EVIDEN

Identity and Access Management

DirX Identity

Using Domains

Version 8.10.14, Edition March 2026

Table of Contents

Preface

This document describes a set of use cases that explain how to use specific features of DirX Identity. It helps administrators to model their use case with DirX Identity and to set up and run their DirX Identity system. It consists of the following chapters:

- Chapter 1 provides an overview of the use cases presented in this document and gives decision support information.
- Chapter 2 presents background information important for understanding the use cases presented in this document.
- Chapters 3, 4 and 5 explain the use cases related to domain setup in detail.
- Chapter 6 describes the extension to the My-Company sample domain that demonstrates how to support multiple tenants within a single Provisioning domain.
- Chapter 7 explains common setup and removal procedures that apply to setting up and maintaining the use cases.

DirX Identity Documentation Set

*Version 8.10.14 | Build 1858 | Date 2026-03-26 *

The DirX Identity document set consists of the following manuals:

- *DirX Identity Introduction*. Use this book to obtain a description of DirX Identity architecture and components.
- *DirX Identity Release Notes*. Use this book to understand the features and limitations of the current release. This document is shipped with the DirX Identity installation as the file **release-notes.pdf**.
- *DirX Identity History of Changes*. Use this book to understand the features of previous releases. This document is shipped with the DirX Identity installation as the file **history-of-changes.pdf**.
- *DirX Identity Tutorial*. Use this book to get familiar quickly with your DirX Identity installation.
- *DirX Identity Provisioning Administration Guide*. Use this book to obtain a description of DirX Identity provisioning architecture and components and to understand the basic tasks of DirX Identity provisioning administration using DirX Identity Manager.
- *DirX Identity Connectivity Administration Guide*. Use this book to obtain a description of DirX Identity connectivity architecture and components and to understand the basic tasks of DirX Identity connectivity administration using DirX Identity Manager.
- *DirX Identity User Interfaces Guide*. Use this book to obtain a description of the user interfaces provided with DirX Identity.
- *DirX Identity Application Development Guide*. Use this book to obtain information how to extend DirX Identity and to use the default applications.
- *DirX Identity Customization Guide*. Use this book to customize your DirX Identity environment.
- *DirX Identity Integration Framework*. Use this book to understand the DirX Identity framework and to obtain a description how to extend DirX Identity.
- *DirX Identity Web Center Reference*. Use this book to obtain reference information about the DirX Identity Web Center.
- *DirX Identity Web Center Customization Guide*. Use this book to obtain information how to customize the DirX Identity Web Center.
- *DirX Identity Meta Controller Reference*. Use this book to obtain reference information about the DirX Identity meta controller and its associated command-line programs and files.
- *DirX Identity Connectivity Reference*. Use this book to obtain reference information about the DirX Identity agent programs, scripts, and files.
- *DirX Identity Troubleshooting Guide*. Use this book to track down and solve problems in your DirX Identity installation.
- *DirX Identity Installation Guide*. Use this book to install DirX Identity.

- *DirX Identity Migration Guide*. Use this book to migrate from previous versions.

Notation Conventions

Boldface type
In command syntax, bold words and characters represent commands or keywords that must be entered exactly as shown.

In examples, bold words and characters represent user input.

Italic type
In command syntax, italic words and characters represent placeholders for information that you must supply.

[ ]
In command syntax, square braces enclose optional items.

{ }
In command syntax, braces enclose a list from which you must choose one item.

In Tcl syntax, you must actually type in the braces, which will appear in boldface type.

|
In command syntax, the vertical bar separates items in a list of choices.

...
In command syntax, ellipses indicate that the previous item can be repeated.

userID_home_directory
The exact name of the home directory. The default home directory is the home directory of the specified UNIX user, who is logged in on UNIX systems. In this manual, the home pathname is represented by the notation *userID_home_directory*.

install_path
The exact name of the root of the directory where DirX Identity programs and files are installed. The default installation directory is *userID_home_directory*/**DirX Identity** on UNIX systems and **C:\Program Files\DirX\Identity** on Windows systems. During installation the installation directory can be specified. In this manual, the installation-specific portion of pathnames is represented by the notation *install_path*.

dirx_install_path
The exact name of the root of the directory where DirX programs and files are installed. The default installation directory is *userID_home_directory*/**DirX** on UNIX systems and **C:\Program Files\DirX** on Windows systems. During installation the installation directory can be specified. In this manual, the installation-specific portion of pathname is represented by the notation *dirx_install_path*.

dxi_java_home
The exact name of the root directory of the Java environment for DirX Identity. This location is specified while installing the product. For details see the sections "Installation" and "The Java for DirX Identity".

tmp_path

The exact name of the tmp directory. The default tmp directory is /tmp on UNIX systems. In this manual, the tmp pathname is represented by the notation *tmp_path*.

tomcat_install_path
The exact name of the root of the directory where Apache Tomcat programs and files are installed. This location is defined during product installation.

mount_point
The mount point for DVD device (for example, **/cdrom/cdrom0**).

1. Overview

The DirX Identity domain is the entity that supports multi-tenant operation: the ability to configure and run different user populations, privilege models, rules, policies and workflows that remain completely separate from each other within a single DirX Identity installation. Users and administrators operating in a particular domain cannot view or access the data contained in a different domain.

A DirX Identity domain is composed of:

- The System domain, which contains target systems.
- The Provisioning domain, which contains the domain-specific user (identity) data and the Provisioning configuration data; for example, its privilege structure and its policies. This data is clearly separated between different Provisioning domains and is not accessible between them.
- The Connectivity domain, which contains the system-specific data required for synchronization from source systems and to target systems, such as IP addresses and bind profiles. Each Provisioning domain is related to a Connectivity domain. You can configure a separate Connectivity domain for each Provisioning domain or you can have your Provisioning domains share a single Connectivity domain.

The System domain and the Provisioning domains form the Provisioning configuration, and reside on one directory server. The Connectivity domains form the Connectivity configuration. You can have only one Connectivity domain per directory server because of the fixed context prefix **dxmC=DirXmetahub**. This domain can be shared by several Provisioning domains residing on the same directory server and also on other directory servers.

The following figure illustrates the DirX Identity domain architecture.

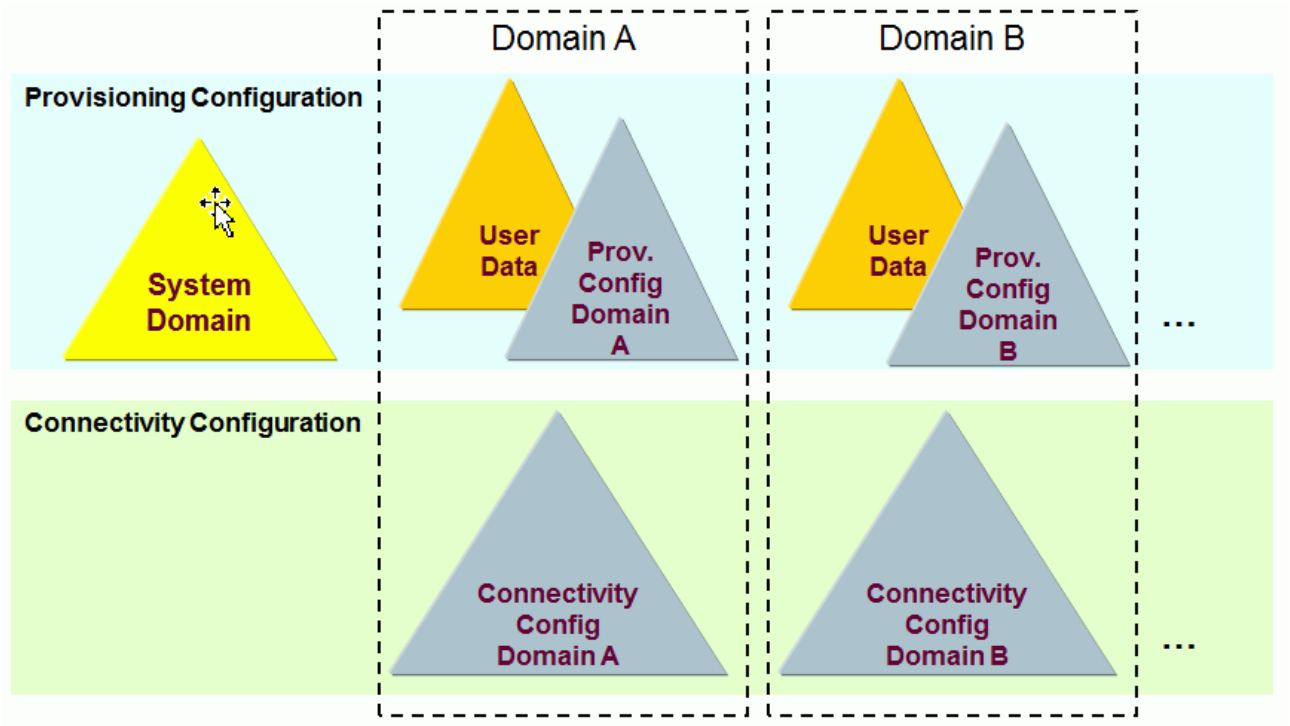

Figure 1. DirX Identity Domain Architecture

A Provisioning domain can host one or many tenants. In a multiple-tenant configuration, each tenant has its own user data and Provisioning configuration data. The Provisioning configuration contains in particular tenant-specific access policies and object creation and approval request workflows. These items help to keep the tenant's data separate from other tenants. There are also Provisioning domain configuration data, which are shared by all tenants. These items include the object descriptions, event policies, attribute modification policies and delete policies. The following figure illustrates this architecture.

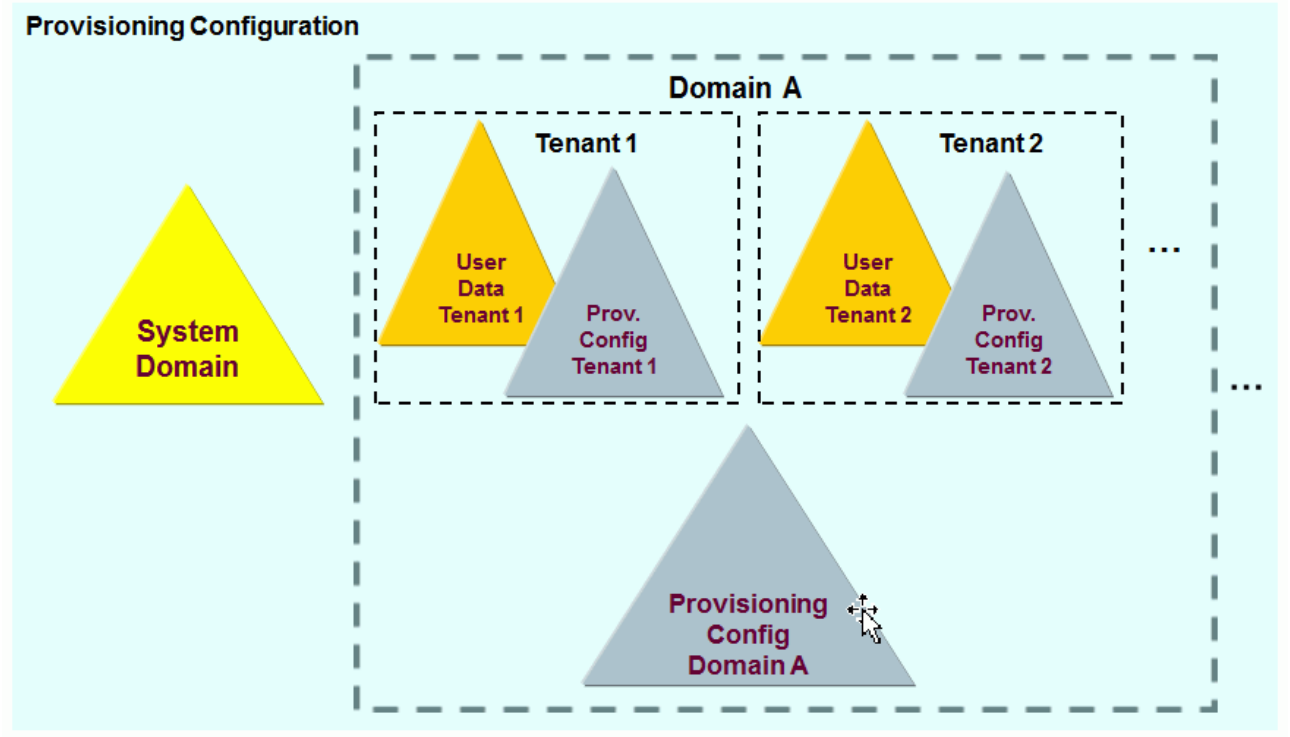

Figure 2. Multi-Tenant Provisioning Domain Architecture

Multi-tenancy can thus be implemented in a variety of ways:

- By configuring one System domain, one Provisioning domain and one Connectivity domain on a single directory server and then duplicating this configuration on an additional directory server for each additional tenant. This configuration is described in the use case “Single Domain”.
- By configuring multiple Provisioning domains that share a single Connectivity domain. This configuration is described in the use case “Multiple Provisioning Domains with a Common Connectivity Domain”.
- By configuring multiple Provisioning-and-Connectivity domain pairs over several different directory servers. This configuration is described in the use case “Multiple Provisioning Domains with Separate Connectivity Domains”.
- By configuring a single Provisioning domain to support multiple tenants. This configuration is described in the use case “Multiple Tenants within a Single Provisioning Domain”.

The remainder of this document provides more information about these use cases.

1.1. Use Cases

The following sections describe typical use cases that occur in customer environments. There are three basic use cases for domain configuration with a standard setup of servers and Web Center. There is another use case that shows how to configure multiple tenants within a single Provisioning domain.

Be aware that other use cases are possible that are not described in this document.

Note also that this document does not discuss distribution aspects that are not relevant in the context of this document; for example, the distribution of C++-based Servers.

1.1.1. Single Domain

If you run only one Provisioning domain, you can keep the System domain, the Provisioning domain and the Connectivity domain on one machine. The following figure illustrates this configuration.

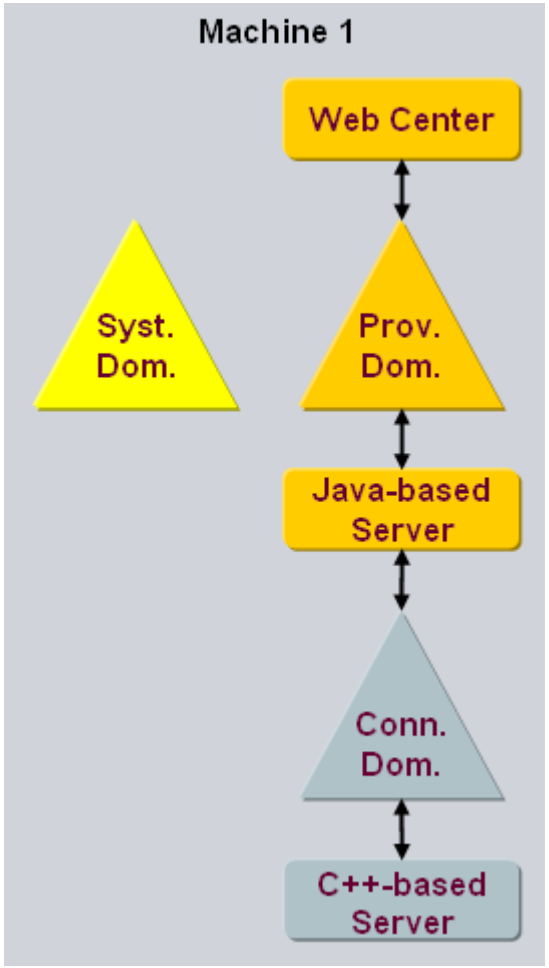

Figure 3. Single Domain on One Machine

As illustrated in the figure, this use case comprises:

- A single Provisioning domain (**Prov. Dom.**) and a related Connectivity domain (**Conn. Dom.**). The System domain (**Syst. Dom.**) is also available.
- A **Web Center** for managing objects in the Provisioning domain.
- A **Java-based Server** that reads and executes request workflows from the Provisioning domain and Java-based provisioning workflows from the Connectivity domain.
- A **C++-based Server** that reads and executes Tcl-based workflows from the Connectivity domain.

This is the standard configuration that most customers use.

You can implement multi-tenancy with this use case by setting up completely separate DirX Identity installations with this domain configuration on multiple machines, one for each tenant. This configuration allows for strict separation of the data but costs more in terms of software maintenance.

1.1.2. Multiple Provisioning Domains with a Common Connectivity Domain

If you intend to run several Provisioning domains, you have two options: you can use a

common Connectivity domain for all the Provisioning domains (this use case) or separate Connectivity domains for each Provisioning domain (the next use case).

Note that this use case assumes that all the Provisioning domains and the Connectivity domain are stored in the same directory server and that all components – directory server, Java- and C++-based Identity servers and Web Center – run on the same machine. It is also possible to have a subset of the Provisioning domains on another directory server and the associated Java-based servers and Web Center on another machine. But this is not discussed here.

Note that this use case requires a machine with enough hardware resources (processors, main memory and file storage) to handle the configuration. The following figure illustrates this use case.

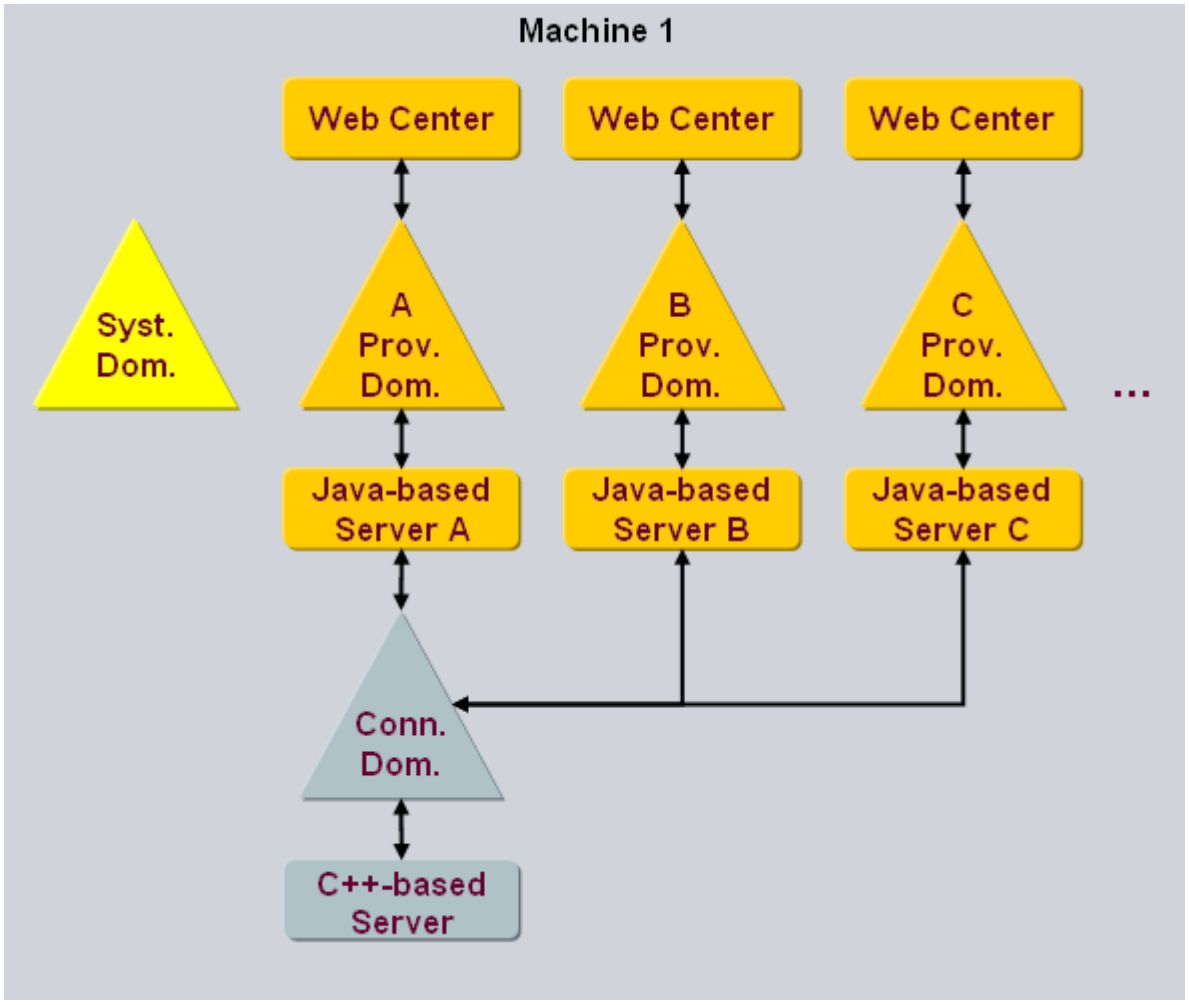

Figure 4. Multiple Provisioning Domains with One Common Connectivity Configuration

This use case comprises:

- Several Provisioning domains (**Prov. Dom. A, B, C ...**) and one common Connectivity Domain (**Conn. Dom.**). The System domain (**Syst. Dom.**) is also available.
- A separate **Web Center** for each Provisioning domain.
- A separate **Java-based Server** for each Provisioning domain. It reads and executes request workflows from the specific Provisioning domain and Java-based workflows

from the common Connectivity domain.

- One **C++-based Server** that reads and executes Tcl-based workflows from the Connectivity domain.

Customers that host several small to mid-size Provisioning domains should use this configuration. User, role, policies and request workflows of the Provisioning domains are clearly separated. All of the provisioning workflows for these domains are kept in the common Connectivity domain but are logically separated using different folder structures.

1.1.3. Multiple Provisioning Domains with Separate Connectivity Domains

If each Provisioning domain needs to have its own Connectivity domain, you must distribute the Connectivity configuration data over different machines, where each one runs its own directory server. The following figure illustrates this configuration.

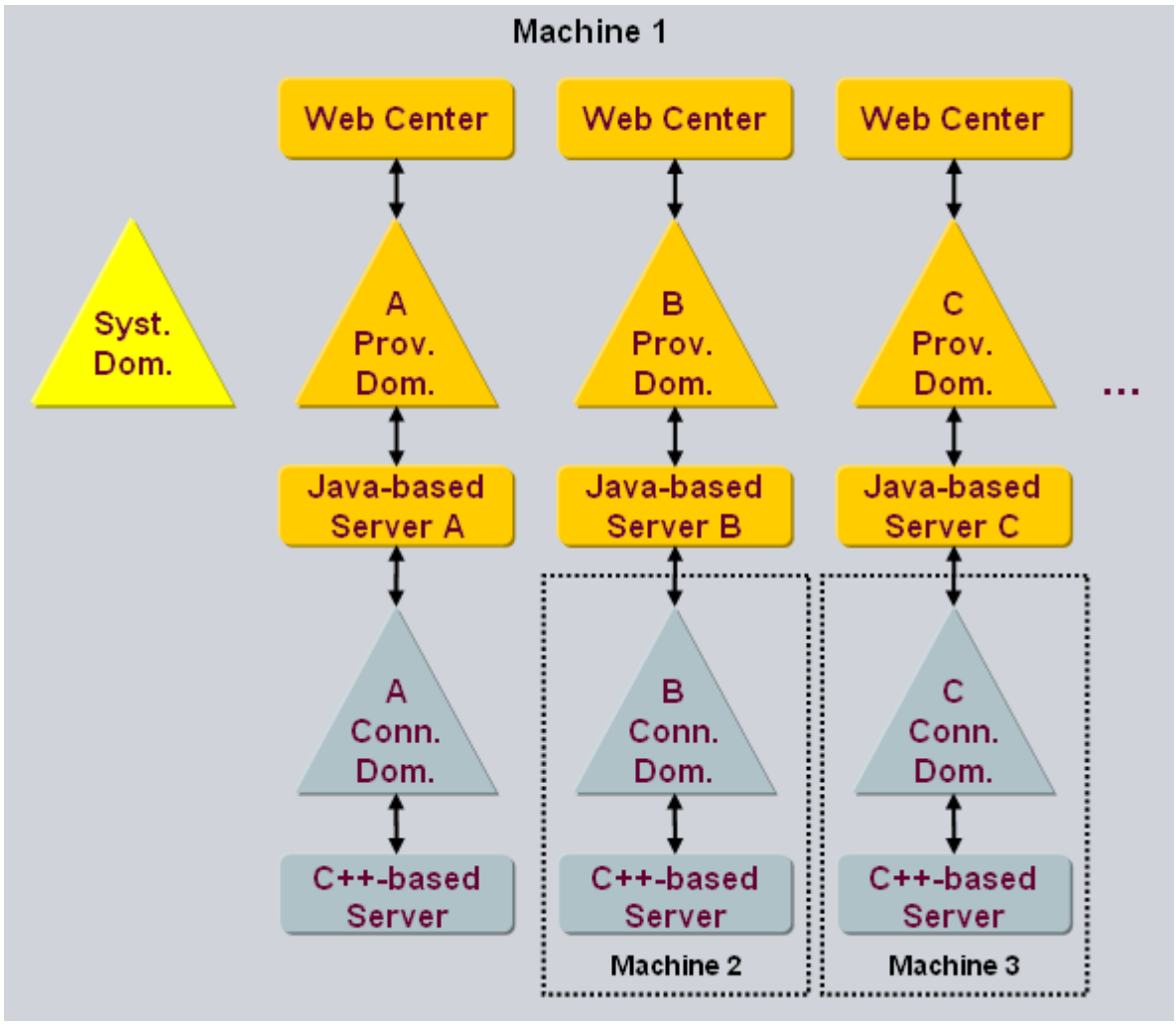

Figure 5. Multiple Provisioning Domains with Separate Connectivity Configurations

As illustrated in the figure, this use case comprises:

- Several Provisioning domains (**Prov. Dom. A, B, C ...**) with corresponding Connectivity domains (**Conn. Dom. A, B, C, ...**). The System domain (**Syst. Dom.**) is also available.

- A separate **Web Center** for managing objects in each Provisioning domain.
- A separate **Java-based Server** that reads and executes request workflows from the specific Provisioning domain and Java-based workflows from the corresponding Connectivity domain.
- A separate **C++-based Server** that reads and executes Tcl-based workflows from the relevant Connectivity domain.

In this scenario, the system domain, all Provisioning domains and the first Connectivity domain can reside on machine 1. You need an additional machine for each additional Connectivity domain.

1.1.4. Multiple Tenants within a Single Provisioning Domain

Customers who find that they frequently need to add new tenants to their DirX Identity installation may want to set up these tenants within a single Provisioning domain, as illustrated in Figure 2, rather than adding multiple separate Provisioning domains. The single Provisioning domain-multiple tenant use case requires less configuration effort. However, their options for specific customizations are limited, because all tenants use the same set of attributes. The customization must focus on object descriptions, especially for users. As a result, this use case does not apply to tenants that want their own data schema and further specific customizations.

This use case comprises:

- A single Provisioning domain as described in the chapter "Single Domain".
- Extensions to default access policies, user and privilege structures and creation workflows necessary for separating multiple tenants within a single Provisioning domain.

Note that the proposed solution for this use case requires the deployment of request workflows and thus the DirX Identity Professional Suite. See the DirX Identity data sheet and the *DirX Identity Installation Guide* for details.

1.1.5. Extensions for Scalability and Higher Throughput

Regardless of whether you run one or multiple Provisioning domains or whether you have one or more Connectivity domains, you can set up each DirX Identity domain with only one or multiple Java-based Servers and Web Centers. The following figure illustrates this concept.

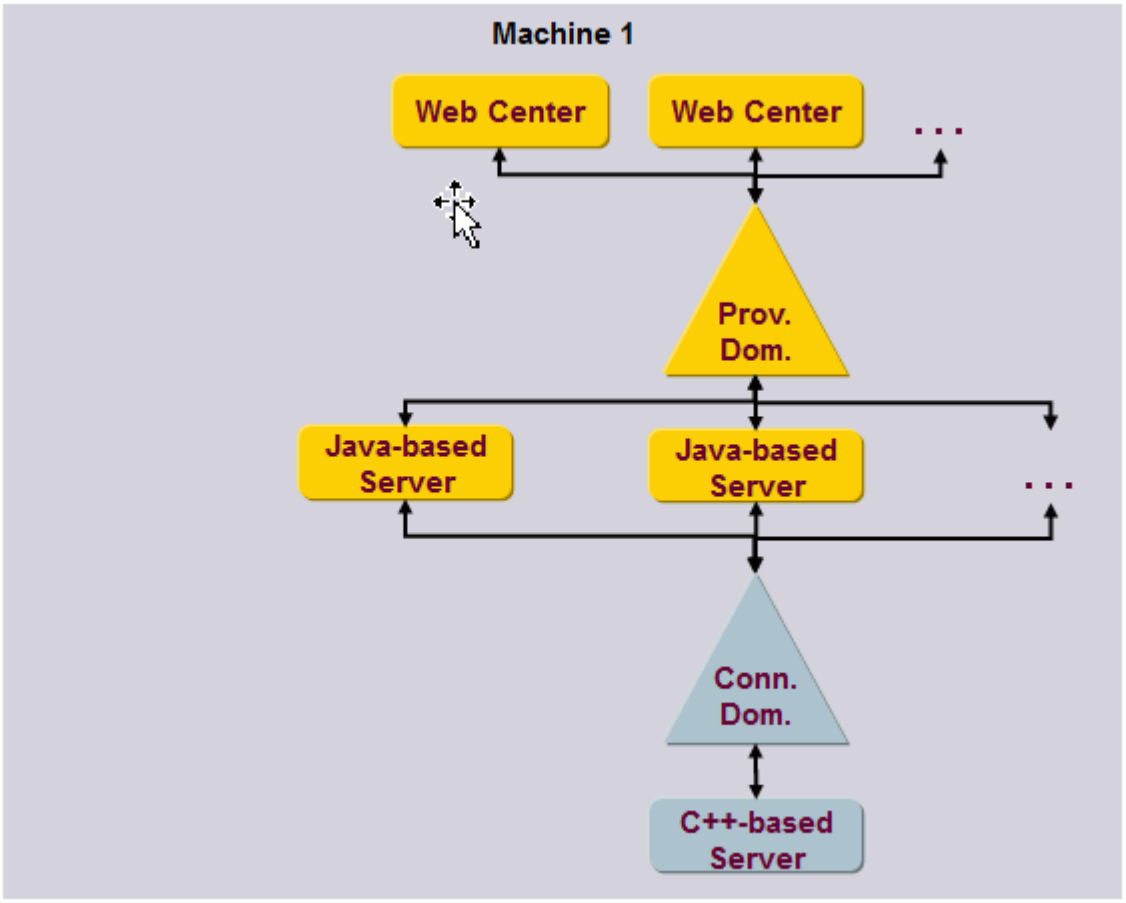

Figure 6. Options for Scalability and Higher Throughput

As shown in the figure:

- You can set up multiple **Web Center** applications on one or more Tomcat servers either to separate the applications for specific user types (end users, administrators) or to handle scalability issues.
- For each Provisioning domain, you can set up and run multiple **Java-based Servers**. You may want to separate specific scenarios from each other - for example, interactive applications from automated background applications - or you may want to enhance performance of the total solution.

You should plan your solution thoroughly for the first setup, but you can add additional resources later on if more performance or separation of specific applications is required. For complete details about scalability and higher throughput, see the use case document *High Availability*.

1.2. Use Case Comparison

The following table compares the use cases described in this document to help you with your decision process.

1.2.1. Number of Provisioning and Connectivity Domains

This comparison evaluates the number of Provisioning and Connectivity domains that

must be set up for each use case.

Table 1. Use Case Comparison - Number of Provisioning and Connectivity Domains

Criteria	Single Provisioning / Connectivity Domain	Multiple Provisioning Domains / Common Connectivity Domain	N Tenants in one Provisioning Domain / Connectivity Domain	Multiple Provisioning Domains / Separate Connectivity Domains
# Provisioning domains	1	N	1	N
# Connectivity domains	1	1	1	N
Strict separation	Yes	No	No	Yes
Moderate separation	Yes	Yes	No	Yes
Separation by access policies needed	No	No	Yes	No
# of Directory Servers	1 to 2	1 to 2	1	N - 1
Set up complexity	Low	Medium	High	High

The table presents the following evaluation criteria:

Provisioning domains - the number of configured Provisioning domains supported by the use case.

Connectivity domains - the number of configured Connectivity domains supported by the use case.

Strict separation – whether the use case offers strict separation of tenant data in Provisioning and Connectivity domains. This requires a single domain scenario per hosted tenant. However, this scenario multiplies the maintenance costs for the installed software and hardware.

Moderate separation – whether the tenant data in a Provisioning domain are strictly separated, but in a common Connectivity domain. Separation of Provisioning domains is sufficient if the hosted customers work only with Web Center or with Identity Manager, but in the Provisioning view group.

Separation by access policies needed – separation of tenant data within a Provisioning domain requires appropriate access policies. The workflow configuration in the Connectivity domain is in common with the “Moderate separation” criterion.

Directory Servers - the number of installed directory servers supported by the use case. For the first two use cases, the customer can select between a common Provisioning and

Connectivity database (1 server) or a distributed database (2 servers).

Set up complexity - the complexity and effort to set up and maintain the use case.

1.2.2. Number of Tenants in a Provisioning Domain

The following table compares the different methods for implementing multi-tenancy: within a single Provisioning domain or across multiple Provisioning domains, one for each tenant.

Table 2. Use Case Comparison: Number of Tenants per Provisioning Domain

Criteria	One Tenant per Provisioning Domain / Multiple Provisioning Domains	Multiple Tenants per Provisioning Domain
Per-tenant data schemas	Yes	No
Per-tenant customizations	Yes	Limited
# Java-based Servers	*N* (one additional server for each Provisioning domain)	1
Professional Suite required	No	Yes

The table presents the following evaluation criteria:

Per-tenant data schemas – whether the use case allows different tenants to have different LDAP data schemas.

Per-tenant customizations – whether the use case allows each tenant to have its own specific customizations. The "*n* tenants per domain" case requires common object descriptions. If tenants use the same target systems, they also share the same object descriptions (including user-to-account attribute mapping) for accounts and groups.

 # Java-based Servers – the number of Java-based Servers that must be deployed.

Professional Suite required – whether the use case requires the Professional Suite version of DirX Identity to be installed.

1.2.3. Scalability and Throughput

For each Provisioning domain, you need to deploy at least one additional Java-based server. Thus, the main difference between the "*n* tenants in 1 Provisioning domain" use case and the "*n* Provisioning domains" use case is that you need to deploy at least 1 to *n* Java-based Servers.

Multiple Provisioning domains and thus multiple customers can share the same C++-based server. You can install at most one C++-based server per system.

The following use case comparison is for one Provisioning domain only.

Table 3. Use Case Comparison: Scalability and Throughput

Criteria	Distribution over Multiple Java-based Servers	Distribution with Resource Families
Distribution type	Cross Servers	Within Server
Resources used	Processes	Threads
Restart after reconfiguration	No	Yes*
Restrictions	Limited by hardware	Limited by hardware

The table presents the following evaluation criteria:

Distribution type - the distribution of messages to Java-based Servers is dynamic. The distribution of messages within a Server depends on the number of configured resource families.

Resources used – whether the use case uses separate processes or threads.

Restart after configuration – whether the use case requires a restart after configuration or re-configuration. Using Web Admin, the administrator can change the number of threads per resource family. This is valid only until the next server start. To make this number persistent, the administrator must use DirX Identity Manager. However, these changes require a re-start of the server.

Restrictions – the restrictions associated with the use case.

2. Background Information

This chapter describes basic information that is important for understanding the descriptions of the use cases presented in this document.

2.1. About Domain Names

You can choose the **Domain Name** of the Provisioning domain freely. Typically, it reflects the customer name or the project name.

Due to restrictions of various operating systems (especially UNIX) it is not possible to use any character of the UTF-8 character set in file, folder or service names. As a result, DirX Identity generates a **Technical Domain Name** from the user's **Domain Name**. This **Technical Domain Name** contains only alphanumeric characters, underlines and dashes.

DirX Identity uses the domain and technical domain name to distinguish items such as files, folders and services.

For details about naming schemes, see the chapter "Managing DirX Identity Servers → Managing the Java-based Server → Naming Schemes" in the *DirX Identity Connectivity Administration Guide*.

Note that the domain name for which a specific Java-based Server works is visible in the Java-based Server tab of Java-based Server object in the Connectivity configuration.

2.2. About Message Topics

If there is a common Connectivity domain for all Provisioning domains, there is only one messaging service instance. This instance transfers messages common to all Provisioning domains and messages specific to one Provisioning domain.

DirX Identity uses the technical domain name in the message topic to associate the messages with their related Provisioning domain.

For a detailed explanation of the Messaging Service in DirX Identity, see the section "Managing DirX Identity Servers → Managing the Messaging Service → Using Messages in DirX Identity " in the *DirX Identity Connectivity Administration Guide*.

Note that the flag **Include domain into topic** in the **General** tab of the Domain object specifies whether publishers - that is, all clients - must include the domain name into the topic. This is the default setting. Resetting this flag is only allowed for specific migration scenarios. See the *DirX Identity Migration Guide* for more information.

Be sure to configure clients that do not read this flag correctly; for example, the Windows Password Listener.

According to this flag, the subscribers - for example, the Java-based Server adaptors - only retrieve messages that belong to their domain.

2.3. About Workflow Configuration

You can configure request and provisioning workflows for each Provisioning domain.

2.3.1. Request Workflows

Configure these workflows in the corresponding **Provisioning** domain.

2.3.2. Java-based Workflows

Java-based workflows include provisioning workflows and password change workflows that work between a DirX Identity domain and connected systems as well as event-based processing workflows that work on entries within the Provisioning domain.

Configure these workflows in the common **Connectivity Configuration** domain in the folder

Workflows → *domain_name*

Note: When using the Target System Wizard to set up new target systems, these workflows are created in these domain-specific folders by default.

2.3.3. Tcl-based Workflows

There is no technical restriction on where to configure Tcl-based Workflows. We recommend that you store domain specific workflows in the folder:

Workflows → *domain_name*

This makes it easy to understand what a specific workflow is for.

2.3.4. Schedules

The Java-based Server loads all active Java-based schedules only for a specific domain from the path **Connectivity** → **Schedules** → *domain*.

3. Single Domain

This chapter describes the use case of installing and configuring a single DirX Identity domain with one System domain, one Provisioning domain and one Connectivity domain. This use case is valid for most customers.

3.1. About this Use Case

Here we install all components on one machine, including:

- The directory server with the **Provisioning domain** and **Connectivity configuration**.
- A **DirX Identity Manager** that allows for managing objects both in the Connectivity and Provisioning configuration.
- A **Web Center** that allows for managing the objects in the Provisioning domain.
- A **C++-based Server** to handle the Tcl-based workflows that are defined in the Connectivity configuration.
- A **Java-based Server** to handle the Java-based workflows that are defined in the Connectivity configuration and the request workflows that are defined in the Provisioning configuration.

See the section "Single Domain" for an illustration of this use case.

Make sure that the machine has enough resources to handle all processes.

3.2. Setup and Configuration

The procedure to set up a single DirX Identity domain is described in detail in the *DirX Identity Installation Guide* in the chapter "Installation on a Single Machine".

After completing the procedure, you will have the following items:

- The directory server with one **Provisioning configuration** and a single **Connectivity configuration**.
- One **DirX Identity Manager** that allows for managing objects both in the Connectivity and in the Provisioning configuration. Login profiles were created for both parts.
- One **Web Center** that allows for managing the objects in the first Provisioning configuration.
- One **C++-based Server** to handle the workflows that are defined in the Connectivity configuration.
- One **Java-based Server** to handle the workflows that are defined in the Connectivity configuration and the Provisioning configuration.
- One **Message Broker** service to process messages.

3.3. Running the Use Case

After installation and configuration, your solution is ready to run. You can now configure and create all necessary objects. This step is highly dependent on your custom Identity Management solution. Read the corresponding guides to perform these tasks.

4. Multiple Provisioning Domains / Common Connectivity Domain

This chapter describes the use case of installing and configuring multiple Provisioning domains and only one common Connectivity domain.

This use case only makes sense for customers that host Identity Management solutions for other customers.

4.1. About this Use Case

In this scenario, we install all components on one machine, including:

- The directory server with several **Provisioning domains** and a single **Connectivity configuration**.
- One **DirX Identity Messaging Service** based on the Apache Active MQ message broker.
- Several **DirX Identity Managers** that allow managing objects both in the common Connectivity configuration and in one of the Provisioning domains.
- Several **Web Centers** that each allow for managing the objects in exactly one Provisioning configuration.
- A **C++-based Server** to handle the workflows that are defined in the Connectivity configuration for all Provisioning domains.
- Several **Java-based Servers** to handle the provisioning and the request workflows that are related to exactly one Provisioning domain.

See the section "Multiple Provisioning Domains with a Common Connectivity Domain" for an illustration of this use case.

Make sure that the machine has enough resources to handle all processes.

4.2. Setup and Configuration

To set up this scenario:

- Set up the first DirX Identity domain.
- Set up additional Provisioning domains.

4.2.1. Setting up the First Domain

Read the section "Setup and Configuration" in the chapter "Single Domain" to set up the first domain. After this step, you will have the following items:

- The directory server with one **Provisioning domain** and a single **Connectivity configuration**.
- One **DirX Identity Messaging Service** based on the Apache Active MQ message broker.

- One **DirX Identity Manager** that allows for managing objects in both the Connectivity and the first Provisioning configuration. Login profiles were created for both parts.
- One **Web Center** that allows for managing the objects in the first Provisioning configuration.
- One **C++-based Server** to handle the workflows that are defined in the Connectivity configuration.
- One **Java-based Server** to handle the request workflows that are related to the first Provisioning domain.

4.2.2. Setting up Additional Provisioning Domains

For each additional Provisioning domain to be set up, perform the steps described in the section "Deploying a Complete Additional Domain" in the chapter "Use Case Deployment Quick Reference".

For each Provisioning domain, you will have:

- A new **Provisioning domain** in your directory server.
- An additional **Web Center** that allows managing the objects in the new Provisioning configuration.
- An additional **Java-based Server** to handle the workflows that are defined in the Connectivity configuration and that are related to the new Provisioning configuration.

You will also have a new login profile for your new Provisioning configuration for the **DirX Identity Manager** that allows managing the objects in the new Provisioning configuration.

4.3. Running the Use Case

After installation and configuration, your solution is ready to run. You can now configure and create all necessary objects. This step is highly dependent on your custom Identity Management solution. Read the corresponding guides to perform these tasks.

5. Multiple Provisioning Domains / Separate Connectivity Domains

An installation with multiple Provisioning domains, where each of them has a separate corresponding Connectivity domain, requires installing and deploying one Single Domain use case on an additional physical machine or virtual machine for each domain. All of these deployments are completely independent of one another. See the chapter "Single Domain" for the installation information.

6. Multiple Tenants / Single Provisioning Domain

The preceding use cases described in this document demonstrate how to model DirX Identity's multi-tenant capability using separate one-domain/one-machine configurations for each tenant or separate Provisioning domains for each tenant.

This use case demonstrates how to model multi-tenancy within a single Provisioning domain, a requirement that has emerged with cloud computing.

6.1. About this Use Case

This use case is an extension to the My-Company sample domain provided with DirX Identity. It demonstrates how to host multiple tenants within a single Provisioning domain, which is the My-Company domain for this use case. In the use case, each tenant in the My-Company domain is modeled as a separate company – CompanyA, CompanyB, and Company C.

Each company has the following characteristics:

- Each company is managed by an administrator called a "company admin".
- Each company admin has access to the DirX Identity Web Center for user and privilege management.
- Each company admin can create, read, modify and delete users of his company. Users of other companies are not visible to him.
- Each company admin grants privileges of his company to users of his company. Privileges of other companies are not visible to him.
- Each company admin can maintain the privilege structure (role, permissions, and groups) of his company. Privileges of other companies are not visible to him.
- Each company admin can manage groups of shared target systems. The shared target systems can be accessed by multiple company admins.
- Each company admin can maintain the business object structure (company, departments, countries, and locations) of his company. Business objects of other companies are not visible to him.

The use case achieves multi-tenancy in a single Provisioning domain by using special structures for users and privileges, special creation workflows and special access policies. The extension requires the DirX Identity Professional Suite license and installation because it requires the creation workflows that are part of this package.

The next sections describe these objects and how they are structured and configured. This description represents the configuration steps to be performed for each tenant in a multi-tenant Provisioning domain.

6.1.1. Access Policies

The use case is designed to use a small number of generic access policies whenever possible. For this purpose, the group **CompanyAdmins** is created in the **DirXmetaRole** target system. All users who are members of this group are allowed to manage users of their company.

The access policies provided with the use case are located under the following subtrees:

- **cn=Generic,cn=Company Specific,cn=AccessPolicies,cn=Policies** - holds generic access policies that apply to all companies.
- **cn=**company***,cn=Company Specific,cn=AccessPolicies,cn=Policies* – holds each company's access policies.

The following figure shows the company-specific policies for CompanyA and the generic policies that are common to all three companies:

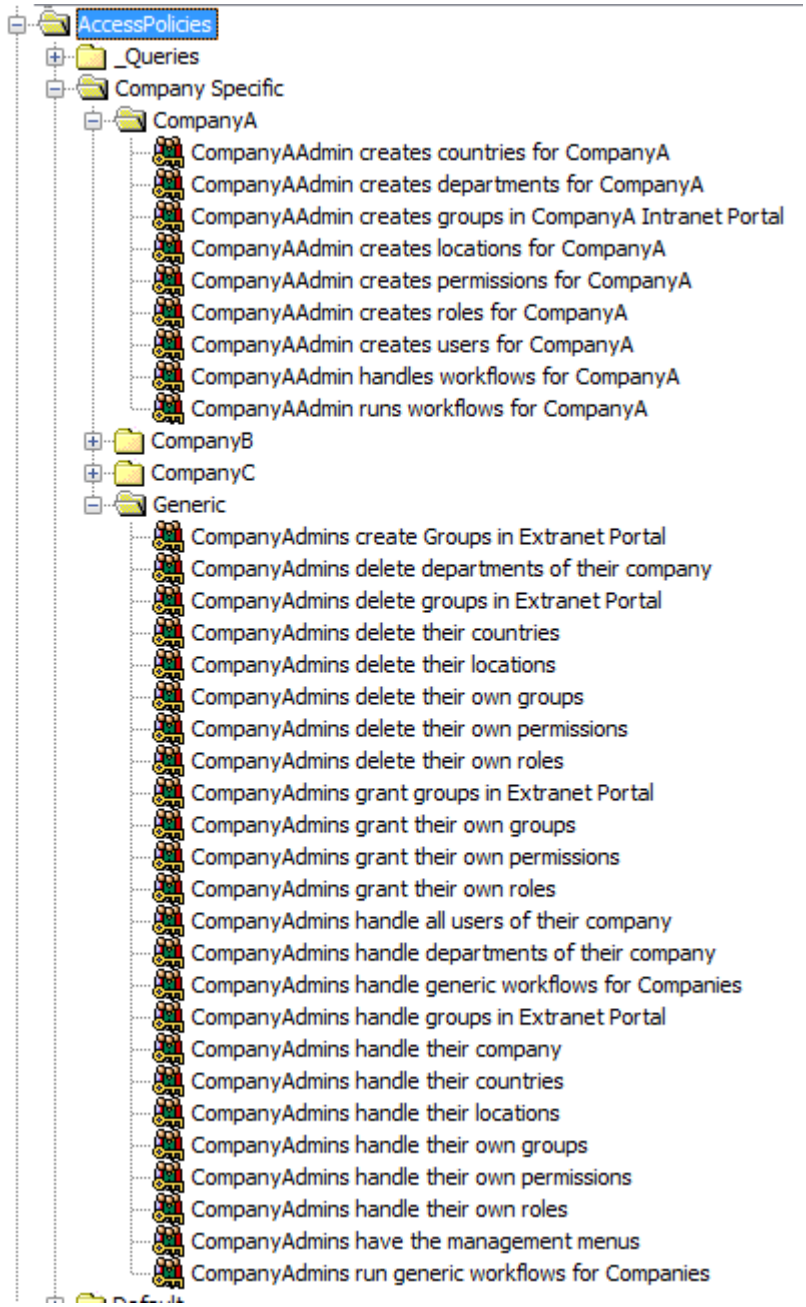

Figure 7. CompanyA and Generic Access Policies

6.1.1.1. Menu Policies

All company admins need access to the Web Center menus for user and privilege management, which is enabled by the generic menu policy **CompanyAdmins have the management menus**. The result of this access policy is that all company admins can use the Users, Roles, Permissions, Groups, Companies, Departments, Countries and Localities menus in Web Center.

6.1.1.2. Create Policies

The create policy holds the path where the object is to be created. Thus, each company and each object type needs its own company-specific create policy configured. The figure shows the create policies that exist for CompanyA.

6.1.1.3. Read and Modify Policies

The use case models read and modify policies in a generic way using the following rules:

- $(subject.o)="$(resource.o)" – for users
- $(subject.dn)="$(resource.owner)" - for privileges and business objects

Note that you need to disable some domain-wide read and modify policies to avoid access to information between different companies inside the single Provisioning domain. For instructions, see the "Setup and Configuraton" section of this use case.

6.1.1.4. Delete Policies

The use case models delete policies in a generic way using the following rules:

- $(subject.o)="$(resource.o)" - for users
- $(subject.dn)="$(resource.owner)" - for privileges and business objects

6.1.2. User Management

Each company has its own subtree **o=*company***,cn=Users,cn=$(domain)**, where *company* is a placeholder for the company's name (for example, CompanyA) and *$(domain)** stands for the Provisioning domain name (My-Company in this use case).

For each company, a company admin is created: **cn=***company***Admin,o=***company***,cn=Users,cn=$(domain)**

The following figure shows the users provided with the use case:

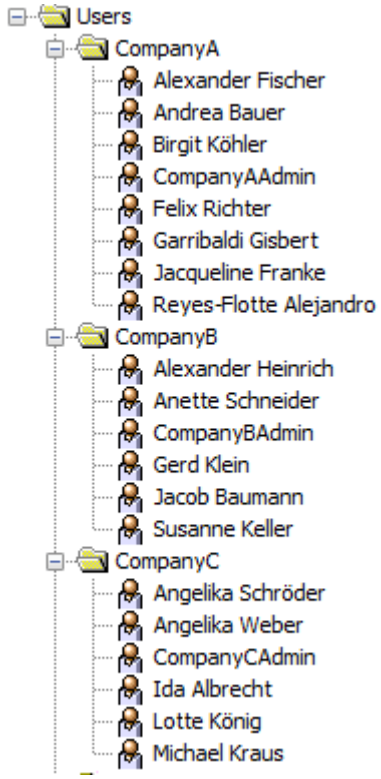

Figure 8. Use Case User Tree

Each company sample consists of a company admin (for example, CompanyAAdmin), and a handful of test users.

6.1.2.1. User Creation

The new users must be created under separate subtrees:

o=*company***,cn=Users,cn=$(domain)**

Each company admin needs a company-specific access policy:

CompanyAdmin **creates users for** *company*

This access policy must be configured to hold the path for user creation **o=** *company***,cn=Users,cn=$(domain)**.

Request workflows (under the Professional License) carry out user creation.

Each company needs its own user creation workflow configured because the new user's parent folder must be configured in the workflow's enter attributes activity (User Base), which is different for different companies.

Two company-specific access policies are required to make sure that the correct creation workflow is started for the company admin:

- *CompanyAdmin* handles workflows for *Company*
- *CompanyAdmin* runs workflows for *Company*

Both of these policies reference the folder that holds the company-specific workflows:

cn=*Company***,cn=Company Specific,cn=Definitions,cn=wfRoot,cn=$(domain)**

Note that the user creation workflow has an activity *AddCompany*. In this activity, the dxrOrganizationLink is set, and points to the company object that is related to the new user's company. This configuration ensures that the new user's organization attribute **o** is correctly populated when the user is saved so that the generic access policies for read/modify will operate correctly.

6.1.2.2. Read and Modify Users

Reading and modifying users is controlled by the generic access policy **CompanyAdmins handle all users of their company**.

The policy grants the read and modify right for a company admin for all users of his company. Using the rule $(subject.o)="$(resource.o)", which means that the subject (the company admin) and the resource (the considered user) must be in the same organization.

6.1.2.3. Assign Privileges to Users

In this sample use case, the following generic access policies configure the assignment of roles and/or groups to a user:

- **CompanyAdmins grant their own roles**
- **CompanyAdmins grant their own groups**

The policies grant the assign right for a company admin for all privileges he owns using the rule $(subject.dn)="$(resource.owner)", which means that the subject's (the company admin's) dn and the resource's (the considered privilege's) owner must match. To make sure that this rule applies to all privileges the company admin creates, he is assigned as owner to the new privileges in an **AddOwner** activity of the creation workflow.

6.1.3. Role Management

The company's roles are located under the following subtrees:

- **cn=**company***,cn=Company Specific,cn=RoleCatalogue,cn=$(domain)*
- **cn=Generic,cn=Company Specific,cn=RoleCatalogue,cn=$(domain)**

The generic folder contains the role **CompanyAdmin**, which gives the right to administer the company to which the user belongs. This role must be assigned to the company admin during the company's initial setup.

Since no grant policy for this role is configured, a company admin cannot grant the administration right to another user of his company.

The company-specific role **CompanyIntranet** is part of the sample. It gives a user of the company access to the company's intranet portal.

6.1.3.1. Role Creation

Each company's new roles must be created under separate subtrees:

cn=*company***,cn=Company Specific,cn=RoleCatalogue,cn=$(domain)**

The company admin needs a company-specific access policy:

CompanyAdmin **creates roles for** *company*

The path for role creation (**cn=**company***,cn=Company Specific,cn=RoleCatalogue,cn=$(domain)*) is configured in this access policy.

Request workflows (under professional license) carry out role creation. Each company must have its own role creation workflow configured, because the new role's parent folder must be configured in the workflow's enter attributes activity (**User Base**) and is different for different companies.

To make sure that the correct creation workflow is started for the company admin, the following company-specific access policies are required:

- *CompanyAdmin* **handles workflows for** *Company*
- *CompanyAdmin* **runs workflows for** *Company*

Both policies reference the folder that holds the company-specific workflows:

cn=*Company*,cn=Company Specific,cn=Definitions,cn=wfRoot,cn=$(domain)

Note that the role creation workflow has an activity **AddOwner**. In this activity, the company admin that starts the workflow is set as the role, which ensures that the generic access policies for read/modify will operate correctly.

6.1.3.2. Read and Modify Roles

Reading and modifying roles is controlled by the generic access policy **CompanyAdmins handle their own roles**.

The policy grants the read and modify right for a company admin for all roles he owns with the rule $(subject.dn)="$(resource.owner)" , which means that the subject (the company admin) must be the owner of the resource (the considered role).

6.1.3.3. Delete Roles

Deleting roles is controlled by the generic access policy **CompanyAdmins delete their own roles**.

The policy grants the delete right for a company admin for all roles he owns with the rule $(subject.dn)="$(resource.owner)", which means that the subject (the company admin) must be the owner of the resource (the considered role) he wants to delete.

6.1.3.4. Assign Junior Roles and Permissions

The company admin can assign junior roles and permissions to a role.

First, he needs to read the role he wants to process. This access is controlled by the policy:

CompanyAdmins handle their own roles

Next, he needs to assign a role or a permission to the selected role. This access is controlled by the policies :

- **CompanyAdmins grant their own roles**
- **CompanyAdmins grant their own permissions**

These policies ensure that the company admin can only assign roles and permissions of his company. He cannot view or assign privileges that belong to another company.

6.1.4. Permission Management

Each company's permissions are located under the following subtrees:

- **cn=**company***,cn=Company Specific,cn=Permissions,cn=$(domain)*
- **cn=Generic,cn=Company Specific,cn=Permissions,cn=$(domain)**

The generic folder contains the permission **CompanyAdmin**, which gives the right to

administer the company to which the user belongs. This permission is assigned to the role with the same name.

Since no grant policy for this role is configured, a company admin cannot grant the administration right to another user of his company.

The company-specific permission **CompanyIntranet** is part of the sample. It is assigned to the role with the same name.

6.1.4.1. Permission Creation

Permission creation is governed by the same mechanisms as role creation, as described in “Role Management/Role Creation”.

6.1.4.2. Read and Modify Permissions

Permission handling is governed by the same mechanisms as role handling, which is described in “Role Management/Read and Modify Roles”.

6.1.4.3. Delete Permissions

Permission deletion is governed by the same mechanisms as role deletion, which is described in “Role Management/Delete Roles”.

6.1.4.4. Assign Groups

The company admin can assign groups to a permission.

First, he needs to read the permission he wants to process, which is controlled by the policy:

CompanyAdmins handle their own permissions.

Next, he needs to assign the group to the selected permission, which is controlled by the policy:

CompanyAdmins grant their own groups.

In the sample, this policy grants all groups of the target system *company* **Intranet Portal**.

This use case requires a shared target system, which means that all company admins can create, handle and grant groups of this target system. In the sample, the My-Company sample‘s **Extranet Portal** target system is configured as a shared target system.

This setup requires the generic access policies:

- **CompanyAdmins create groups in Extranet Portal**
- **CompanyAdmins handle groups in Extranet Portal**
- **CompanyAdmins grant groups in Extranet Portal**

6.1.5. Group Management

The company's groups are located in the target system:

company **Intranet Portal**.

In addition, each company admin has access to the groups of the shared **Extranet Portal** target system.

6.1.5.1. Group Creation

Group creation is governed by the same mechanisms as role creation, which is described in the section "Role Management/Role Creation".

In addition to creating groups in the company-specific target system *company* **Intranet Portal**, it is requested to create groups in the shared target system **Extranet Portal**, too.

Consequently, the generic workflow **Create Group in Extranet Portal** has been configured.

The following execute policies are required to enable the company admin to start the workflows:

- **CompanyAdmins handle generic workflows for companies**
- **CompanyAdmins run generic workflows for companies**.

6.1.5.2. Read and Modify Groups

Handling of groups is governed by the same mechanisms as role handling, as described in "Role Management/Read and Modify Roles".

In addition to handling groups in the company-specific target system *company* **Intranet Portal**, it is requested to handle groups in the shared target system **Extranet Portal**, too.

The generic policy:

CompanyAdmins handle groups in Extranet Portal

grants all company admins the read and modify right for the groups.

6.1.6. Companies

The companies are located under the subtree:

cn=Companies,cn=BusinessObjects,cn=$(domain)

6.1.6.1. Read and Modify Companies

A company admin can read and modify his company, but he cannot create a new company. Thus, one generic access policy is sufficient, which is:

CompanyAdmins handle their company

The policy uses the rule $(subject.o)="$(resource.o)", which also applies to handling users.

6.1.6.2. Assign Privileges

A company admin can assign privileges to his company. This access is governed by the access policies:

- **CompanyAdmins grant their own roles**
- **CompanyAdmins grant their own permissions**
- **CompanyAdmins grant their own groups**
- **CompanyAdmins grant groups in Extranet Portal**

Note that all privileges assigned to a company are propagated to all users of the company by the business object inheritance mechanism.

6.1.7. Departments

The departments are located under the subtree:

O=*company*,cn=Companies,cn=BusinessObjects,cn=$(domain)*

6.1.7.1. Department Creation

Department creation is governed by the same mechanisms as role creation, which is described in “Role Management/Role Creation”.

6.1.7.2. Read and Modify Departments

Department handling is governed by the same mechanisms as role handling, which is described in “Role Management/Read and Modify Roles”.

6.1.7.3. Assign Privileges

A company admin can assign privileges to a department. This access is governed by the access policies:

- **CompanyAdmins grant their own roles**
- **CompanyAdmins grant their own permissions**
- **CompanyAdmins grant their own groups**
- **CompanyAdmins grant groups in Extranet Portal**

Note that all privileges assigned to a department are propagated to all users of the department by the business object inheritance mechanism.

6.1.8. Countries

The countries are located under the subtree:

cn=*company*,cn=Countries,cn=BusinessObjects,cn=$(domain)*

In this structure, each company has its own country objects to group the locations of the company.

6.1.8.1. Country Creation

Country creation is governed by the same mechanisms as role creation, as described in “Role Management/Role Creation”.

6.1.8.2. Read and Modify Countries

Country handling is governed by the same mechanisms as role handling, as described in “Role Management/Read and Modify Roles”.

6.1.9. Locations

The locations are located under the subtree:

c=*country***,cn=***company***,cn=Countries,cn=BusinessObjects,cn=$(domain)**

6.1.9.1. Location Creation

Location creation is governed by the same mechanisms as role creation, as described in “Role Management/Role Creation”.

6.1.9.2. Read and Modify Locations

Location handling is governed by the same mechanisms as role handling, as described in “Role Management/Read and Modify Roles”.

6.1.10. Request Workflows

This use case extension to the My-Company sample domain requires the DirX Identity Professional Suite, because it requires the request workflows that are part of this license package in order to create users, roles, permissions and groups.

6.1.10.1. Locating Request Workflows

The request workflow definitions for a company are located under the subtree:

cn=*company***,cn=Company Specific,cn=Definitions,cn=wfRoot**

The following figure shows the company-specific workflows of CompanyA and the generic group creation workflow:

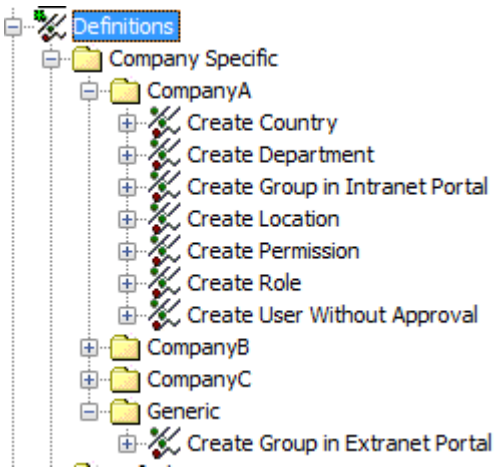

Figure 9. Use Case Workflow Tree

6.1.10.2. User Creation Workflows

The user creation workflows are derived from the sample workflow **Create User Without Approval**. This workflow has been modified to:

- Create the users in the correct company subfolder.
- Automatically add the dxrOrganizationLink, which points to the user's company object, to the new user.

The first task is achieved by changing the **User Base** parameter of the **Enter Attributes** activity to the value **o=*company***,cn=Users,cn=$(domain)*.

The second task is achieved by adding a calculate activity **Add** *company* to the workflow. In this activity, the value:

o=*company***,cn=Companies,cn=BusinessObjects,cn=$(domain)**

is assigned to the user's dxrOrganizationLink attribute.

6.1.10.3. Role Creation Workflows

The role creation workflows are derived from the sample workflow **Create Role**. This workflow has been modified to

- Create the roles in the correct company subfolder.
- Automatically add the owner to the new role.

The first task is achieved by changing the **User Base** parameter of the **Enter General Attributes** activity to the value:

o=*company***,cn=Company Specific,cn=RoleCatalogue,cn=$(domain)**.

The second task is achieved by adding a calculate activity **Add Owner** to the workflow. In this activity, the value:

${initiator}

is assigned to the role's owner attribute. Since the workflow is started (initiated) by the company admin, he will be the owner of the newly created role.

The role creation workflow has an approval step, where a person outside the company must approve role creation. This step demonstrates that an administrator of the entire system can control object creation.

You can remove the approval step or change the participants according to your needs.

6.1.10.4. Permission Creation Workflows

The permission creation workflows are derived from the role creation workflows and simplified by removing the approval step. The workflows have been modified to:

- Create the permissions in the correct company subfolder.
- Automatically add the owner to the new permission.

The modifications are the same as those described in the "Role Creation Workflows" section.

6.1.10.5. Group Creation Workflows

The group create workflows are derived from the permission creation workflows. The workflows have been modified to:

- Create the groups in the correct target system.
- Automatically add the owner to the new group.

The modifications are the same as those described in the "Role Creation Workflows" section.

Two types of group creation workflows exist:

- The company-specific workflows **Create Group in Intranet Portal** for creating a group in the target system *company* **Intranet Portal**.
- The generic workflow **Create Group in Extranet Portal** for creating groups in the shared target system **Extranet Portal**.

6.1.10.6. Department Creation Workflows

The department create workflows are derived from the permission creation workflows. The workflows have been modified to:

- Create the department in the correct company subfolder.
- Automatically add the owner to the new department.

The modifications are the same as those described in the "Role Creation Workflows" section.

6.1.10.7. Country Creation Workflows

The country creation workflows are derived from the permission creation workflows. The workflows have been modified to:

- Create the country in the correct company subfolder
- Automatically add the owner to the new country.

The modifications are the same as those described in the "Role Creation Workflows" section.

6.1.10.8. Location Creation Workflows

The location creation workflows are derived from the permission creation workflows. The workflows have been modified to:

- Create the location in the correct country subfolder.
- Automatically add the owner to the new location.

The modifications are the same as those described in the "Role Creation Workflows" section.

6.2. Setup and Configuration

You can find the extension data for this use case in the path:

*install_path***data\extension\SimpleTenancySample.ldif**.

You can import the extension into the My-Company Provisioning domain. Only CompanyA has the complete set of objects and policies built for the use case. After you import the LDIF file into the My-Company Provisioning domain, you will have the following items:

- The Company-Specific folder and its sub folders
- The set of sample objects described in "About this Use Case".
- Configure the CompanyA sample according to your requirements.
- Customize Web Center (optional).
- Test and debug your Company A configuration.
- Set up your additional tenants using CompanyA as a template.

6.2.1. Configure the Company A Sample

The first step is to configure the Company A sample according to your requirements. This step includes:

- Configuring the sample creation workflows. See the section "Understanding the Creation Workflows" in the *DirX Identity Application Development Guide* for details.
- Making the customizations you would normally make to a Provisioning domain that

you set up to run a single tenant, such as:

- Customizing the sample privilege structure: the roles, permissions and groups.
- Adding any target systems to be separated from or shared by your planned tenants. See the chapter "Managing Target Systems" in the *DirX Identity Provisioning Administraton Guide* for details.

See the *DirX Identity Provisioning Administration Guide* for details.

6.2.2. Customize Web Center

Although you can use the default Web Center configuration with this use case, you may have customizations that you want to make to Web Center. See the *DirX Identity Web Center Customization Guide* for details.

6.2.3. Test the Company A Sample

Now you should test and adjust the function of CompanyA so that it operates according to the configuration described in "About this Use Case".

6.2.4. Set up a New Company from CompanyA

When you have successfully customized and configured the CompanyA sample, you can use it as a template to set up your other tenants. This section describes how to create a new tenant named **NewCompany** using the CompanyA template provided with this use case.

6.2.4.1. Create a Company-Specific User Folder

To create the company-specific user folder:

- Create folder **o=NewCompany,cn=Users,cn=$(domain)**. This folder will contain the users of NewCompany.
- Use **CopyObject** in the Identity Manager to copy CompanyAAdmin from CompanyA to the new user folder, and then rename it to NewCompanyAdmin. Uncheck the “Use as Template” checkbox in the “Operational” tab and then add this new user to the group **CompanyAdmins** in the **DirXmetaRole** target system.
- Set a password for NewCompanyAdmin in the Data View.

6.2.4.2. Create the Company-Specific Access Policies

To create the company-specific access policies:

- Create a folder **cn=NewCompany,cn=Company Specific,cn=AccessPolicies,cn=Policies,cn=$(domain)**
- Use **CopyObject** in the Identity Manager to copy the policies to the new folder step by step. In each policy, replace the strings **CompanyA** with **NewCompany**, yielding, for example, **NewCompanyAdmin creates countries for NewCompany**.
- In the Subjects tab of each policy, replace person **CompanyAAdmin** with **NewCompanyAdmin**.

- In the Resources tab of each policy, replace **CompanyA** with **NewCompany**.

6.2.4.3. Create the Company-Specific Role and Permission Subfolders

Create the folders:

cn=NewCompany,cn=Company Specific,cn=RoleCatalogue,cn=$(domain)

and

cn=NewCompany,cn=Company Specific,cn=Permissions,cn=$(domain).

6.2.4.4. Create the Company-Specific Target System(s)

Create the target system:

cn=NewCompany Intranet Portal,cn=TargetSystems,cn=$(domain)

(or any other target system required for the NewCompany).

6.2.4.5. Create the Company NewCompany

Create the company object:

o=NewCompany,cn=Companies,cn=BusinessObjects,cn=$(domain)

6.2.4.6. Create the Parent Folder for Countries

Create the folder for the countries of the NewCompany:

cn=NewCompany,cn=Countries,cn=BusinessObjects,cn=$(domain).

6.2.4.7. Create the Request Workflows for NewCompany

Create a folder for the request workflow definitions of NewCompany:

cn=NewCompany,cn=Company Specific,cn=Definitions,cn=wfRoot,cn=$(domain)

Use **CopyObject** in the Identity Manager to copy the request workflows of CompanyA to the new folder.

In each workflow's "Enter [General] Attributes" activity, adapt the **User Base** parameter for NewCompany.

6.2.4.8. Deactivate Domain-Wide Read and Modify Policies

You need to disable the domain-wide read and modify policies that may cause information sharing between the different companies in the one Provisioning domain.

Deactivate the following policies:

cn=Users can read locations,cn=Read and Modify Policies,cn=My-

Company,cn=AccessPolicies,cn=Policies,cn=My-Company

cn=Users can read Organizations,cn=Read and Modify Policies,cn=My-Company,cn=AccessPolicies,cn=Policies,cn=My-Company

cn=Users can read OUs,cn=Read and Modify Policies,cn=My-Company,cn=AccessPolicies,cn=Policies,cn=My-Company

cn=Anyone can read locations,cn=Read and Modify Policies,cn=My-Company,cn=AccessPolicies,cn=Policies,cn=My-Company

6.2.4.9. Remove the Search Base Option in Web Center

To avoid displaying the structures of all companies inside the single Provisioning domain (for example, OUs, locations, users and so on), remove the search base used in Web Center and then set it to a search base that points directly to the company structure.

6.2.4.10. Complete the Setup

To complete the NewCompany setup, reload the IdS-J configuration. Now you can let the NewCompanyAdmin do his work in Web Center.

7. Use Case Deployment Quick Reference

This chapter describes step-by-step setup and removal procedures that you can use for the use cases described in this document.

The information and instructions given here have been extracted from information and procedures in the DirX Identity documentation set and tailored to the specifics of deploying and maintaining these use cases. The referenced DirX Identity documentation includes:

- *Installation Guide* – the sections "Using the Configurator" and "Installing Single Components".
- *Connectivity Administration Guide* – in the chapter "Managing DirX Identity Servers", the sections "Managing the Java-based Server" and "Distributed Deployment and Scalability".
- *High Availability* Use Case Document – the subsection "Java-based Server" in the "Configuration" section.

Note that installation and removal procedures and configuration details may change as new versions of DirX Identity are released. Be sure to consult the reference documentation listed above for the latest installation and removal procedures and configuration details.

7.1. Deploying a Complete Additional Domain

The following sections describe how to set up an additional domain and the first related Java-based Server and Web Center.

Note that we do not describe SSL configuration issue here. See the *DirX Identity Connectivity Administration Guide* for information about this topic.

To set up the complete additional domain:

1. Launch the configuration tool:

   Start → **All Programs** → **Atos DirX Identity V*n.n* → *Configuration**
2. Click **Next** to proceed to the **Configuration Options** step. Check that the **Domain Configuration**, **Java-based Server** and **Web Center** options are selected.
3. Proceed to the next step **Directory Server for Connectivity** and enter the host and port information of the directory server where your Connectivity configuration resides.
4. Proceed to the next step **Directory Server for Provisioning** and enter all necessary data for the directory server where your Provisioning configuration resides.
5. Click **Next** to proceed to the **DirX Identity Administrators** dialog and enter the information for both administrators.
6. Click **Next** to proceed to the **Domain Configuration** step. Select **Configure a customer domain**.
7. The configuration process displays input fields for specifying the data for your domain.

In the **Domain** field, specify your domain name.

8. Review the value in the **Technical Domain Name** field. You can change this name, but we recommend keeping it as similar as possible to the domain name you previously entered.
9. Add a password for the account **DomainAdmin** and store it in a safe place.
10. Click **Next** to proceed to the **Java-based Server** step. Be sure that <Create a new Java Server> is selected in the field **Server to update or create**.

    Specify **Port** and **JMX port** so that they are different from each other and from the ports used for the other co-located Java-based Servers.

    Specify the **Dynamic port range** so that it does not conflict with any other ports.
11. Click **Next** to proceed to the **Web Center Information** step. In this step, you specify the appropriate data for Web Center. This step is optional. (See the subsection "Web Center Information" in the section "Using the Configurator" in the chapter "Configuring DirX Identity" in the *DirX Identity Installation Guide* for details.)
12. Click **Next** to proceed to the **Pre-Configuration Summary** step. Carefully review your input data in this summary and then start the configuration process by clicking **Next**.

Wait until the configuration process is complete. Click **Finish**.

The following sections provide information about the results of the configuration process.

7.1.1. Post-Configuration Results - Additional Java-based Server

On Windows platforms, the configurator creates:

- The service **IdS-J-***technical_domain_name***-S1** *version*. For example:

DirX Identity IdS-J-My-Company-S1 V8.3

- The file system *install_path***/ids-j-***technical_domain_name***-S1** for the new Java-based Server.

On Linux platforms, the configurator creates:

- The script **S99dmsvr-***technical_domain_name***-S1** in the folder *install_path*/**etc** for starting and stopping the related Java-based Server.
- Command to start this Java-based Server:
  S99dmsvrj-*technical_domain_name***-S1 start**
- Command to stop this Java-based Server:
  S99dmsvrj-*technical_domain_name***-S1 stop**
- The file system *install_path***/ids-j-***technical_domain_name***-S1** for the new Java-based Server.

7.1.2. Post-Configuration Results - Web Center

After configuring the Web Center for a domain (also for a first Web Center), the configuration process results are:

- A Web Center instance for the related domain accessible by the URL **http://*host*.*tomcatport***/webCenter-**technical_domain_name*. On UNIX platforms, you must restart tomcat before you are able to access Web Center.
- The file system *install_path**/web/webCenter-**technical_domain_name*
- The configuration file **webCenter-*technical_domain_name*.xml in the subfolder *conf/Catalina/localhost** of the tomcat installation.

7.2. Deploying Additional Components

There are several options that you can configure freely as required. For any of the use cases described in this document, you can:

- Set up additional Java-based Servers for each of the configured domains to distribute load and separate applications.
- Set up additional Web Center applications.

You may want to add more of these components to improve the performance of the solution that you've set up according to the use cases described in this document. The next sections describe how to add these components. For a detailed description of scalability and distribution in DirX Identity, see the following documentation:

- *High Availability* Use Case Document
- The section "Distributed Deployment and Scalability" in the chapter "Managing DirX Identity Servers" in the *DirX Identity Connectivity Administration Guide*.

These documents describe the methods that are available for distributing process workload and separating processes.

7.2.1. Setting up Additional Java-based Servers

This procedure consists of two steps:

- Setting up the Java-based Server
- Distributing the workflow types and adaptors

7.2.1.1. Setting up the Java-based Server

For each additional Java-based Server you want to create:

- Run the Configurator (either **Configuration** or **Initial Configuration**).
- Select the **Java-based Server Configuration** from the **Configuration Options**.
- Define the domain name in the **Domain Configuration** step.

- Select **Create a new Java Server** from the dropdown list of the **Server to update or create** field of the **Java-based Server** step.
- Define the relevant parameters. Be careful to use free ports.

7.2.1.2. Distributing Workflow Types and Adaptors

You can configure each Java-based Server for specific workflow types or for specific adaptors. For example, you can configure one Java-based Server to handle request workflows, a second Java-based Server to handle user and account password change workflows, a third one to handle entry change workflows and a fourth one to handle provisioning request workflows.

For each Java-based Server to be created and configured:

- Start DirX Identity Manager (**Connectivity** view group) and then select **Expert View.**
- Open **Configuration** → **DirX Identity Servers** → **Java Servers**. You should see all your configured server instances and if you open them all created adaptors.
- Select **Manage IdS-J Configuration** from the context menu of a Java Server node.
- In the Adaptors tab, activate the relevant adaptors for this service instance and deactivate the ones that should not be active. By default, all adaptors are activated.
- In the Request Workflow Timeout Check tab, select exactly one server to run the Request Workflow Timeout Check.
- In the Scheduler tab, select exactly one server to run the Scheduler.
- Click **OK** to store the configuration or **Cancel** to abort it.
- Restart all Java-based Servers to load the updated configuration. Perform the restart by stopping all Java-based Servers; after the last server is stopped, start them all again.
- Use Web Admin to check that the adaptors are configured correctly.

7.2.2. Setting Up Web Center Applications

You can use the Configuration tool to set up one Web Center application for one domain. If you want to set up additional Web Center applications to distribute load or tasks, you can do it manually by performing the steps described below in the manual configuration section.

7.2.2.1. Setting Up a Web Center Application with the Configurator

- Run the Configurator (either **Configuration** or **Initial Configuration**).
- Select the **Web Center Configuration** from the **Configuration Options**.
- Define the domain name in the **Domain Configuration** step.
- Finish the configuration.
- Repeat this procedure for every domain where a Web Center has not already been configured.

The Configurator extracts all Web Center files in webManager.zip to the domain specific

folder *install_path*\web***webCenter-** *domain*, configures the files web.xml and webCenter.xml with some necessary parameters like server name, user bind name and tomcat path and then deploys the application to Tomcat.

7.2.2.2. Setting Up an Additional Application Manually

- Extract **webManager.zip** to your desired *folder_name* under *install_path***\web.**
- Copy **webCenter.xml** to *folder_name***.xml**.
- Specify the folder in **docBase** of *folder_name***.xml**.
- Deploy *folder_name***.xml** to Tomcat.
- Adapt the Tomcat path in *install_path***\web** *folder_name***\endorsed\postInstallWebMgr.bat** (or .sh) and then call the script to update some jar files in the Tomcat-endorsed folder.
- Adapt the parameters in *install_path***\web***folder_name***\Web-inf\web.xml** to your needs.
  For a detailed description of the **web.xml** parameters, see the chapter "Using DirX Identity Web Center → Configuring the Web Center → Using the Web Center Configuration File" in the *DirX Identity User Interfaces Guide*.
- Configure the bind passwords in *install_path*\ **web*folder_name*\Web-inf\password.properties.**

Removing Additional Components

The following procedures help you to cleanse your environment of nonessential Java-based Servers or Web Center instances.

7.3. Removing an Additional Component

This section provides instructions for unconfiguring and removing an additional Java-based Server and an additional Web Center application.

7.3.1. Removing a Java-based Server

Before you remove a Java-based Server, be sure to move all tasks of this server to one of the remaining servers.

7.3.1.1. Windows Platform

To unconfigure an additional Java-based Server:

1. Stop the related Java-based Server.
2. Unregister the service:
   Run the script **unregisterNTServer.bat** in the folder *install_path***/ids-j-***technical_domain_name***-S***n***/bin**.
3. Remove the related Java-based Server configuration entry.
4. Delete the corresponding file folder in the installation area.

7.3.1.2. Linux Platform

To unconfigure an additional Java-based Server:

1. Stop the related IdS-J Server:
   Run the script **S99dmsvrj-***technical_domain_name* in the folder *install_path***/etc**.
2. Remove the script **S99dmsvrj**-*technical_domain_name* from the folder *install_path* **/etc**.
3. Remove the related Java-based Server configuration entry.
4. Delete the corresponding file folder in the installation area.

7.3.2. Removing an Additional Web Center

To unconfigure an additional Web Center:

1. Stop the Tomcat service used by the Web Center.
2. Remove the configuration file **webCenter-***technical_domain_name***.xml** from the subfolder **conf/Catalina/localhost** of the tomcat installation.
3. Delete the corresponding file folder under the **web** subfolder in the installation area.

DirX Product Suite

The DirX product suite provides the basis for fully integrated identity and access management; it includes the following products, which can be ordered separately.

DirX Identity

DirX Identity provides a comprehensive, process-driven, customizable, cloud-enabled, scalable, and highly available identity management solution for businesses and organizations. It provides overarching, risk-based identity and access governance functionality seamlessly integrated with automated provisioning. Functionality includes lifecycle management for users and roles, cross-platform and rule-based real-time provisioning, web-based self-service functions for users, delegated administration, request workflows, access certification, password management, metadirectory as well as auditing and reporting functionality.

DirX Directory

DirX Directory provides a standards-compliant, high-performance, highly available, highly reliable, highly scalable, and secure LDAP and X.500 Directory Server and LDAP Proxy with very high linear scalability. DirX Directory can serve as an identity store for employees, customers, partners, subscribers, and other IoT entities. It can also serve as a provisioning, access management and metadirectory repository, to provide a single point of access to the information within disparate and heterogeneous directories available in an enterprise network or cloud environment for user management and provisioning.

DirX Access

DirX Access is a comprehensive, cloud-ready, scalable, and highly available access management solution providing policy- and risk-based authentication, authorization based on XACML and federation for Web applications and services. DirX Access delivers single sign-on, versatile authentication including FIDO, identity federation based on SAML, OAuth and OpenID Connect, just-in-time provisioning, entitlement management and policy enforcement for applications and services in the cloud or on-premises.

DirX Audit

DirX Audit provides auditors, security compliance officers and audit administrators with analytical insight and transparency for identity and access. Based on historical identity data and recorded events from the identity and access management processes, DirX Audit allows answering the “what, when, where, who and why” questions of user access and entitlements. DirX Audit features historical views and reports on identity data, a graphical dashboard with drill-down into individual events, an analysis view for filtering, evaluating, correlating, and reviewing of identity-related events and job management for report generation.

For more information: support.dirx.solutions/about

Legal remarks

On the account of certain regional limitations of sales rights and service availability, we cannot guarantee that all products included in this document are available through the Eviden sales organization worldwide. Availability and packaging may vary by country and is subject to change without prior notice. Some/All of the features and products described herein may not be available locally. The information in this document contains general technical descriptions of specifications and options as well as standard and optional features which do not always have to be present in individual cases. Eviden reserves the right to modify the design, packaging, specifications and options described herein without prior notice. Please contact your local Eviden sales representative for the most current information. Note: Any technical data contained in this document may vary within defined tolerances. Original images always lose a certain amount of detail when reproduced.