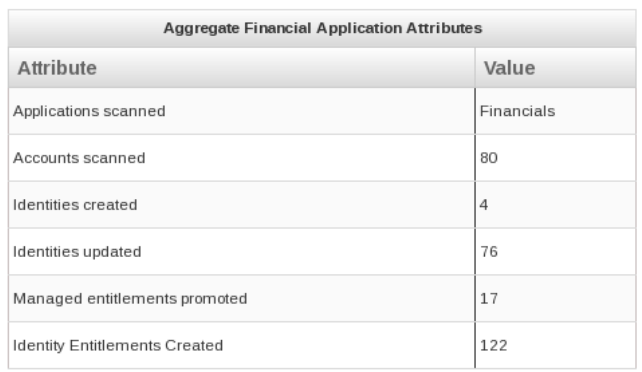


Figure 13: Task Result of Account Aggregation

1. Confirm that Accounts and managed entitlements were properly loaded
2. Navigate to **Identities Identity Warehouse** and find Adam.Kennedy.
3. ****Click **Application Accounts** and it should like this:

Figure 14: Verifying that the user is mapped to Financials Applications account

1. Click the **Financials** account and check the attributes for the Financials application:

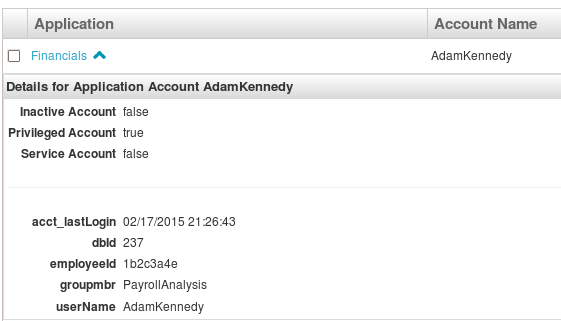
****

Figure 15: Attributes of the Financials Application

1. Within the same Identity (Adam.Kennedy), click **Entitlements** and then under Entitlements, select the groupmbr entitlement to expand the information related to the entitlement.
2. Navigate to **Applications Entitlement Catalog** and notice that the entitlement values from the Financials application have been loaded.

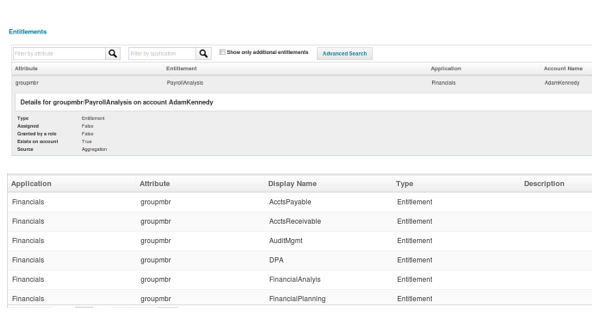
****

Figure 16: Entitlements of the Financials Application

### USE-CASE II: Loading Identity Cubes from Authoritative Source

Objective: The objective of this exercise is to create Identity Cubes from authoritative sources, create new extended identity attributes and mapping identity attributes,

Overview: Assume client has authoritative data stored in two sources. An HR application, stores employee data, and the other application stores contractor data. When defining the applications, Identity Attribute would be “employeeId” and “fullName” as the Display Attribute. I would also create an additional Identity Attribute called “Status” to determine if a user is an Employee or Contractor.

* + 1. Define Employee and Contractor Application
       1. Create a new Application definition for the Employee Data
  1. Navigate to **Applications Application Definition** and select **Add New Application**
  2. Configure the Application as follows:

1. Name: HR System - Employees
2. Owner: spadmin (The Administrator)
3. Application Type: Delimited File
4. Authoritative Application: Checked
   1. Navigate to **Configuration Settings** and configure as follows:
      * 1. File Path: /home/spadmin/ImplementerTraining/data/AuthEmployees.csv
        2. Delimiter: ,
        3. File has column header on first line: Checked

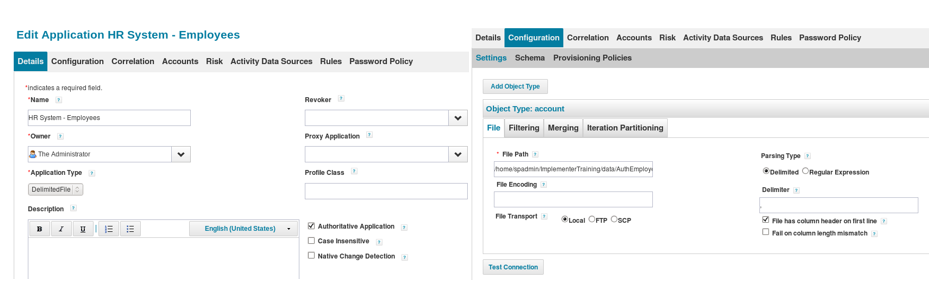


Figure 17: HR System Application

* + 1. We now need to tell the system what attributes we want to read from the file.

a. Navigate to **Configuration Schema**

b. Define the account schema as follows:

i. Native Object Type: account

ii. Identity Attribute: employeeId

iii. Display Attribute: fullName

Note: These above fields define which attributes that we are reading will be used to define uniqueness (“Identity Attribute”) and a friendly display name (“Display Attribute”).

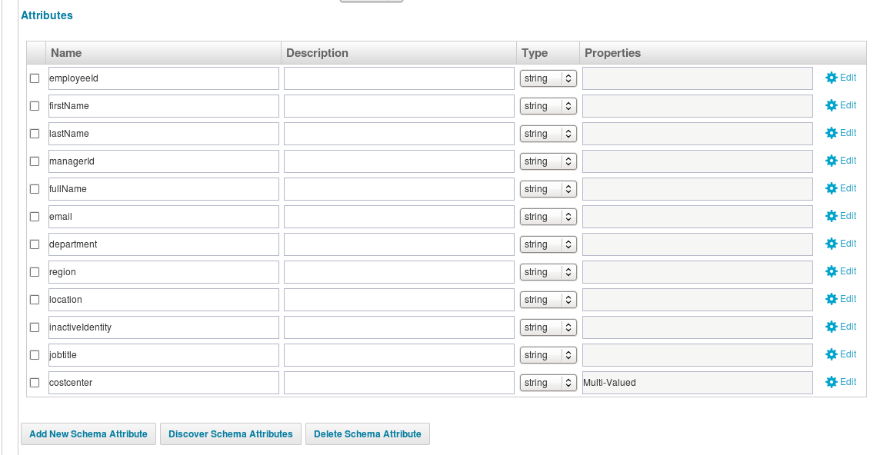
****

Figure 18: Attributes of HR System Application

* + 1. We will now create a rule that will set the default password for each new Identity as we create them.

a. At the top of the page, select **Rules**

b. To the right of the Creation Rule, select the button **(…)**

c. You will now see the Rule Editor

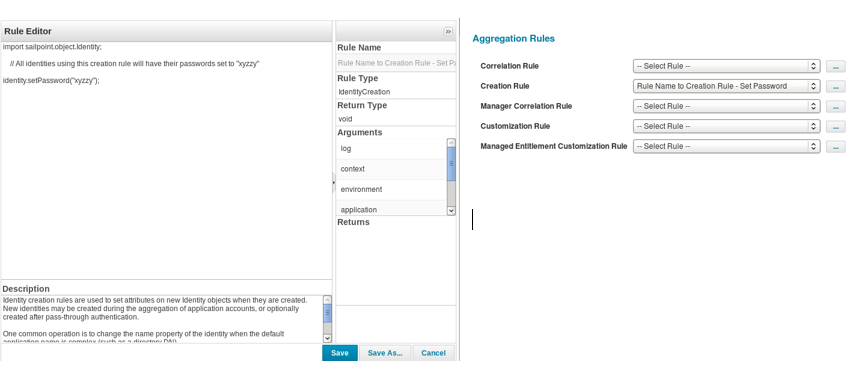
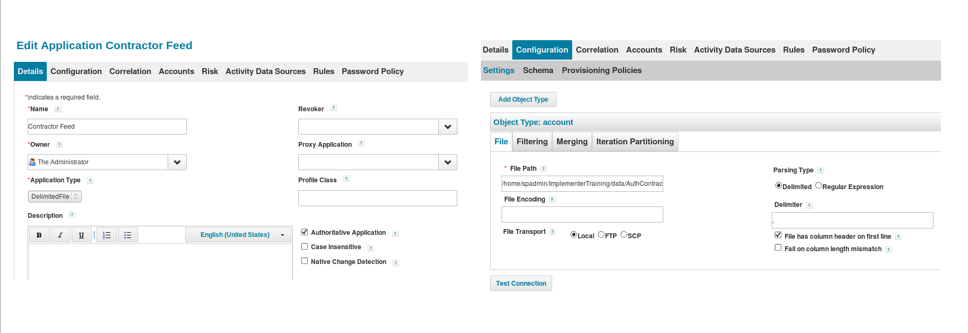
d. Set the Rule Name to Creation Rule - **Set Password** as shown and then select **Save**

Figure 19: Rule for default password "xyz"

4. Perform the same steps for Contractor Application



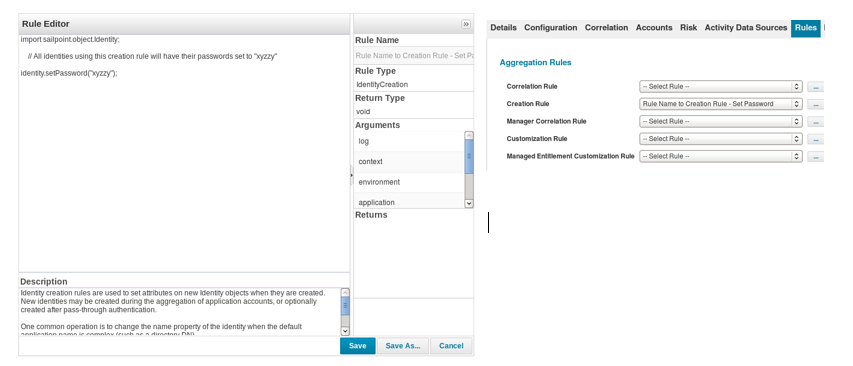


Figure 20: Contractor Feed Application

* + 1. Aggregate the Employee and Contractor Application

1. We will load the Employee and Contractor data from the two delimited files by creating an Account Aggregation task.
   * + - 1. Define the Task as follows:
2. Name: Aggregate Employees and Contractors
3. Description: Aggregate Employees from HR Data and Contractors from Contractor Feed.
4. Select applications to scan: HR System – Employees, Contractor Feed
   * + - 1. Click the Aggregate Employees and Contractors entry to open the task result.
       1. Confirm that aggregation was successful
          1. Navigate to **Identities Identity Warehouse**
          2. Confirm that the Creation Rule was successful

Note: Aggregation is the process of loading account data from Authoritative Applications to generate Identity Cubes.

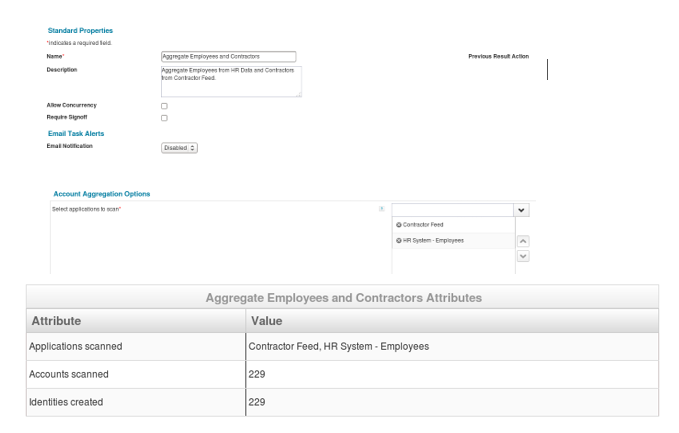


Figure 21: Aggregation of Employee and Contractor Application

* + 1. Configure Identity Mappings for Standard Attributes

1. We now must define what data from these authoritative sources we will use to populate our identity data. Navigate to **Settings Global Settings Identity Mappings**
2. Choose Email from the list of identity attributes
   1. Scroll down to Source Mappings and click **Add** Source to configure the source of this Identity Attribute
   2. Configure the following Settings
3. Choose Application Attribute
4. Application: HR System - Employees
5. Attribute: email
6. Repeat the process for all remaining attributes.
7. After editing the standard identity attributes, it should look like this

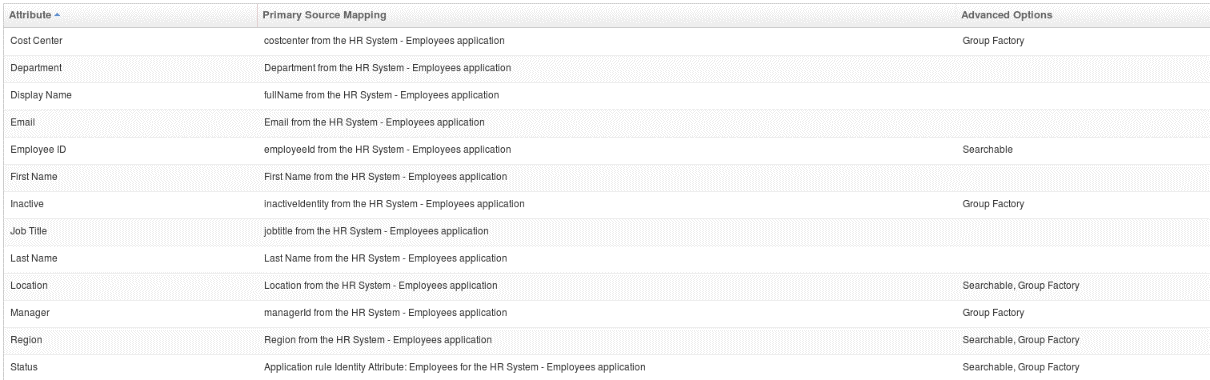


Figure 22: All the standard identity attributes

* + 1. Define Extended Identity Attributes
    2. Click the **Add New** Attribute button on the Identity Attributes page and enter the following:

a. Attribute Name: department

b. Display Name: Department

c. Under Source Mappings, select Add Source

i. Choose Application Attribute

ii. Application: HR System - Employees

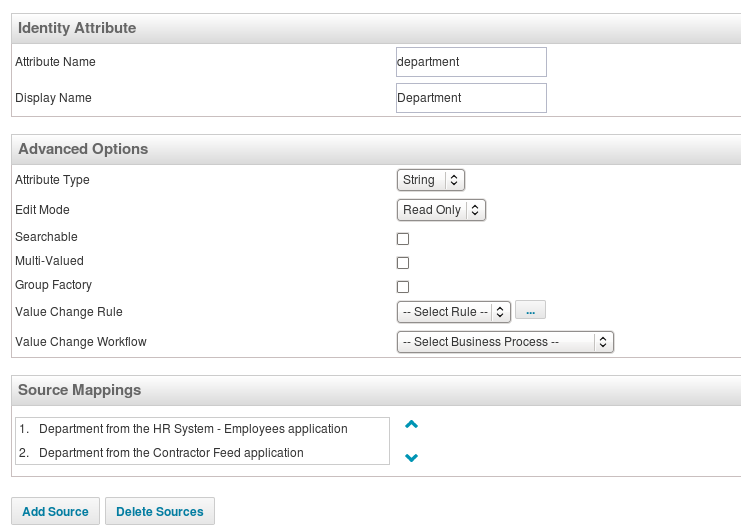
iii. Attribute: department

Figure 23: Creating 'department' attribute

* + 1. We will add an identity’s status attribute which will be either “Employee” or “Contractor”

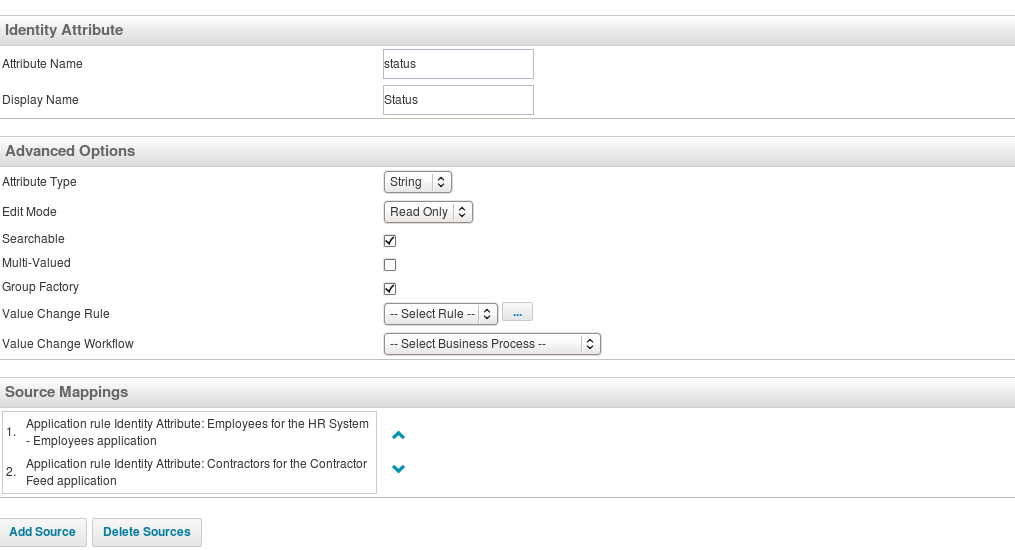


Figure 24: Creating ‘Status’ attribute

* + 1. Update Manager Status

1. Define Manager Correlation for the HR System - Employees application
2. Navigate to **Correlation** tab
3. Under Manager Correlation, set the Application Attribute to “managerId” and the Identity Attribute to “Employee ID”
4. Define Manager Correlation for the Contractor-Feed application
5. Repeat the above steps for the Contractor Feed application.

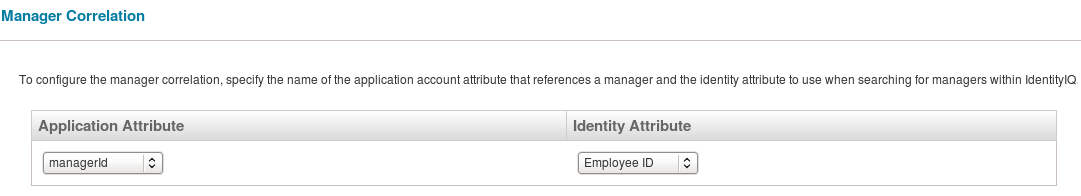


Figure 25: Doing manager correlation

* + 1. Display new Identity Attributes

1. Navigate to **Setup Tasks** and scroll down looking for the Refresh Identity Cube task
2. Visit Task Results and monitor the Refresh Identity Cube task.
3. When the refresh is complete, Navigate to **Identities Identity Warehouse**

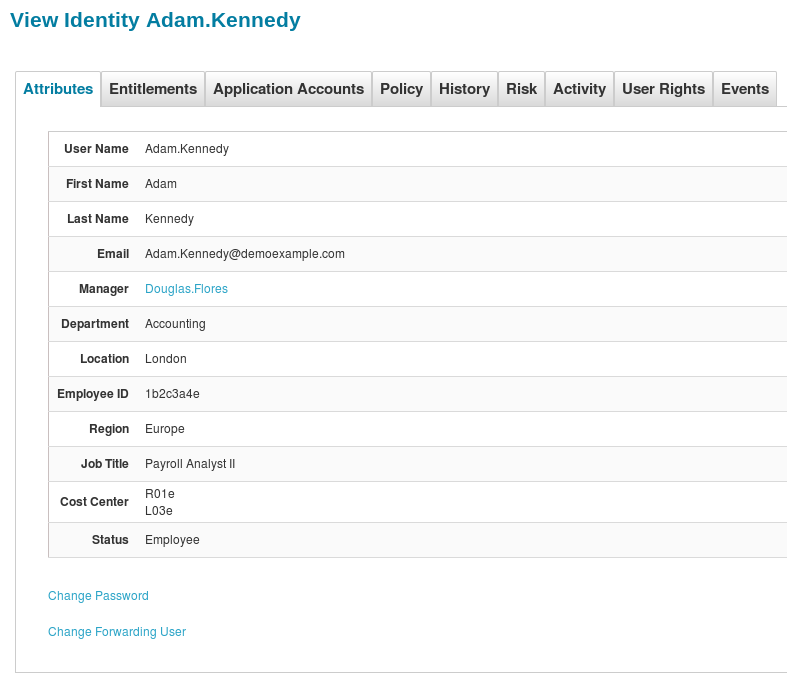


Figure 26: Verifying the attribute-value pair of a user

# Chapter 4

# JAVA Bootcamp

## 4.1 INTRODUCTION TO JAVA

[Java](https://www.geeksforgeeks.org/java/) is a class-based, object-oriented programming language. It is a platform independent language made for developers to write the program once and run it anywhere as compiled Java code can run on all platforms that support Java. Java applications are compiled to byte code that can run on any Java Virtual Machine.

## 4.2 TERMINOLOGIES IN JAVA

i. Java Development Kit (JDK): It is a complete Java development kit that includes compiler, Java Runtime Environment (JRE), java debuggers, java docs, etc.

ii. Java Virtual Machine (JVM): It is generally referred as JVM. JVM executes the bytecode generated by the compiler and output produced by JVM will be same across all operating systems.

iii. Bytecode: The JAVAC compiler of JDK compiles the java source code into bytecode so that it can be executed by JVM.

iv. Java Runtime Environment (JRE):  For running the java program, a computer needs JRE. JRE includes a browser, JVM, applet supports, and plugins.

v. Garbage collectors: In Java, programmers can’t delete the objects. To delete or the memory, JVM has a program called [Garbage Collector](https://www.geeksforgeeks.org/garbage-collection-java/).

vi. Classpath: The [classpath](https://www.geeksforgeeks.org/classpath-in-java/) is the file path where the java runtime and Java compiler look for**.class** files to load

vii. JDBC: JDBC stands for Java Database Connectivity. It is a standard Java API for database connectivity between the Java programming language and relational databases.

viii. Object Oriented Programming (OOPs): Object-oriented programming aims to implement real-world entities like inheritance, hiding, polymorphism etc. in programming. The main aim of OOP is to bind together the data and the functions that operate on them so that no other part of the code can access this data except that function.

ix. Methods:  A Java method can perform some specific tasks without returning anything. Methods in Java allow us to **reuse** the code without retyping the code

x. Variables: **Variable in Java** is a data container that saves the data values during Java program execution. Every variable is assigned a data type that designates the type and quantity of value it can hold.

## 4.3 OOPS CONCEPT IN JAVA:



Figure 27: OOPs Concepts

There are four pillar of OOPs which includes Abstraction, Encapsulation, Inheritance and Polymorphism.

### 4.3.1 CLASS AND OBJECTS

Class: A [class](https://www.geeksforgeeks.org/classes-objects-java/)is a user defined blueprint from which objects are created. It represents the set of properties that are common to all objects of one type

Class declarations can include the following components

1. **Modifiers**: A class can be public or has default access
2. **Class name:** The name should begin with an initial letter.
3. **Superclass:** The name of the class’s parent (superclass), if any, preceded by the keyword extends. A class can only extend (subclass) one parent.
4. **Interfaces:** A comma-separated list of interfaces implemented by the class, if any, preceded by the keyword implements. A class can implement more than one interface.
5. **Body:** The class body surrounded by braces, { }.

Object: Itis a basic unit of Object-Oriented Programming and represents the real-life entities.

### 4.3.2 ABSTRACTION

Data Abstraction is the property by virtue of which only the essential details are displayed to the user. For example, in a car driver does not know about the inner mechanism of the car or the implementation of accelerator, brakes etc. In java, abstraction is achieved by [interfaces](https://www.geeksforgeeks.org/interfaces-in-java/) and [abstract classes](https://www.geeksforgeeks.org/abstract-classes-in-java/). We can achieve 100% abstraction using interfaces.

### 4.3.3 ENCAPSULATION

It is defined as the wrapping up of data under a single unit. It is the mechanism that binds together code and the data it manipulates. Some of the important features of Encapsulation are described below:

1. The variables or data of a class is hidden from any other class and can be accessed only through any member function of own class in which they are declared.
2. As in encapsulation, the data in a class is hidden from other classes, so it is also known as data-hiding.
3. Encapsulation can be achieved by declaring all the variables in the class as private and writing public methods in the class to set and get the values of variables.

### 4.3.4 INHERITANCE

It is the mechanism in java by which one class is allowed to inherit the features (variables and methods) of another class.

1. Super Class: The class whose features are inherited is known as superclass or a base class or a parent class.
2. Sub Class: The class that inherits the other class is known as subclass or a derived class, extended class, or child class. The subclass can add its own fields and methods in addition to the superclass fields and methods.

Example: class Manager extends Employee{}

Here Employee is a super class and Manager is a sub class

### 4.3.5 POLYMORPHISM

It refers to the ability of OOPs programming languages to differentiate between entities with the same name efficient. Polymorphism in Java are mainly of 2 types:

1. [Overloading](https://www.geeksforgeeks.org/overloading-in-java/)
2. [Overriding](https://www.geeksforgeeks.org/overriding-in-java/)

Example of Polymorphism:

public class Sum {

    public int sum(int x, int y)

    {

        return (x + y); // This sum takes two integer parameters

    }

    public int sum(int x, int y, int z)

    {

        return (x + y + z); // This sum takes three integer parameters

    }

    public double sum(double x, double y)

    {

        return (x + y); // This sum takes two double parameters

    }

   public static void main(String args[])

    {

        Sum s = new Sum();

        System.out.println(s.sum(10, 20));

        System.out.println(s.sum(10, 20, 30));

        System.out.println(s.sum(10.5, 20.5));

    }

}

## 4.4 MVC DESIGN PATTERN

MVC stands for Model View Controller. The **Model View Controller** (MVC) design pattern specifies that an application consist of a data model, presentation information, and control information. In MVC model each of these should be separated into different objects.

Design components of MVC are as follows:

1. Model: The Model contains only the pure application data and no logic. It describes how to present the data to a user. It represents the business layer of the application.
2. View: The **View** presents the model’s data to the user. The view knows how to access the model’s data, but it does not know what this data means or what the user can do to manipulate it. It defines the presentation of the application.
3. Controller: The **Controller** exists between the view and the model. It listens to events triggered by the view and executes the appropriate reaction to these events. It manages the flow of the application.

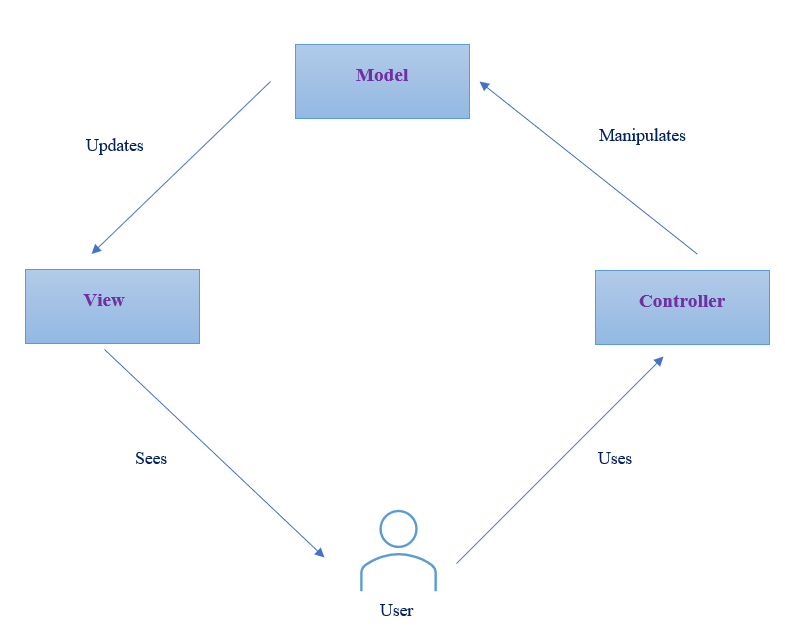


Figure 28: MVC Design

## 4.5 IMPLEMENTATION OF JAVA CONCEPTS

### 4.5.1 CREATING ‘FOODCART’ APPLICATION USING MVC DESIGN PATTERN

Description: I had created a FoodCart; application using MVC design Pattern. This application would comprise of three components.

1. Model: It had been used to connect to the SQL database. Inside SQL database, I had created a schema known as ‘food’. In turn schema ‘food’ had a table by name ‘foodcart’. Model class was used to fetch the data from the table and forward the data of table to Controller class.
2. View: It would be a JSP file which is used to present the data of model to the user.
3. Controller: It would manage the request coming from the browser and request the data from the model. Once it gets the data, it would process it and send to view or JSP.

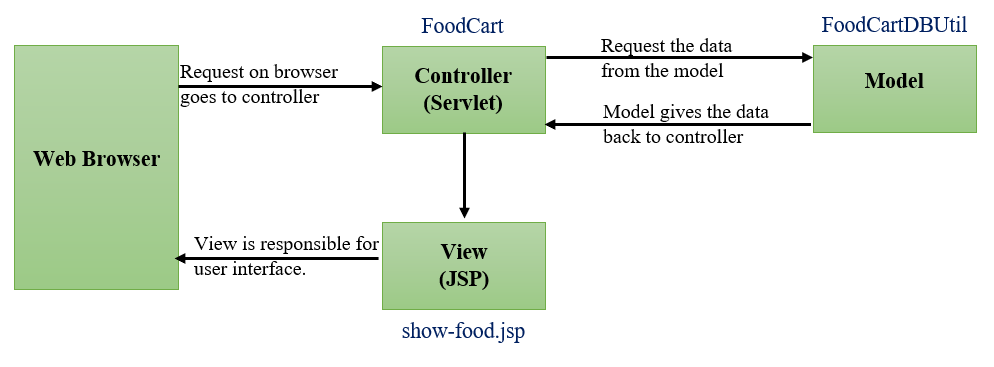


Figure 29: MVC model of FoodCart Application

* + - 1. Create Model component of ‘FoodCart’ Application

|  |  |  |  |
| --- | --- | --- | --- |
|  | Steps | | Screenshots |
| 1 | Table called ‘foodcart’ created inside the database. |  | |
| 2 | Create a food class which defines the variables and methods such as constructor, getters and setters. |  | |
| 3 | Create a model or a helper class which is used to connect to the Database and fetch the data inside the table.  Steps in connecting to the DB are:   1. Passing the DB details such as connection url, userName and password of the DB. 2. Load the MySQL driver. 3. Get the connection details 4. Create a statement 5. Execute the statement and loop over the result set. |  | |

Table 2: Model component of FoodCart

* + - 1. Create Controller component of ‘FoodCart’ Application

Table 3: Controller and View component of 'FoodCart'

|  |  |  |  |
| --- | --- | --- | --- |
|  | Steps | | Screenshots |
| 1 | Create a controller class which manages all the request coming from the browser.  Steps involved while defining controller class are as follows:   1. Get the data from the model class. 2. Map the data coming from model class to a request object. 3. Redirect to a JSP or view class using request dispatcher. |  | |
| 2 | Create a JSP which acts as a View component of the application ‘FoodCart’. |  | |

* + - 1. TESTING THE ‘FOODCART’ APPLICATION

Table 4: Testing the 'FoodCart' Application

|  |  |  |  |
| --- | --- | --- | --- |
|  | Steps | | Screenshots |
| 1 | Create a index.jsp file in order to run the application as a single component. |  | |
| 2 | Running the Application on tomcat server and the webpage as shown in the image would be displayed.  Note: The Application is running on a localhost.  URL: http://localhost:8086/foodcart/ |  | |
| 3 | After clicking on menu, all webpage as shown in the aside image would be displayed. |  | |

### 4.5.2 CREATING SAMPLE JAVA APPLICATION TO ACCESS DERBY DATABASE

Description: In this implementation I have created a JAVA application in order to access and modify Derby database. There is a table created inside the db *ramanadb* named as *student.* Inside the student table there are three attributes

1. StudentId
2. Name
3. Marks

Now the values of these attributes would be sent from the JAVA application and these values would be updated in student table of the database.

|  |  |  |
| --- | --- | --- |
|  | Steps | Screenshots |
| 1 | Connecting to Derby database using Drivermanager.getConnect  ion(“url”) |  |
| 2 | Inserting the attributes values into student table through JAVA application |  |
| 3 | Running the JAVA application |  |
| 4 | Checking the student table inside derby db. |  |

Table 5: Creating JAVA Application to access Derby database

### 4.5.3 CREATING SAMPLE MICROSERVICES

Description: In this implementation I have created three microservices which are BookService, AuthorService and PublisherService. These microservice are handled by single centralized unit known as BookDataServices. BookService holds the details of the book such as bookid, title, publisherid, authorid and price. AuthorService hold the details of the author such as authorid, author name, subject and mobile number. Publisher Service holds the details of the publisher such as publisherid, publisher name and address. Details of each microservice can be fetched through API call using Postman software. All these microservices are running on different ports.

|  |  |  |
| --- | --- | --- |
|  | Steps | Screenshots |
| 1. | Creating controller class for Author microservice |  |
| 2 | Creating controller class for Book microservice |  |
| 3 | Creating controller class for Publisher microservice |  |
| 4. | Creating controller class for BookData. |  |
| 5. | Output of Author microservice |  |
| 6. | Output of Book microservice |  |
| 7. | Output of Publisher microservice |  |

Table 6: Microservices Implementation

# Chapter 5

# PingIdentity

## 5.1 INTRODUCTION TO PINGIDENTITY

PingIdentity is a Commercial off the Shelf (COTS) web access management product provides secure and seamless access to applications on cloud, mobile, SaaS and on-premise applications. It provides Identity Access Management and single Sign-On (SSO) solution to enterprises and organizations across the industries.

PingAccess, PingFederate, PingDirectory, PingOne, PingDataGovernance and PingID are products of PingIdentity.

1. PingID: It is a cloud-based authentication service that binds user identities to mobile devices providing seamless access to the end-users
2. PingOne: It is an identity-as-a-service (IDaaS) single sign-on (SSO) solution that allows businesses to provide federated access to apps with a single click from a secure, cloud-based dock that can be accessed from any browser or mobile device.
3. PingDataGovernance: It provides a graphical user interface for business users to collaboratively build, test and enforce access control policies to data across user directories and APIs.
4. PingDirectory: It is a fast, scalable directory used to store identity and profile data of the clients.
5. PingFederate: It is an enterprise federation server that allows for single sign-on and user authentication. It acts as a central authentication solution, allowing customers, and partners to securely access all of their applications from any device.
6. PingAccess: PingAccess is a centralized access security solution with a comprehensive policy engine that provides secure access to applications and ensures that only authorized users access the resources they need

## 5.2 OPEN STANDARD PROTOCOLS

The protocols used in PingIdentity are:

1. SAML 2.0
2. OAuth 2.0
3. OpenID Connect

### 5.2.1 SAML 2.0

SAML stands for Security Assertion Markup Language. It is an XML based standard which allows users to log into multiple applications using single sign on option. As the user’s identity is stored in directory so it allows to fetch those identities from the directory and log users into different applications using same credentials. SAML protocol has three primary components:

1. User Agent: Generally, it is user’s browser.
2. Identity Provider (IdP): It is a centralized system where user credentials are stored. It is a system which authenticates user credentials. It authenticates the user's identity and authorizes to the service provider.
3. Service Provider (SP): It provides services to end user. It authorizes the given user to access the requested resource. Service providers do not authenticate users but instead request authentication decisions from an identity provider.

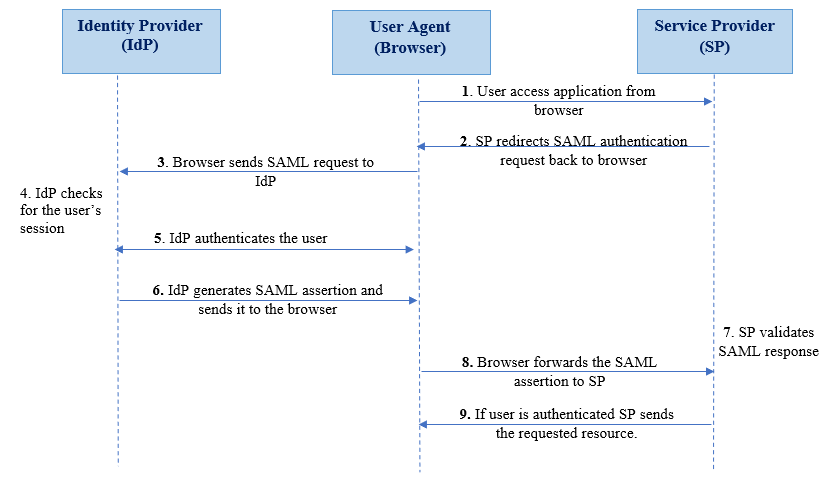


Figure 30: Sequence Diagram of SAML 2.0

### 5.2.2 OAUTH 2.0

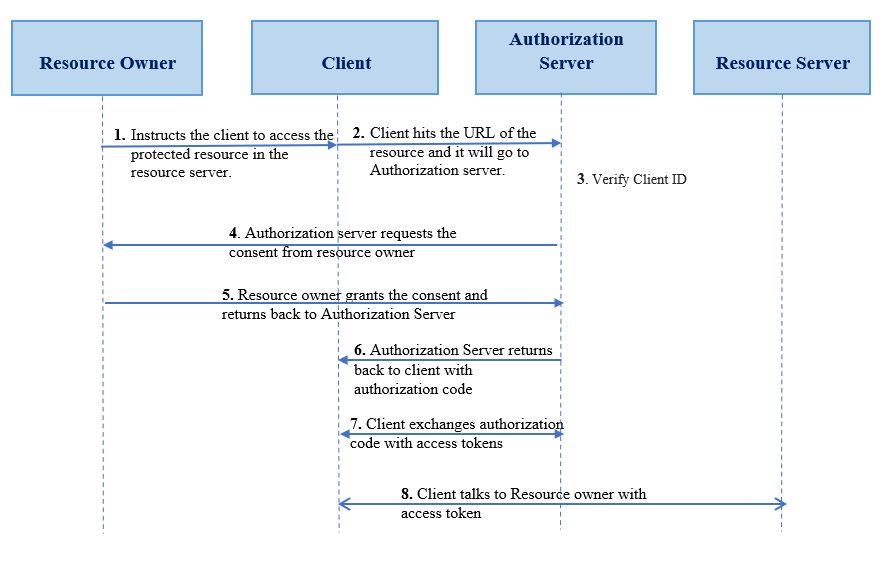
OAuth stands for Open Authorization. It is a standard protocol which allows an application or website to have access to resources which are hosted by some other application. It allows a client application to perform some actions on the resources on behalf on user without having user’s credentials. OAuth is an authorization standard and not an authentication one.

Figure 31: Sequence Diagram of OAuth 2.0

### 5.2.3 OPENID CONNECT

OpenID connect is a simple identity layer on the top of OAuth 2.0 protocol. It enables client to verify the identities of End-user used based on authentication performed by authentication server as well to obtain basic profile information about the End-user

## 5.3 COMPARISON OF OPEN STANDARD PROTOCOLS

|  |  |  |  |
| --- | --- | --- | --- |
|  | **SAML 2.0** | **OAuth 2.0** | **OIDC 2.0** |
| Acronym | Security Assertion Mark-up Language | Open Authorization | OpenID Connect |
| What is it? | Open standard for authentication & authorization | Open standard for authorization | Open standard for authentication |
| History | Developed by OASIS in 2001 | Developed by Twitter and Google in 2006 | Developed by OpenID foundation in 2014 |
| Application | SSO For enterprise | API Authorization | SSO for consumer application |
| Format | XML | JSON | JSON |
| Protocols | HTTP, SOAP | HTTP | HTTP |

Table 7: SAML v/s OAuth v/s OIDC

# Chapter 6

# PingDirectory

## 6.1 INTRODUCTION TO PINGDIRECTORY

PingDirectory is a directory service, which securely stores and maintains identity of users at a large scale. PingDirectory is used by enterprises and organizations to store and manage sensitive customer, partner, and employee data, such as credentials, profiles, preferences, and privacy settings. PingDirectory allows to manage numerous identities and the corresponding attributes associated with these identities. PingDirectory allows enterprises to save costs by avoiding redundant and inconsistent user data. It prevents cyberattacks as identities and profile data is stored in encrypted format. It can be deployed on Ping’s, public and premise cloud. It uses LDAP protocol to access the identities.

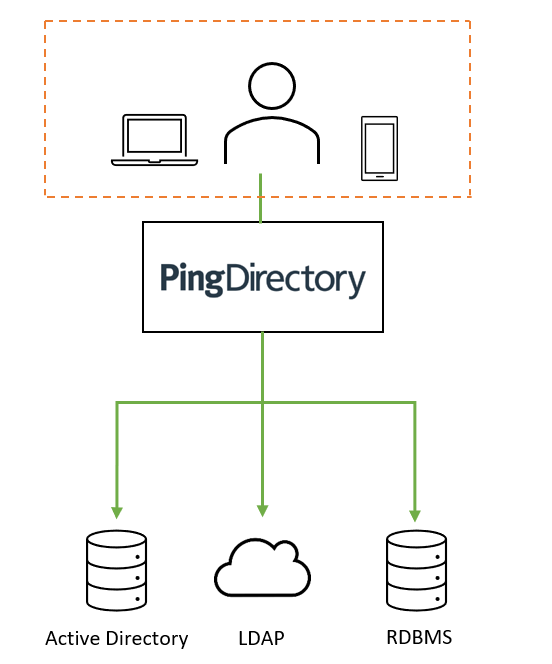


Figure 32: PingDirectory connectivity to various data stores

## 6.2 INSTALLATION OF PINGDIRECTORY

|  |  |  |
| --- | --- | --- |
|  | **Steps** | **Screenshots** |
| 1. | Request the license key for PingDirectory on below URL  <https://www.pingidentity.com/en/account/request-license-key.html> |  |
| 2. | Go to the link: <https://support.pingidentity.com/s/> and then download the license key under the tab  ***Manage License Keys → View → Download*** |  |
| 3. | Inside the VM, go to the URL shown below and download the Linux based Product Distribution (ZIP) file.  [https://www.pingidentity.com/en/resources/ downloads/pingdirectory-downloads.html](https://www.pingidentity.com/en/resources/%20downloads/pingdirectory-downloads.html) |  |
| 4. | Extract the downloaded PingDirectory zip file. |  |

Table 8: Installation Steps of PingDirectory

## 6.3 CONFIGURATION OF PINGDIRECTORY

|  |  |  |
| --- | --- | --- |
|  | **Steps** | **Screenshots** |
| 1. | Inside the PingDirectory folder, open a terminal and run the commands as shown in the figure to begin installation |  |
| 2. | Copy the acquired license file for ping directory to the location mentioned. |  |
| 3. | Enter the host name as **localhost** |  |
| 4. | Set the password and reconfirm it. |  |
| 5. | Select the default options and press Enter |  |
| 6. | Enter the same password in encryption passphrase that was set earlier during step 3 |  |
| 7. | Enter any instance and location name for the Directory Server |  |
| 8. | Select the default options and final configuration should look as shown in the image aside. |  |

Table 9: Configuration Steps of PingDirectory

## 6.4 FOLDER CONFIGURATION



Figure 33: PingDirectory Installation Path

The PingDirectory installation path would have folders such as bat, bin, collector, config etc.

1. Bat folder- Command line utilities for Windows Operating System.
2. Bin folder- Command line utilities for Linux Operating System.
3. Config Folder- It consist of important folders/files:
4. Schema Folder- It comprised of files which contains attributes and object class for defining entry. Custom attributes can also be created
5. Config file- It stores all the information about directory. This is one of the most important files. The extension of this file is ‘.ldif’.
6. Db Folder- In the backend, there Is a database that actually interacts using LDAP. All the databases and cache files are stored in the Db Folder.
7. Log Folder - It consist of following log files:
8. Access Log- It consists of information regarding:

* All operations performed with entries in Directory like adding, deleting, create etc. But attributes will not be present here for user only its DN.
* Error regarding Wrong password entered will be present here.
* Its purpose is to check whether Authentication was successful or not as it contains request and error code for unsuccessful Authentication.

1. Error Logs- These are important for administration server as it will contain information regarding:

* High Resource Utilization
* Memory full
* Restart of any instance

1. Audit Log – Modifications done to the directory server like addition, deletion etc. Attributes for the user will be also present here.

## 6.5 SUPPORTED STANDARDS IN PINGDIRECTORY

PingDirectory uses LDAP protocol to authenticate and communicate with the identities.

### 6.5.1 WHAT IS LDAP?

* LDAP stands for Lightweight Directory Access Protocol. It is a client server based standard protocol used to authenticate the identities stored in the directory.
* As the name suggests the operations are much faster as compared to Relational databases.

### 6.5.2 LDAP V/S RELATIONAL DATABASE

|  |  |
| --- | --- |
| LDAP | Relational Database |
| * LDAP has hierarchical tree like structure. | * It has Tabular structure. |
| * In LDAP, it is difficult to represent complex relations. | * In database, complex relationship can be represented efficiently. |
| * LDAP is more optimized for read operation. | * Relational Databases are more optimized for write operations. |

Table 10: LDAP v/s Relational Database

### 6.5.3 TERMINOLOGIES USED IN LDAP

1. Entry**:** LDAP entry is a collection of information about an entity.Each entry consists of three components which includes distinguished name, collection of attributes and collection of object classes.
2. Directory Information tree (DIT):It is a hierarchal structure which represents the entries of organization. It is also called as namespace or LDAP tree.

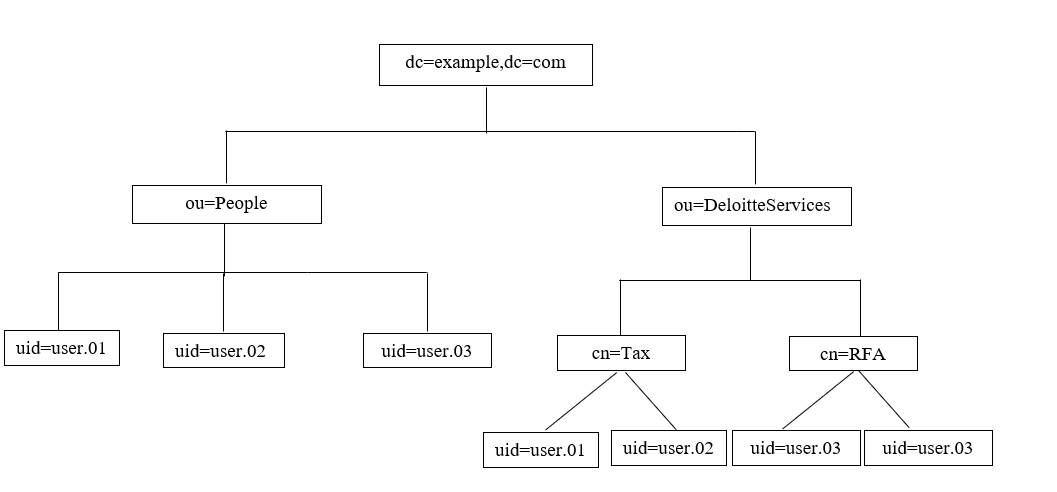


Figure 34: Example of LDAP Tree

1. Distinguished Name (DN):Distinguished name of an entry identifies the unique path of an entry in Directory information tree. It is comprised of attribute-value pairs.

* In figure 34, distinguished name for uid=user.01 can be expressed as: uid=user.01, ou=people, dc=example, dc=com

1. Object Class**:** Object class is used to group related information. It is used to define the structure of entries including the attributes. Each object class allows some attributes which can be seen in schema file.

## 6.6 COMMANDS IN PINGDIRECTORY

|  |  |  |
| --- | --- | --- |
|  | **Commands** | **Screenshots** |
| 1. | **start-server.bat** command is used to start the PingDirectory server |  |
| 2. | **stop-server.bat** command is used to stop the PingDirectory server. |  |
| 3. | Ldapsearch operation is used to fetch the particular account from the Directory.  Command used is:  **ldapsearch.bat -h localhost -p 389 -D cn="Directory Manager" -b dc=example,dc=com uid=\***  where  **-**h is host name  -p is port number  -D is for binding DN  -b is base |  |
| 4. | Ldapmodify command is used to modify the attributes values of one or more than one user accounts.  Command used is: **ldapmodify -h localhost -p 389 -D "cn=Directory Manager" -f "filepath"** |  |

Table 11: Commands in PingDirectory

## 6.7 USE-CASES

### 6.7.1 USE CASE I: Assigning users to the group

Problem Statement: How an organization with numerous employees, clients and external contractors would manage the access of their systems?

Description about the use case: An organization comprising of large number of employees can manage the access of their employees by using Role based Access control. In this use-case above problem solution implementation has been performed. Instead of managing the access of each and every employee, it’s better to restrict the access based on roles. In this use case a group and subgroup has been created. In turn users are mapped to this group.

|  |  |  |
| --- | --- | --- |
|  | **Steps** | **Screenshots** |
| 1 | Creating new organizational unit named ‘DeloitteServices’ using ldapadd command. |  |
| 2 | Adding sub-group named ‘Tax’ under the ou ‘DeloitteServices’. |  |
| 3 | Adding sub-group named ‘RFA’ under the ou ‘DeloitteServices’ |  |
| 4 | Assigning members to the subgroup ‘Tax’ from another group named ‘People’. |  |
| 5 | Assigning members to subgroup ‘RFA’ from another group named ‘People’. |  |
| 6 | Structure of the organizational unit ‘DeloitteServices’ should look similar to the figure shown aside. |  |
| 7 | Now user attributes will be fetched from the newly created ou ‘DeloitteServices’ using ldapsearch command. |  |

Table 12: Use-case of PingDirectory

# Chapter 7

# PingFederate

## 7.1 INTRODUCTION TO PINGFEDERATE

It is an enterprise federation server that allows for single sign-on and user authentication. It acts a central federation authentication solution, allowing customers and partners to securely access all of their applications from any device. Open-standard protocols such as SAML 2.0, OAuth 2.0 and OpenID Connect are supported by PingFederate.

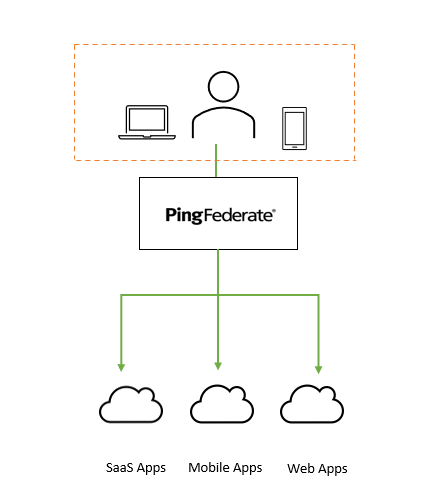


Figure 35: Protection of application such as SaaS and Web apps through PingFederate.

## 7.2 INSTALLATION OF PINGFEDERATE

|  |  |  |  |
| --- | --- | --- | --- |
|  | Steps | Screenshots | |
| 1 | Request the PingFederate key from below URL:  <https://www.pingidentity.com/en/account/request-license-key.html> |  |
| 2 | Go to the URL: <https://support.pingidentity.com/s/> and then download the license key under the tab  ***Manage License Keys → View → Download*** |  | |
| 3 | Inside the VM, go to the URL shown below and download the Linux based Product Distribution (ZIP) file.  <https://www.pingidentity.com/en/resources/downloads/pingfederate.html> |  | |
| 4 | Extract the downloaded PingFederate zip file |  | |

## 7.3 CONFIGURATION OF PINGFEDERATE

|  |  |  |
| --- | --- | --- |
|  | Steps | Screenshots |
| 1 | Go to the bin folder inside the extracted PingFederate folder and add the following line at the beginning of run.sh file  **JAVA\_HOME="/home/chigaur/Downloads/openlogic-openjdk-11.0.8+10-linux-x64"** |  |
| 2 | Open the terminal inside the bin folder and type the command **sudo chmod +x run.sh** |  |
| 3 | Run PingFederate by running the **sudo ./run.sh** command in the terminal |  |
| 4 | Open <https://localhost:9999/> to access the PingFederate console application |  |
| 5 | Click on **Accept** and then select No, Set Up Without PingOne for Enterprise and click **Next** |  |
| 6 | Choose your license file and set up PingFederate console with the default settings. |  |
| 7 | Open the below URL to access the PingFederate console application.  <https://localhost:9999/> |  |

Table 13: Configuration of PingFederate

## 7.4 LOGS IN PINGFEDERATE

### 7.4.1 LOG FOLDER CONFIGURATION

PingFederate generates the following logs as shown in below table in order to store a record of all the events from PingFederate application. PingFederate logs file are written to the PingFederate log directory. The default location is <pf\_install>/pingfederate/log.

|  |  |  |
| --- | --- | --- |
| # | Log File | Description |
| 1 | audit.log | Records a selected transaction log information at runtime such as Transaction time and date. |
| 2 | provisioner.log | Records only provisioning activity. |
| 3 | runtime-api.log | Records actions performed by API users using the OAuth Client Management Service, the OAuth Access Grant Management Service. |
| 4 | server.log | Records PingFederate runtime and administrative server activities. |
| 5 | init.log | Records only jetty messages generated prior to PingFederate setup. |
| 6 | admin.log | Records actions performed by administrative console users. |

Table 14: Types of log file in PingFederate

### 7.4.2 AUDIT LOG

Audit.log file records a selected transaction log information at runtime in order to facilitate security, auditing and regulatory compliances. Audit log holds the details as shown in below table.

|  |  |  |
| --- | --- | --- |
| # | Item | Description |
| 1 | %d | Transaction time |
| 2 | adapterid | The ID of an adapter instance |
| 3 | app | The target SP application |
| 4 | assertionid | The unique ID for the SAML assertion |
| 5 | attributes | User attributes received for an SP log or sent for an IdP log |
| 6 | attrackingid | The tracking ID for OAuth access token |
| 7 | event | Type of transaction (e.g., SSO or Authentication) |
| 8 | granttype | OAuth grant type |
| 9 | host | Pingfederate host name or IP address |
| 10 | initiator | The federation that initiated the SSO or SLO. (SP or IdP) |

Table 15: Audit Log Configuration

#### 7.4.2.1 Demonstrating audit.log file with the help of use-case

|  |  |  |
| --- | --- | --- |
| # | Steps | Screenshots |
| 1 | Access the below URL of the salesforce application. Select **TestSSO** option to login to the application.<atemeinc.my.salesforce.com> |  |
| 2 | Case I: Enter incorrect username while signing in.**Note:** I have used ‘*chirag@deloitte.com’* as the incorrect username. |  |
| 3 | Now, audit.log shows the Authentication details done during step 2.**Note:** The incorrect username entered earlier is also recorded in the audit.log. Status can be seen as **InProgress.** |  |
| 4 | Case II: Enter correct username but wrong password while signing in.**Note:** [mailofsample2@deloitte.com](mailto:mailofsample2@deloitte.com) is the username used to sign in. |  |
| 5 | Now, audit.log shows the transaction details performed during step 4. |  |
| 6 | CASE III: Enter correct credentials to sign into the application. |  |
| 7 | The audit.log displays the type of authentication attempt and also the status is changed to **success** as user is successfully authenticated to the application. |  |

Table 16: Understanding audit.log file

### 7.4.3 LOGGING LEVELS

A logging level is a way of classifying the entries in the log file in terms of urgency. Classification helps in filtering the log files during search and controls the amount of information in logs.

The most common logging levels include FATAL, ERROR, WARN, INFO, DEBUG, TRACE, ALL, and OFF. Some of them are important, others less important, while others are meta-considerations. The [standard ranking](https://www.tutorialspoint.com/log4j/log4j_logging_levels.htm) of logging levels is as follows: ALL < TRACE < DEBUG < INFO < WARN < ERROR < FATAL < OFF.

|  |  |  |
| --- | --- | --- |
| # | Level | Description |
| 1 | OFF | OFF log level does not log anything. It is used to turn off logging |
| 2 | FATAL | FATAL level of logging shows severe error events that will presumably lead the application to abort. |
| 3 | ERROR | ERROR log level shows error that might still allow the application to continue running. |
| 4 | WARN | The WARN log level is used when application have detected an unexpected problem. It shows potentially harmful situations. |
| 5 | INFO | INFO level logs the normal behavior of applications. For example, if a particular service stopped or started or something added to the database. |
| 6 | DEBUG | DEBUG logging level is used to fetch information needed to diagnose, troubleshoot, or test an application |
| 7 | TRACE | TRACE is mostly diagnostic and is more granular and finer than DEBUG log level. |
| 8 | ALL | This log level logs any logging levels that are defined. It logs everything and includes custom logging levels as well. |

Table 17: Logging levels

### 7.4.4 CHANGING LOGGING LEVELS

Overview: PingFederate only records messages that are tagged with log level INFO, WARN, ERROR, and FATAL to the server log. The default logging level in PingFederate is less verbose. Changing the default logging level from INFO to DEBUG helps to fetch more details regarding debugging, diagnosis and troubleshoot of an application.

Steps to change the logging levels inside PingFederate

1. Open the log4j2.xml file using command line terminal in the directory server/default/conf.
2. Make the following changes inside log4j2.xml file to change the logging level from INFO to DEBUG.

i. From:

<Logger name="org.sourceid" level="INFO" />

To:

<Logger name="org.sourceid" level="DEBUG" />

ii. From:

<Logger name="org.sourceid.saml20.util.SystemUtil" level="INFO" additivity="false">

To:

<Logger name="org.sourceid.saml20.util.SystemUtil" level="DEBUG" additivity="false">

iii. From:

<Logger name="com.pingidentity" level="INFO" />

To:

<Logger name="com.pingidentity" level="DEBUG" />

iv. From:

<Logger name="com.pingidentity.common.util.ErrorHandler" level="INFO" additivity="false">

To:

<Logger name="com.pingidentity.common.util.ErrorHandler" level="DEBUG" additivity="false">

v. From:

<Logger name="com.pingidentity.appserver.jetty.PingFederateInit" level="ERROR" additivity="false" includeLocation="false">  
    <AppenderRef ref="CONSOLE" />  
</Logger>

To:

<Logger name="com.pingidentity.appserver.jetty.PingFederateInit" level="INFO" additivity="false" includeLocation="false">  
    <AppenderRef ref="CONSOLE" />  
    <AppenderRef ref="FILE" />  
</Logger>

c.) Save the log4j2.xml file and close it.

Verifying the updated (i.e., DEBUG) logging level in PingFederate

|  |  |  |
| --- | --- | --- |
| # | Steps | Screenshots |
| 1. | Changing the logging level from INFO to DEBUG inside log4j2.xml file. |  |
| 2. | Now, terminal where PingFederate instance is running, populates all the debug and diagnostics messages. It shows all fine-grained informational events that are most useful to debug an application. |  |

Table 18: Updating the logging level

## 7.5 SUPPORTED STANDARDS IN PINGFEDERATE

PingFederate provides flexible, integrated support for the Security Assertion Markup Language (SAML) protocols, WS-Federation, OAuth and OpenID Connect. However, SAML 2.0 is extensively used by the clients and customer of PingFederate to provide the solutions of Single Sign-On and federated identity management.

### 7.4.1 WORKING OF SAML 2.0

SAML protocol has three primary components:

1. User Agent: Generally, it is user’s browser.
2. Identity Provider (IdP): It is a centralized system where user credentials are stored. It is a system which authenticates user credentials. It authenticates the user's identity and authorizes to the service provider.
3. Service Provider (SP): It provides services to end user. It authorizes the given user to access the requested resource. Service providers do not authenticate users but instead request authentication decisions from an identity provider.

When configuring SAML federation we establish a trust relationship between IdP and SP. A user who wants to access the service provider must first authenticate into IdP and if the user successfully authenticates to IdP then IdP generates a SAML assertion. This SAML assertion is sent to the application (SP) and then the user is allowed access to the application. Since the user is already authenticated to IdP so the user can SSO in multiple applications.

### 7.4.2 ROLE OF METADATA FILE IN SAML

The configuration or rule are very essential to configure SAML federation between both IdP and SP. Metadata file contains the configuration and certificates in XML format. These metadata files can be exchanged between IdP and SP to configure the federation.

Metadata file comprises of

1. User Identifier: IdP knows about the user and its attributes. When IdP generates the SAML assertion, it is sent along with user identifier. SP reads the user identifier and maps it to the user in its own user store.

* For ex: SP may request the IdP that the user identifier should be email address and responding to the request IdP will send the SAML assertion with user identifier as email address.

1. NameId-format: It is the format of user identifier.
2. Certificate: Metadata file also contains the sender’s certificate so that receiver end can validate the sender and establish a connection.

### 7.4.3 TYPES OF SSO INITIATION

* 1. IdP Initiated SSO

In this flow user starts by accessing the IdP and then the user is prompted for authentication. Once it is done the user can request a service. If the user is authorized, then IdP generates a SAML assertion. This assertion is sent to the SP through user agent using post message. User agent acts a transport mechanism for this assertion. SP verifies the assertion and maps to local user and then the user’s session can start.

* 1. SP initiated SSO

Here the user starts by reaching out to Service provider. Since the user is not authenticated so SP redirects the user to IdP using a request for authentication message. Once the user is validated the IdP generates the SAML assertion. The assertion is sent to the SP via user agent and session can start.

## 7.5 USE CASES

### 7.5.1 USE CASE I: Integrating a sample website using PingFederate

Description: IAM Showcase website is used as a Service Provider (SP) and PingFederate acts as an Identity Provider (IdP). In this use case, following steps are demonstrated:

1. Creating an instance of Password Credential Validators (PCV) which allows to create user credentials manually (username and password) in ping federate.
2. An IdP adapter is used to provide session information and user identification to PingFederate. We must configure at least one instance of an IdP adapter in order to set up connections to SP partners. It will act on top of PCV, will fetch the information and pass it to SP.
3. SP connection is used to specify which application needs to be configured with IdP. In our case, IAM showcase has been used to act as Service Provider and perform IdP initiated flow. We will download the metadata file from IAM showcase website and will use that metadata file later to establish a SP connection.
4. Creating an instance of PCV and IdP adapter

|  |  |  |
| --- | --- | --- |
|  | **Steps** | **Screenshots** |
| 1 | Navigate to **System**  **Password Credentials Validators** and Create a new PCV |  |
| 2 | Create a new IdP adapter under **Authentication** tab. |  |
| 3 | Select the username attribute as a unique identifier. |  |

Table 19: PCV and IdP adapter creation

1. Creating SP connection

|  |  |  |
| --- | --- | --- |
|  | **Steps** | **Screenshots** |
| 1 | Go to the URL shown below and download the metadata file for IdP initiated SSO.  <https://sptest.iamshowcase.com/> |  |
| 2 | Navigate to **Applications SP Connections** and Select **Create Connection.** |  |
| 3 | Configure the following options under the **Connection Type**  BROWSER SSO PROFILES: **Checked**  PROTOCOL: Select **SAML 2.0** |  |
| 4 | Under the **Import Metadata** Tab, Select **Choose File** option. Upload the metadata file that was downloaded earlier and Select **Next**. |  |
| 5 | Configure the options inside **Assertion Criteria** under Browser SSO Tab. |  |
| 6 | Configure the options under Protocol Settings as shown in the figure aside. |  |
| 7 | Create a certificate under **Certificate’s** tab |  |
| **8** | Configure the changes shown in the figure aside to create a new certificate. |  |
| **9** | Access the IdP initiated URL |  |
| **10** | Sign on with **USERNAME** and **PASSWORD** set up earlier in PCV steps. |  |
| **11** | After signing in following web page will be displayed. IDP Initiated SSO is successfully performed. |  |

Table 20: Creating SP connection

### 7.5.2 USE CASE II: Integrating a SaaS application using PingFederate

Description: In this use case Salesforce application is protected using PingFederate (PingFederate acts as an IdP and Salesforce as an SP). SAML 2.0 is used as a protocol to establish SP connection between both the entities. A user is successfully authenticated using IdP initiated Single Sign-On and SP initiated SSO.

1. Building the connection between PingFederate and Salesforce

|  |  |  |
| --- | --- | --- |
|  | **Steps** | **Screenshots** |
| **1** | To configure PingDirectory with PingFederate create a new data store under the system tab of PingFederate |  |
| **2** | Navigate to **System**  **Password Credentials Validators** and Create a new PCV in order to map PCV to external LDAP storage. |  |
| **3** | Now, go to **Protocol Metadata** under System tab and configure the options shown aside. |  |
| **4** | Export the metadata file of PingFederate and save it. |  |
| **5** | Login to salesforce account and check the box ‘SAML enabled’ under Single Sign-On Settings. Download the metadata file of Salesforce which acts as a Service Provider. |  |
| **6** | Configure the following changes under the set-up tab of Salesforce application |  |
| **7** | Configure the changes under the domain section of salesforce. |  |
| **8** | Add the identity of the user in PingDirectory |  |
| **9** | Create the same user in Salesforce application under the **Users** tab |  |
| **10** | Create a new SP connection in PingFederate |  |

Table 21: Building the connection between SP and IdP

1. Accessing the URL and performing authentication of a sample user.

|  |  |  |
| --- | --- | --- |
|  | Steps | Screenshots |
| **1** | Accessing test application using SP initiated URL shown below.  [atemeinc.my.salesforce.com](https://amedeloitte-my.sharepoint.com/personal/chigaur_deloitte_com/Documents/Desktop/College_Report/Updated/Capstone.docx) |  |
| **2** | Accessing test application using IdP initiated URL shown below.  [https://localhost:9031/idp/startSSO. Ping?PartnerSpId=https %3A%2F%2Fatemeinc.my. salesforce.com](https://localhost:9031/idp/startSSO.%20Ping?PartnerSpId=https%20%3A%2F%2Fatemeinc.my.%20salesforce.com) |  |
| **3** | After entering the credentials in both the cases, browser redirects to salesforce application |  |

Table 22: Accessing the URL

Steps involved in above use-case are :

1. External LDAP data store is configured with PingFederate.
2. Creating an instance of Password Credential Validators (PCV) which allows to link LDAP data store to IdP adapter .
3. Mapping an adapter instance with PCV in order to get session information and user identification.
4. Then SP connection is created in order to specify PingFederate that salesforce needs to be configured.
5. When end user uses SSO to sign in then SP sends the SAML request to browser. In turn browser relays the request to IdP. IdP generates SAML response and sends it to SP through browser.
6. SP validates the SAML response and if user is authenticated then session of user can start in the salesforce application.

# Chapter 8

# PingAccess

## 8.1 INTRODUCTION TO PINGACCESS

* Ping Access is a Web Access Management Solution.
* Ping Access is used to protect websites, APIs, and other web resources using rules and other authentication criteria. It allows both internal and external users to access web applications securely.

## 8.2 INSTALLATION OF PINGACCESS

|  |  |  |
| --- | --- | --- |
|  | **Steps** | **Screenshots** |
| **1** | Request the license key for PingAccess on the below URL:  <https://www.pingidentity.com/en/account/request-license-key.html> |  |
| **2** | Go to the link: <https://support.pingidentity.com/s/> and then download the license key under the tab ***Manage License Keys → View → Download*** |  |
| **3** | Inside the VM, go to the URL shown below and download the Linux based Product Distribution (ZIP) file.  <https://www.pingidentity.com/en/resources/downloads/pingaccess.html> |  |
| **4** | Extract the downloaded zip file of PingAccess. |  |

Table 23: Installation of PingAccess

## 8.3 CONFIGURATION OF PINGACCESS

Table 24: Configuration of PingAccess

|  |  |  |
| --- | --- | --- |
|  | **Steps** | **Screenshots** |
| **1** | Go to the bin folder inside the extracted PingAccess folder and add the following line at the beginning of run.sh file:  **JAVA\_HOME="/home/chigaur/Downloads/openlogic-openjdk-11.0.8+10-linux-x64"** |  |
| **2** | Open the terminal inside the bin folder and type the command **sudo chmod +x run.sh** |  |
| **3** | Run PingAccess by running the **sudo ./run.sh** command in the terminal |  |
| **4** | Open the link: <https://localhost:9000/> to access the PingAccess console application. Select your previously downloaded license file and import it. |  |

## 8.4 WHY PINGACCESS?

* Some Applications that don’t support SAML or may need granular access control. For that purpose, Ping Access is required.
* Ping Access can be deployed as a reverse proxy for all traffic to the application. After authentication by PingFederate, Ping Access can provide session data to the application through HTTP headers, JWT tokens or proprietary token injection.
* When combined with PingFederate, Ping Access enhances security by applying resource policies (URL) and transaction-specific policies with a wider range of contextual data used for authorization.
* Ping Access can validate authentication tokens by communicating with PingFederate in predetermined time intervals.

## 8.5 SUPPORTED STANDARDS IN PINGACCESS

Ping Access uses OAuth 2.0 and OpenID connect. These protocols allow to have a granular access in Ping Access.

### 8.5.1 TERMINOLOGIES USED IN OAUTH 2.0

1. Resource: It is something that is protected and needs to be accessed by some different service. It is also referred as protected resource.
2. Resource Owner: It is an entity who has the access to the resource. An entity capable of granting access to a protected resource. Generally, user is the resource owner.
3. Resource Server: It is the Server that is hosting the Resource.
4. Client: It is an application making protected resource requests on behalf of the resource owner and with its authorization
5. Authorization Server: Resource server is coupled with an Authorization Server. Authorization Server is responsible for making sure that whoever is accessing the resource server is authorized. Authorization server receives the request for access token from client and after successful authentication, it grants them the access token.

1. Redirect URI: Authorization server redirects the user to a location once application has been successfully authorized and granted an authorization code using URL known as Redirect URI. This process is called as Callback.
2. Access Tokens: Access token is a piece of some data which allows to perform authorization to gain access of resource.
3. Scope: Scopes are list of permissions requested from client side. Scopes are sent from client to authorization server while sending the request to access the resource. They are used to specify the type of access (i.e., read only access or read and modify etc.) to be granted to client. Multiple scopes can also be sent.
4. Consent: Consent is used by Authorization server to user asking whether client can do the actions mentioned in the scope.
5. Back channel: It is a Highly secure channel and used to send request from user’s server to other API server from backend. No one can decrypt the information.
6. Front channel: It is Less secure channel and used to send request from browser. As browser might have loopholes like putting secret key in web app inside html then one can see it using view source with inspect element or with chrome developer tools etc.

### 8.5.2 HOW OAUTH 2.0 WORKS?

Here is a use case to understand the OAuth 2.0 standard:

1. Resource owner (i.e., user) will be at Client website and for our use case let’s assume the client website to be **Zomato.com**. Now this website zomato.com needs to access the contacts present in Gmail (Google account. When Resource owner selects the option **‘Connect with Google’** in client website, it is redirected to Authorization server (accounts.google.com be the Authorization Server) and While Redirecting, configuration such as Redirect URL, Response type and Scope is passed from Client to Authorization Server.

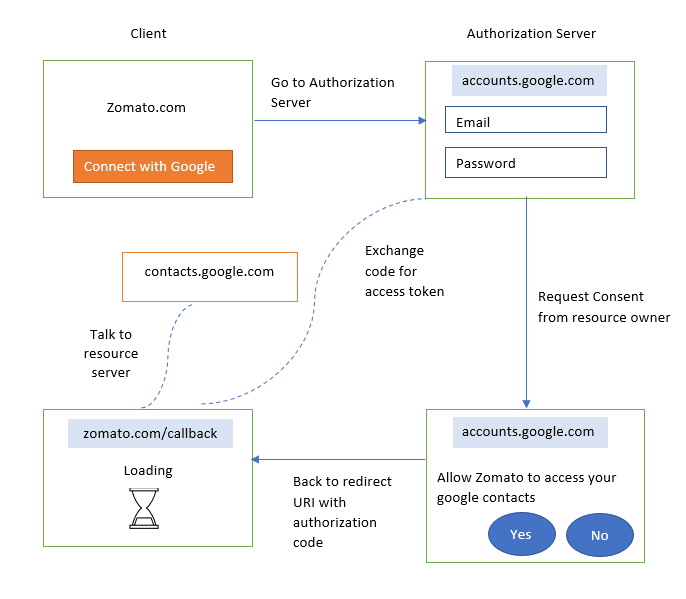


Figure 36: Working of OAuth 2.0

1. Authorization Server requests consent from resource owner to either grant or deny the permission.



1. If consent is granted from the resource owner, then Authorization server redirects to an URL known as Redirect URL along with authorization code.
2. Client goes back to authorization server and exchanges this code for access tokens.
3. Client contacts the resource server with access token that contains the information regarding actions (read only) and after successful validation of access tokens by the resource server, client can access the requested resource.

### 8.5.3 HOW OAUTH 2.0 AND OPENID CONNECT WORKS TOGETHER?

* Since OAuth was designed only for authorization so there is no standard way for getting user’s information. Hence OIDC is used to authenticate the users.
* It is an extra identity layer added on top of the OAuth. It allows to check the identity of the user based on authorization done by authorization server.

Figure 37: OAuth 2.0 and OpenID Works Together

* + In this we can fetch additional info about end user
  + OpenID connect is for Authentication and OAuth 2.0 is for authorization.
  + OpenID adds the following to OAuth 2.0:
* ID token
* If more information is required, then it connects with user information endpoint.
* Standardized implementation
* Standard set of scopes

# Chapter-9

# Conclusion

During my internship at Deloitte Touche Tohmatsu LLC as a Business Technology Analyst Intern, I had various opportunities to learn and grow. I got a chance to get a deeper understanding of the analysis of extensive data of multiple clients and widening the learning curve while using a combination of different tools, software, and techniques to assist the team in ensuring proper process of the team. The insights gained during the internship have helped me in multiple ways. The internship at Deloitte provided me with an opportunity to learn about Identity Access Management which is one of the important aspects of Cybersecurity. PingIdentity products as a tool was used to demonstrate the concepts of Federated identity Management, Identity Access Management and Single Sign On (SSO) solutions. PingDirectory, PingFederate and PingAccess understanding helped me to implement the use cases. PingDirectory is a data store that can be used to maintain identity of users at a large scale. PingFederate is used to provide authentication and SSO services whereas PingAccess aims at delegated access of web applications and API’s to users. The above mentioned tools helps to prevent cyber-attacks.

Apart from technical skills, four months internship at Deloitte provided me with an exposure of corporate world. It assisted me in developing soft skills such as analytical abilities, time management and teamwork. Internship at Deloitte helped me improve my Communication and writing skills.

Interning at Deloitte was one of my first industrial experiences, and it taught me a lot about how businesses operate. It taught me how to run a business and being the world leader in the hospitality industry taught me how to run this section of the industry. From the perspective of the partners, I realized how tiny things can make a big difference when you've invested a lot of money to generate revenue. The success of the entire value chain is determined by how successfully the client's services are supplied. Communication, development, and migration must all be carried out effectively throughout the project's lifespan. In today's commercial world, there isn't much space for error.

**REFERENCES**

1. <https://oauth.net/2/grant-types/authorization-code/>
2. <https://developer.okta.com/blog/2018/04/10/oauth-authorization-code-grant-type>
3. <https://www.pingidentity.com/en/resources/content-library/data-sheets/pingaccess-data-sheet.html>
4. <https://www.pingidentity.com/en/pingone/pingaccess.html#:~:text=PingAccess%20is%20a%20centralized%20access,access%20the%20resources%20they%20need>.
5. <https://openid.net/connect/>
6. https//www.okta.com/identity-101/risk-based-authentication/
7. <https://developer.okta.com/docs/concepts/saml/>
8. <https://developer.okta.com/docs/concepts/oauth-openid/>
9. <https://developer.okta.com/code/java/>
10. Y. Ashibani and Q. H. Mahmoud, "An Intelligent Risk-Based Authentication Approach for Smartphone Applications," 2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC), 2020, pp. 3807-3812, doi: 10.1109/SMC42975.2020.9283267.
11. R. Chivukula, T. Jaya Lakshmi, L. Ranganadha Reddy Kandula and K. Alla, "A Study of Cyber Security Issues and Challenges," 2021 IEEE Bombay Section Signature Conference (IBSSC), 2021, pp. 1-5, doi: 10.1109/IBSSC53889.2021.9673270.