EVIDEN

Identity and Access Management

DirX Identity

Creating a Custom Target System Type

Version 9.0.3, Edition October 2026

All product names quoted are trademarks or registered trademarks of the manufacturers concerned.

© 2026 Atos Group

All Rights Reserved

Distribution and reproduction not permitted without the consent of Atos Group.

Table of Contents

Copyright . . . ii
Preface . . . 1
DirX Identity Documentation Set . . . 2
Notation Conventions . . . 4
1. Overview . . . 6
1.1. Use Cases . . . 6
1.1.1. Defining a Custom Target System Instance . . . 6
1.1.2. Defining a Custom Target System Template . . . 6
1.1.3. Creating Java-based Provisioning Workflows . . . 7
1.2. Use Case Comparison . . . 7
1.3. Evaluation of the System to Connect . . . 7
1.3.1. Representation of Identities . . . 7
1.3.1.1. How Does the System Represent Identities? . . . 7
1.3.1.2. How Many Accounts Exist and What is the Typical Change Frequency? . . . 8
1.3.1.3. Can Accounts be Disabled? . . . 8
1.3.2. How does the system model access control? . . . 8
1.3.2.1. Access control model . . . 8
1.3.2.2. Does the System Allow For Nested Groups? . . . 9
1.3.3. Structural Conditions . . . 9
1.3.3.1. Are Groups and Accounts Stored in a Hierarchical or Flat structure? . . . 9
1.3.3.2. Are Memberships Stored at Accounts or at Groups? . . . 9
1.3.4. Which provisioning interfaces are available? . . . 10
2. Defining a Custom Target System Instance . . . 11
2.1. About this Use Case . . . 11
2.1.1. Documentation Hints . . . 11
2.1.1.1. Tutorial . . . 11
2.1.1.2. User Interfaces Guide . . . 11
2.1.1.3. Application Development Guide . . . 11
2.1.1.4. Customization Guide . . . 11
2.1.1.5. Provisioning Administration Guide . . . 11
2.1.2. Specific Hints and Guidelines . . . 12
2.2. Setup and Configuration . . . 12
2.2.1. Create a Target System Instance of the Generic Type . . . 12
2.2.2. Customizing the Newly Created Target System Instance . . . 13
2.2.2.1. Check and Set Up the Target System Tabs . . . 13
2.2.2.2. Check and Set Up the Target System Configuration . . . 13
2.3. Alternative or Extended Configurations . . . 14
3. Defining a Custom Target System Template . . . 15
3.1. About this Use Case . . . 15

3.2. Compatibility . . . 15
3.2.1. Documentation Hints . . . 15
3.2.2. Specific Hints and Guidelines . . . 15
3.3. Setup and Configuration . . . 16
3.3.1. Prepare a Target System Instance . . . 16
3.3.2. Create Your Custom Target System Template . . . 17
3.3.2.1. Copying the Target System Instance . . . 17
3.3.2.2. Configure the Target System Template . . . 17
3.3.3. Use Your Custom Target System Template . . . 18
3.4. Alternative or Extended Configurations . . . 18
3.4.1. Creating the Custom Target System Template Using a Built-in Template . . . 18
3.4.2. Creating the Custom Target System Template Using a Custom Template . . . 19
4. Creating Java-based Provisioning Workflows . . . 20
4.1. About this Use Case . . . 20
4.1.1. Documentation Hints . . . 20
4.1.1.1. Connectivity Administration Guide . . . 20
4.1.1.2. Connectivity Reference . . . 20
4.1.1.3. Application Development Guide . . . 21
4.1.1.4. User Interfaces Guide . . . 21
4.1.1.5. Use Case Description: Java Programming in DirX Identity . . . 21
4.1.1.6. Integration Framework . . . 21
4.1.2. Specific Hints and Guidelines . . . 21
4.2. Setup and Configuration . . . 22
4.2.1. Creating a Custom Scenario . . . 22
4.2.2. Creating a New Connected Directory . . . 23
4.2.3. Creating the New Java-based Provisioning Workflows . . . 24
4.2.4. Refining the Objects . . . 24
4.2.4.1. Refining the Connected Directory . . . 24
4.2.4.2. Refining the Workflow(s) . . . 25
4.2.4.3. Refining the Channels . . . 26
4.2.5. Testing the Workflows . . . 27
4.2.5.1. Check Correctness . . . 27
4.2.5.2. Live Testing . . . 27
5. Transferring Custom Target System Templates . . . 28
Legal Remarks . . . 30

Preface

This document describes a set of use cases that explain how to use specific features of DirX Identity. It helps users to model their use case with DirX Identity and to set up and run their DirX Identity system.

The goal of this document is to explain how to define and use new target system types for DirX Identity and to prepare the connectivity configuration accordingly.

It consists of the following chapters:

- Chapter 1 provides an overview on the described use cases.
- Chapter 2 explains how to define a single instance of a new target system type.
- Chapter 3 explains how to define a template for a new target system type.
- Chapter 4 describes how to set up the connectivity configuration for Java-based provisioning workflows for newly created target system types.
- Chapter 5 explains how to transfer the templates for new target system types between different environments.

DirX Identity Documentation Set

Version: 9.0.3 | Date: 2026-10-20 | Timestamp: 12:12:45 UTC

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 Resolution Service Setup Guide*. Use this book to set up the DirX Identity Resolution Service as a standalone service.
- *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.
- *DirX Identity REST Service 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

In a typical customer environment for an identity management solution you have to connect standard connected systems like Active Directory or IBM Notes. DirX Identity comes with a comprehensive set of pre-defined target system types and the corresponding provisioning workflows.

In many cases there are other legacy systems or connected system types in place that shall be connected to DirX Identity where the standard connectivity of DirX Identity does not provide a solution. This document explains what needs to be done to connect such systems.

The generic procedure to implement a viable solution is:

- Evaluate the system to connect thoroughly (see the section "Evaluation of the System to Connect" below).
- Define a target system instance or template (described as the first two use cases)
- Implement a connector to access the target system or use an alternative access method. How to implement a connector is not part of this document. Read the *DirX Identity Integration Framework Guide* for more information.
- Configure the necessary provisioning workflows (described as the third use case)

DirX Identity supports this process completely. Correctly defined objects in provisioning and connectivity configuration allow seamless integration of new connected systems into DirX Identity.

1.1. Use Cases

This document describes three use cases in detail. Be aware that other use cases are possible that are not described in this document.

1.1.1. Defining a Custom Target System Instance

This use case allows you to create a new target system instance through customization of a generic built-in target system template. According to the properties and the API of the connected system you have to set up a target system instance together with the necessary connected directory and the provisioning workflows.

This method is applicable if you need only one instance of the new custom target system.

1.1.2. Defining a Custom Target System Template

This use case is a natural extension of the previous one. After definition of a new target system template you can create a set of target system instances.

Typically you will use this case when modeling a more common target system type that is not supported by DirX Identity but shall exist in multiple instances within the company.

1.1.3. Creating Java-based Provisioning Workflows

This use case explains how to build the necessary Java-based provisioning workflows for a custom target system type or one of its instances.

1.2. Use Case Comparison

The following table compares the first two use cases described in this document to help you with your decision process. The third use case is independent from the other two and necessary in any case.

Table 1. Use Case Comparison

Criteria	Custom TS Instance	Custom TS Type
Number of instances required	1	N
Customization depth	Low	High
Re-use necessary	No	Yes

The table presents the following evaluation criteria:

Number of instances required - the number of target system instances the customer needs.

Customization depth - the number of necessary customization regarding one of the existing target system templates. High customization occurs if there are many custom JavaScripts and reports necessary.

Re-use necessary - in hosting environments it might be necessary to copy the target system templates to various domains.

1.3. Evaluation of the System to Connect

Before you start any implementation, evaluate the following questions and determine the requirements that are necessary to implement connectivity to the system you want to connect to.

Additionally, answering these questions can help to choose an existing target system type as a starting point for your custom target system instance or type.

1.3.1. Representation of Identities

It is important to understand how the system represents identities which means users.

1.3.1.1. How Does the System Represent Identities?

There are various ways how identities can be represented:

- Most systems use so called accounts to represent the identity of the specific user.
- Some systems have several types of identities, for example a user entry that represents

the basic identity and a set of related entries that are related to this basic identity and keep more specific information.

In the first case it is quite easy to connect to DirX Identity because an account on the target system side represents exactly one account on the connected system side.

In the second case you have to map an account on the target system side to an account structure on the connected system side. This requires setting up multiple channels in the Java-based workflows to handle this situation correctly.

1.3.1.2. How Many Accounts Exist and What is the Typical Change Frequency?

These two numbers help to decide for the appropriate provisioning method:

- If there are only a few hundred accounts or a larger number but the change frequency is low, you could think about using manual provisioning (for more information see the chapter "Manual Provisioning of Offline Systems" in the *DirX Identity Use Case Document Service Management*).
- If there are thousands of accounts and a medium or high change frequency it makes sense to think about automatic provisioning via a connector.

The automatic provisioning method requires much more effort because you need to implement a connector and a set of provisioning workflows.

1.3.1.3. Can Accounts be Disabled?

This aspect is mainly relevant for the connectivity configuration. It influences the handling of the **dxrTSState** attribute within the connectivity workflows.

1.3.2. How does the system model access control?

In this section you have to evaluate the access control model in various dimensions.

1.3.2.1. Access control model

There are several methods how access control can be implemented:

- Some systems implement access rights to resources as attributes at the account entries. This method is called DAC (discretionary access control). Main disadvantage is that general changes in the access control model requires individual adaption of the access rights for a set of accounts.
- To avoid this problem, many systems offer a group based approach. In these systems groups can be defined that have specific access rights to resources. Assigning an account to one or more of these groups inherits the corresponding access rights to the accounts. Changing the access rights of such a group changes the access rights of the assigned accounts automatically.
- A third category of systems allows modeling a hierarchical model of groups. In this case the higher level groups are called roles (this method is called RBAC = role based access control). They aggregate some lower level roles or groups. Assigning one or more of these roles to a user inherits all lower level access rights to this user.

Dependent on the found model it is not straight forward to find an appropriate model in DirX Identity. We recommend the following methods:

- If there are only accounts with access control attributes, create for each access control attribute a corresponding group in the target system. Assigning the group manages the corresponding attributes at the account via DirX Identity's obligation mechanism.
- If there are accounts and a simple group model you can use the natural DirX Identity model with accounts and group (a one to one relationship).
- If the system uses groups and roles you should decide for one of the levels. Your decision should be driven by the granularity you want to control and make visible in DirX Identity. In any case you will model either the groups or the roles as groups in the target system.

Note that in case that the system has only accounts, the account management of such systems in DirX Identity can be modeled via the assignment and unassignment of a default dummy group which will be not synchronized to the connected system at all. An example of such an approach may be found in the Imprivata OneSign target system definition and its corresponding connectivity workflows.

1.3.2.2. Does the System Allow For Nested Groups?

Check whether a group in the system to connect to can contain another group as a member (so called nested groups)? DirX Identity supports nested groups in a limited manner. Try to get rid of the nested groups if possible, that means restructure the system accordingly.

1.3.3. Structural Conditions

The way how objects are organized and structured in the target system can influence the complexity of your provisioning approach.

1.3.3.1. Are Groups and Accounts Stored in a Hierarchical or Flat structure?

There are systems that keep all accounts and groups in a flat structure, others use object hierarchies to order and structure the objects.

A hierarchical structure is common for LDAP-based systems and also for Active Directory.

Databases keep the accounts and groups in separate tables which is a flat structure.

1.3.3.2. Are Memberships Stored at Accounts or at Groups?

These types can exist:

- In most cases the memberships are stored at the group side. A big disadvantage of this method is that all these systems have either an upper limit (for example RACF allows only 5000 memberships per group) or performance decreases drastically for high numbers of memberships. Unfortunately the maximum number of memberships can be equal to the number of identities if all users shall be member of a specific group. Examples for such systems are LDAP and Active Directory.

- Much better is the approach to store the memberships at the account side because even if you have a large number of groups in your system, no account will have more than some thousand memberships. Due to that fact high member numbers do not occur. An example for such a system is SAP UM R3.
- Some systems use both methods together. For example the UNIX-PAM target system type stores memberships at the group object but the primary group must be directly stored at the account object.
- Special types of systems are the ones that are based on relational databases. They store the membership in a special table which requires an advanced connectivity configuration for Java-based workflows.

In a DirX Identity target system you can configure both methods. Switching the flag triggers an automatic reconfiguration of the target system that can last a long time if a lot of accounts and groups is to change.

Additionally the provisioning workflows can handle cross memberships, that means the target system keeps the memberships on the account side and the connected system keeps it on the group side (the group side versus account side scenario is possible but does not make any sense because in this case the account side / account side scenario should be used).

We recommend strongly using account side memberships on the target system side whenever possible.

1.3.4. Which provisioning interfaces are available?

There are many methods possible:

- An optimal solution is if the system provides an SPML compatible web service interface. In this case the standard SPML connector of DirX Identity can be used.
- Many systems have their own proprietary provisioning API or an API that can be used for provisioning. In this case you have to implement a custom connector that provisions via this API. Alternatively you can use the manual provisioning method (for more information see the chapter "Manual Provisioning of Offline Systems" in the *DirX Identity Use Case Document Service Management*).
- If a system provides only a file-based interface, you can implement a custom connector that implements this interface or use the manual provisioning method (for more information see the chapter "Manual Provisioning of Offline Systems" in the *DirX Identity Use Case Document Service Management*).
- If a system does not provide any suitable interface you have two options:
  Use the template "Generic" or other available target system and derive a variant of this connectivity package or use the manual provisioning method (for more information see the chapter "Manual Provisioning of Offline Systems" in the *DirX Identity Use Case Document Service Management*).

2. Defining a Custom Target System Instance

This chapter describes how to set up a new custom target system instance by customizing a generic built-in target system template.

2.1. About this Use Case

This use case allows you to create a new target system instance through customization of a generic built-in target system template. According to the properties and the API of the connected system you have to set up a target system instance together with the necessary connected directory and the provisioning workflows.

This method is applicable if you need only one instance of the new custom target system.

2.1.1. Documentation Hints

You can find additional information related to this use case in the following documents:

2.1.1.1. Tutorial

Getting Started → Setting Up a New Target System – describes the steps to set up a target system instance of a supported type.

2.1.1.2. User Interfaces Guide

Using DirX Identity Manager → Using Wizards → How the Target System Wizard Works – describes the functionality of the target system wizard.

2.1.1.3. Application Development Guide

Configuring Custom Scenarios – provides overall information about configuration of new scenarios for provisioning and connectivity configuration including the creation of new target system instances of a supported type.

2.1.1.4. Customization Guide

- **Customizing Wizards → Customizing the Provisioning Target System Wizard** – contains general information about customization of target system wizard.
- **Customizing Target System** – provides a detailed description of customization possibilities of the target system instances of an existing type. It is the most important part of the DirX Identity documentation for the use cases described in this guide.

2.1.1.5. Provisioning Administration Guide

Use the context sensitive help when customizing the created target system instance. Alternatively you can access these parts of the documentation directly through the online help.

Context-Sensitive Help → Target Systems View - general hints how to manage and set up specific features for the target system instances.

2.1.2. Specific Hints and Guidelines

Use a built-in generic type target system to create a single instance of a custom target system.

Regard the following issues when you configure it:

- Try to model only the really necessary features of the specific target system.
- Use standard features for DirX Identity target systems management and try to reuse existing settings as much as possible.
- The generic target system type is built on two levels of object descriptions. The first level is defined by the object descriptors contained in the target system instance in **cn=Object Descriptions,*target system name*,cn=TargetSystems,cn=*domain***.
  The object descriptors of the first level include and reference the object descriptors stored in the default target system configuration in **cn=Object Descriptions,cn=Default,cn=TargetSystems,cn=Configuration,cn=*domain***.
  It is common for built-in target system types that the target system specific object descriptions are built on three levels hierarchy. It is also possible to create target system specific object descriptions that do not include any other files. We recommend using two levels of object descriptions as suggested by the generic target system type here.

2.2. Setup and Configuration

To set up a new instance of a custom target system perform these steps:

- Create a target system instance of the generic type
- Customize the newly created target system instance

2.2.1. Create a Target System Instance of the Generic Type

Login into the **Provisioning** configuration and then:

- Select the **Target Systems** view.
- Select the target system container or cluster under which you want to create the new instance.
- Select **New → Target System** from the context menu to start the target system wizard.
- In the **Target System Selection** step select the **Generic** type. You can check the option **Account and groups in common subtree** if desired.
- Fill the next target system wizard step **Target System General** as desired.
- Proceed to the step called **Target System Advanced**. Note that the generic target system type allows you to set and redefine the **Type** of the target system which is stored in the **dxrType** attribute. The value of the **dxrType** attribute is important when configuring the connectivity workflows. Typically the **dxrType** attribute value matches

the value of the corresponding connected directory type. See existing templates and documentation for more details.

- Go on to the **Associated Connected Directory** step. Since the target system template for generic type has no default connected directory associated, you can freely pick one of the existing connected directories that is suitable for your purposes. This step allows you to copy all related connectivity configuration objects. You can customize them later on.
- Proceed with the next steps as described in the documentation until you reach the **Provisioning Workflows** step. Select all workflows that you want to copy to your instance. You can customize them later on.
- Go to the end step and then **Finish** the wizard.

Your new target system instance is now created together with the related objects in the connectivity configuration (a connected directory and the selected provisioning workflows).

2.2.2. Customizing the Newly Created Target System Instance

In the previous step you created a fully independent target system instance that now needs to be customized. Study the customization guide carefully before you proceed with the following tasks.

2.2.2.1. Check and Set Up the Target System Tabs

Ensure that the target system tab layout shows the required attributes. If not, adapt the object description accordingly.

2.2.2.2. Check and Set Up the Target System Configuration

Typically a target system instance configuration contains the following information:

JavaScripts – a container for JavaScript implementations needed for this target system instance. Typically these JavaScripts are referenced from other object descriptions of this target system instance.

Object Descriptions – a container for object descriptions related to target system specific objects. Typically it contains files with the target system instance definition (**TS.xml**), the account definition (**TSAccount.xml**) and the group definition (**Group.xml**). It may also contain definitions of additional target system specific types of objects. This container must exist and contain at least the object descriptions for the target system instance (**TS.xml**) and for the group objects (**Group.xml**). The object descriptions for accounts are required only if the membership in the target system groups is stored at the accounts and not at user objects.

Obligations – contains obligation definitions if these shall exist. An obligation allows the execution of special actions triggered by group assignment or unassignment. For examples see the configuration of the ADS or UNIX-PAM target system type definitions.

Property Page Descriptions – may contain target system specific definition of property page descriptions written in Beans Markup Language (BML).

Note: this feature is deprecated and should only be used when adapting a legacy instance of a target system.

Proposal Lists – optional container for target system specific proposal lists.

Reports – may contain target system instance specific reports. Adapt it freely to your needs. Do not forget to create the corresponding access policies, otherwise your reports are not visible in Web Center.

2.3. Alternative or Extended Configurations

There are currently no extended or alternative configurations available for the described use case.

3. Defining a Custom Target System Template

This chapter describes how to set up a custom target system template that can be used later on to create multiple instances of this target system type.

3.1. About this Use Case

This use case is a natural extension of the previous one. After definition of a new target system template you can create a set of target system instances.

We explain three methods for target system template creation:

- Use an already existing instance of a target system (the main use case). The advantage of this method is that you can model and especially test a custom target system instance thoroughly before you use it as a new template.
- Customize one of the built-in target system templates (an alternate configuration proposal).
- Copy one of the existing custom target system templates (an alternate configuration proposal).

Typically you will use this case when modeling a more common target system type that is not supported by DirX Identity but shall exist in multiple instances within the company.

3.2. Compatibility

If you have already used the folder **cn=TargetSystems,cn=Customer Extensions,cn=Configuration,cn=*domain*** check if there are TargetSytem Type specific object descriptions. If yes you have to move these definitions to another folder (for example: **"cn=TargetSystemDefinitions,cn=Customer Extensions,cn=Configuration,cn=domain"**) and adjust all the imports referencing these object description. Otherwise you may run in problems like the one described in MZQLRZ: Incorrect Object descriptor nach DXI8.2B Update.

3.2.1. Documentation Hints

The documentation hints of the previous use case are also valid for this one. Read the appropriate chapter above.

3.2.2. Specific Hints and Guidelines

Use a built-in generic type target system to create a single instance of a custom target system.

Regard the following hints and guidelines when you configure you custom target system template:

- A good starting point is to use the target system instances that you created in the previous use case.
- Your new target system template should use two levels of target system object descriptions if possible. For more information see the "Specific Hints and Guidelines" section of the previous use case.
- Use a three level object description hierarchy for your target system template only when you want to customize an existing built-in target system template of other type than generic. In such a case the target system type (**dxrType**) should be unchanged. This is the preferred method when you plan common customization of a larger number of target system instances of a supported type. Assume for example that you need more specialized target system instances of an ADS or JDBC type. In such a case a new target system template of a known type may be prepared and then simply be reused.
- Target system templates stored in **cn=TargetSystems,cn=Customer Extensions,cn=Configuration,cn=*domain*** are based on common target system instances. They use the same object descriptions and folders. They should never be used directly while implementing a corresponding connectivity workflow. The group and account objects within a target system template do not trigger any real time events for provisioning workflows.
- The custom target system template should use a unique customer specific name. That name is used by the target system wizard.

3.3. Setup and Configuration

Set up of a custom target system template while reusing an existing target system instance requires you to perform the following steps:

- Prepare a target system instance
- Create your custom target system template
- Use your custom target system template

3.3.1. Prepare a Target System Instance

Perform these steps:

- Login into the **Provisioning** configuration.
- Select the **Target Systems** view
- Prepare a new or existing target system instance. Later on you can use this instance as basis for your custom target system template.

Note that you can use the target system instance that you generated based on the built-in generic target system type in the previous use case.

Any existing target system instance can be used as a target system template. Be sure that the selected one is correctly configured and then continue with the next step.

3.3.2. Create Your Custom Target System Template

Simply copy a well prepared target system instance to create your custom target system template.

All templates must be stored in the folder:

cn=TargetSystems,cn=Customer Extensions,cn=Configuration,cn=*domain*

Otherwise the target system wizard is not able to locate them.

3.3.2.1. Copying the Target System Instance

Perform these steps:

- Select an existing target system instance in the **Target Systems** view that you want to reuse.
- Select **Copy** (Ctrl+C) from the context menu.
- Click the Domain Configuration view.
- Navigate to **cn=TargetSystems,cn=Customer Extensions,cn=Configuration,cn=*domain***
- Select **Paste** (Ctrl+V) from the context menu.

The result is a complete copy of your target system instance.

3.3.2.2. Configure the Target System Template

All target system templates must contain the following folder:

Configuration – this folder contains all configuration objects that are relevant for this type of target system. Read the product documentation for details. During creation of a target system instance, the entire content of this folder is copied to the instance.

Dependent on the method to store accounts and groups you have to options:

- Accounts and groups are stored in separate folders that are named **Accounts** and **Groups**.
- Accounts and groups are stored in a common folder that is named **Accounts and Groups**.

While creating a new target system instance with the target system wizard you can select the method via the flag **Accounts and groups in common subtree**. Because the target system wizard requires for full functionality all three containers to exist, create the missing ones:

- Use **New → Account Container** from the context menu to create a missing **Accounts** folder (its type is **dxrTSAccountContainer**).
- Use **New → Groups Container** from the context menu to create a missing **Groups** folder (its type is **dxrTSGroupContainer**).

- Use **New → Account-Group Container** from the context menu to create a missing **Accounts and Groups** folder (its type is **dxrTSAccountGroupContainer**).

The target system wizard checks the presence of these containers and may disable the option **Accounts and groups in common subtree** according to the existing containers.

Note that the target system template should only contain objects that should be copied to each target system instance that is created from this template. Remove all unnecessary objects, for example all accounts and groups and leave only the objects that shall belong to the template. If you need additional objects in the template, create or copy them now.

The target system templates can be also renamed, use common context menu for this purpose.

3.3.3. Use Your Custom Target System Template

If the custom target system template is correctly configured, you can use it to create new target system instances. Perform these steps:

- Click the **Target Systems** view.
- Select a target system container or a cluster and start the target system wizard.
- In the Target System Selection step you should see all built-in templates as well as your new custom target system type.
- Select your new type and fill all necessary tabs and fields to complete the target system creation.

Your new target system instance is created.

3.4. Alternative or Extended Configurations

There are more possibilities to create custom target system templates. Create it with the target system wizard directly from built-in types or copy an already existing custom target system template.

We recommend using the method we described above, creating a template from an instance. The big advantage is that you can configure and test the instance completely, especially the connectivity workflows (this is subject of the next chapter).

3.4.1. Creating the Custom Target System Template Using a Built-in Template

Perform these steps:

- Login into the **Provisioning** configuration.
- Select the **Domain Configuration** view.
- Navigate to **cn=TargetSystems,cn=Customer Extensions,cn=Configuration,cn=*domain***.

- Select **New → Target System** from the context menu. A list containing all built-in templates is displayed. Select one of them and the system will create a target system template including the corresponding connectivity workflows if selected.
- Perform **Reload Object Descriptors**.
- Add the missing **Accounts**, **Groups** or **Accounts and Groups** folders.

Note that the list does not contain any custom target system templates. Only built-in types can be used this way.

Customize the created template to get your custom target system template.

3.4.2. Creating the Custom Target System Template Using a Custom Template

Perform these steps:

- Login into the **Provisioning** configuration.
- Select the **Domain Configuration** view.
- Navigate to **cn=TargetSystems,cn=Customer Extensions,cn=Configuration,cn=*domain***.
- Select one of the existing custom target system templates and choose **Copy Object** from the context menu. Do not forget to change the name before you click **OK**.
- Perform **Reload Object Descriptors**.

Customize the copied template to get your new custom target system template.

4. Creating Java-based Provisioning Workflows

This use case explains how to build the necessary Java-based provisioning workflows for a custom target system type or one of its instances. This is the most complex part of this use case document that requires a sound knowledge base about DirX Identity.

4.1. About this Use Case

If a target system template has a reference to a connected directory in the connectivity configuration, the target system wizard copies the necessary connectivity objects (mainly the connected directory and the associated workflows) as specified.

Correct set up of a custom target system template together with all workflows allows connecting a new connected system instance within minutes. Nevertheless this setup must be done thoroughly.

Note that you can use manual provisioning as an alternative solution if you do not or not yet need direct connectivity. In this case use the built-in **Service Management** target system type as starting point for your custom target system type.

4.1.1. Documentation Hints

You can find additional information related to this use case in the following documents:

4.1.1.1. Connectivity Administration Guide

- **Managing DirX Identity Connectivity → Managing Connected Directories** – describes overall how to manage connected directories in the connectivity configuration.
- **Managing DirX Identity Connectivity → Managing Provisioning Workflows** – describes overall how to manage provisioning workflows in the connectivity configuration.
- **Managing DirX Identity Connectivity → About Java-based Configuration Objects**

  – describes roughly the Java-based workflows configuration.

- **Managing Connected Directories** – describes how to manage connected directories in the connectivity configuration.
- **Managing Provisioning Workflows → Managing Java-based Provisioning Workflows** – describes how to configure and copy existing Java-based workflow objects.

4.1.1.2. Connectivity Reference

Identity Connectors – describes the functionality and configuration of the connectors which are used in the Java-based provisioning workflows.

4.1.1.3. Application Development Guide

- **Understanding the Default Application Workflow Technology → Understanding Java-based Workflows** – provides overall information about configuration and functionality of Java-based provisioning workflows.
- **Using the Default Connectivity Applications** – explains the rules when working with the default configuration objects.
- **Using the Target System (Provisioning) Workflows → Understanding the Java-based Target System Workflows** – describes functionality and implementation of delivered standard Java-based provisioning workflows. This is the most important documentation part for this use case.

4.1.1.4. User Interfaces Guide

Using DirX Identity Manager → Using the Connectivity Views → Using the Expert View – contains general information about usage of connectivity expert view which is necessary for this use case.

4.1.1.5. Use Case Description: Java Programming in DirX Identity

- **Provisioning Workflow Extensions** – describes how to implement custom mappings, user hook and filters used by Java-based provisioning workflows.
- **Implementing a Custom Connector** – explains how to implement a new connector that may be used in new Java-based provisioning workflows.

4.1.1.6. Integration Framework

- **Connector Integration Framework → Java Connector Integration Framework** – contains detailed information about Java framework that must be typically used when implementing own Java components for Java-based provisioning workflow.

4.1.2. Specific Hints and Guidelines

Creation of a completely new connectivity workflow requires experience with Java-based provisioning workflows:

- Try to identify the Java-based workflow that fits best as template for the new connectivity workflow you want to create.
- Consider all necessary aspects of the new connectivity workflow. Most important is that you have a connector ready that implements access to the connected system via the provided API.
- Be sure that you do not create objects with display names that exist already. Use display names that cannot collide with existing ones especially when working in the connectivity **Expert View** within the **Connectivity Configuration Data/Configuration**.
- This guide expects the existence of a customer specific instance of the Identity Store within **Connected Directories/*scenario_name*/Provisioning/Identity Store**. Create it by using the new target system wizard if not yet existing. Do not place any new objects in a directory that contains settings for default objects (contains **Default** in the path) unless

this is mentioned explicitly.

4.2. Setup and Configuration

For configuration of Java-based provisioning workflows follow these steps:

- Create a custom scenario for your custom target system templates
- Create a new connected directory
- Create the required Java-based provisioning workflows
- Refine the copied objects
- Test the new Java-based provisioning workflows

Note that there is an alternative way to create Java-based workflows. You can build all objects from scratch in the Expert view but this requires much more expert knowledge and we do not recommend this method.

4.2.1. Creating a Custom Scenario

We recommend creating a scenario with a fixed name to be able to recognize objects that belong to custom target system templates.

This step is only necessary if the CustomTemplates → Provisioning scenario does not exist, that means you have to perform it when you create the first custom target system template in your domain.

In this case perform these steps:

- Login into the **Connectivity** configuration.
- Select the **Global** view.
- Select the **Scenarios** node and choose **New → Folder** from the context menu.
- Enter **CustomTemplates** as name, set a description and click **OK**.
- Select the CustomTemplates node and choose **New → Scenario** from the context menu.
- Enter **Provisioning** as name, fill the other attributes and click **OK**.

A new and empty scenario exists. Additionally we need an Identity Store.

- Click into the blue empty area and select **New Connected Directory** from the context menu. A connected directory icon with the name '(new') is created.
- Right click the icon and perform **Configure** from the context menu.
- In the wizard's **Choose a Template** step select the **Identity Store** (from the folder Default/Identity Store).
- Step through the wizard steps but do not change anything. At the end click **Finish** to complete the wizard.

The icon is renamed to **Identity Store** and contains all the information of the original

instance. For correct operation, you should link the Identity Store to your already created scenario.

- In the scenario tree of the Global view click the **Provisioning** scenario and select **Properties** from the context menu.
- Click **Edit** and set the link in the **Identity Store** field to your newly created Identity Store (**Connected Directories → CustomTemplates → Provisioning → Identity Store → Identity Store**). Click **OK**.

This step is necessary that the target system wizard can recognize the Identity Store later on.

4.2.2. Creating a New Connected Directory

For each custom target system type we need a connected directory that represents the system to provision in the connectivity configuration.

Use an existing connected directory template that is similar to your new connected directory. Add this template to your scenario:

- Click into the blue empty area and select **New Connected Directory** from the context menu. A connected directory icon with the name '(new') is created.
- Right click the icon and perform **Configure** from the context menu.
- In the wizard's **Choose a Template** step select the template that you want to use as a starting point. Click Next.
- In the **Supply General Information** step set the name equal to the name of your already existing custom target system template. Set the other parameters in this step as required.
  Note: here you could also define a new connected directory type if you already created it. If not, you can perform this task later on or you can stay with the current type. Click **Next**.
- Fill the relevant parameters in the **Set Provisioning Parameters** step. Click **Next**.
- In the **Set Bind Profiles** step you can define or redefine bind profiles. Click **Next**.
- The next steps are dependent on your selected template. Typically you have to adapt the attribute configuration file according to the real attributes of the new connected directory. Fill all information according to the documentation or online help. At the end of the wizard click **Finish**.

The new connected directory exists. You can find it in the Expert view in the folder:

Connected Directories → CustomTemplates → Provisioning → Target Scheduled

You can perform additional changes here in the Expert view or you can re-run the wizard in the Global view.

Now we have to link this new connected directory to our already created custom target system template:

- Click the **Provisioning** view group and select **Domain Configuration**.
- Navigate to the custom target system template in **Customer Extensions → Target Systems**.
- Click it and perform **Edit**.
- In the **Relationships** area of the **General** tab set the link in the **Connected Directory** field to your newly created connected directory.

This step is necessary that you can later on configure and run provisioning workflows from the Provisioning configuration side.

4.2.3. Creating the New Java-based Provisioning Workflows

The next task is to copy the necessary workflows. First you should create a workflow line between the newly created connected directory and the Identity Store:

- Click into the blue empty area and select **New Workflow Line** from the context menu. Click both connected directory icons to create the line.

For each workflow you want to create perform these steps:

- Select the workflow line and perform **New** from the context menu. The workflow wizard opens and shows all workflows that fit between these two endpoints. Select the correct template and then click **Next**.
  Note: if you cannot find the required template, deselect the **Matching Endpoints** flag. Now you can see all workflows in your connectivity configuration regardless whether they fit the two endpoints or not. Be aware that you have to adapt such a copied template afterwards before it can work correctly.
- In the **General Workflow Parameters** step set the correct name for your new workflow and adapt all the other parameters as described in the online help.
- Step through the other tabs and set all parameters as required. At the end click **Finish**.

Each workflow is normally copied to the folder

Workflows → CustomTemplates → Provisioning → Target Realtime → *type*

You can perform additional changes here in the Expert view or you can re-run the workflow wizard in the Global view.

4.2.4. Refining the Objects

The copied connected directory as well as the copied workflows should be adapted further to reflect all requirements.

4.2.4.1. Refining the Connected Directory

You can redefine almost anything. Consider the following issues:

New Connected Directory Type?

DirX Identity comes with a lot of pre-defined connected directory types (verify this in the Expert view under **Configuration → Connected Directory Types**). A connected directory type can be used by many connected directory definitions (for example the connected directory type LDAP).

If you need a new one, create it under Configuration → Connected Directory Types. Use an existing connected directory type object, copy it and rename it. Next you can redefine its objects descriptions and wizards in the corresponding subfolders of the connected directory type.

Having the connected directory type created, update the references for Wizard and Directory Type at the new connected directory object which was previously created.

Tab Layout and the Displayed Attributes?

If you want to use other layout for the new connected directory, edit the object description for the related connected directory type. Alternatively you can use the Design Mode for simple adaptations of the layout. For more information see the documentation.

Attribute Configuration?

The attribute configuration of the connected directory reflects the schema of the system to connect. If you have an LDAP type connected directory, you can reload the schema to adapt the attribute configuration. Otherwise you have to enter and maintain all attributes by hand.

Note that the attribute configuration is only necessary for Tcl-based workflows.

4.2.4.2. Refining the Workflow(s)

The workflow wizard copied all workflow objects correctly and set the corresponding links (for example the links from the ports to the channels). Nevertheless you can adapt everything as required. Consider these issues:

Workflow → General?

Adapt the **Is applicable for**, the **Timeout** and the **Wizard Parameters** sections. See the online help for more information.

Join → General?

Adapt the **Error Handling** section. See the online help for more information.

Consider also the creation of a new resource family for running the activities of this new Java-based workflow. You can create a new one under **Configuration → Resource Families**. Create the folder CustomTemplates and then copy an existing resource family, rename it and set it for the activities of the new workflow object (typically error and join activity).

Always check that the configured resource families are deployed for a running Java-based server otherwise the activity cannot run.

Join → Controller?

You can also consider a change of the used controller. See the documentation for further information.

If you intend to use global user hooks for the new workflow, enter the appropriate class name of the implementing class to **Controller** tab of the join activity.

TS Port → Target System?

If you created a new connector, you have to set the Connector field correctly. You can define new connector definitions under **Configuration → Connector Types**. As a template use an existing one, copy it and define a component description for it (under the **Component Descriptions** subfolder). Set the correct class name for the class that implements the connectivity to the new connected directory.

4.2.4.3. Refining the Channels

The workflow wizard copied already all necessary channels from the selected workflow template on both sides (connected directory and Identity Store). All links were correctly adapted.

Adapt the Channel Structure

Typically four types of the channels should be present for a new connected directory:

- Channel for account object validation and synchronization
- Channel for group object validation and synchronization
- Channel for membership validation and synchronization
- Channel used for password synchronization (needed only by password sync workflows)

Each channel synchronizes one type of target system specific object if possible. These channels exist typically in pairs. One channel exists at the connected directory definition side. The other corresponding channel exists at the Identity Store side. The password channel exists only at the connected directory side. You can use more channels, if there are more types of objects to be synchronized.

If you need additional channels, copy or create them and configure them accordingly. You have so set up the channels below the connected directory → Channel folder as well as under the Identity Store → Channels folder. Be sure to set up the corresponding and correct folder structure. Link the channel pairs correctly.

If you intend to use password synchronization, set also correctly the **Password Primary Channel** for the password channel (typically the account channel). Set the **Member Channel** for the correct channel (the channel that handles objects that store the group membership).

Refining Mappings, Post Mappings and Channel User Hooks

Setup the mappings for the Java-based workflow channels as required. You can also define post mappings and specify the class name of the Java class implementing the channel user hook. Use Java class mapping instead of Java source wherever possible. It allows you to

debug Java mappings in a Java IDE. Define a new package name used for Java mappings which must be also defined at the channel object. Use the common naming conventions.

4.2.5. Testing the Workflows

We recommend performing the tests in two steps:

4.2.5.1. Check Correctness

Check the correctness of the new workflow. Enable **Design mode** to see the **Content resolved** tab at the Java-based provisioning workflow object.

View the content of this tab and try to find the characters **###** in the displayed XML content. This would mean that some references are incorrectly resolved. That does not always mean that an error occurred, but we recommend evaluating these inconsistencies and fix them if possible.

4.2.5.2. Live Testing

Now you are ready to perform a live test.

Create a new target system instance that is based on your custom target system template and that comes with the corresponding workflows. Test the workflow using the DirX Identity manager and the Java-based server.

You can debug the global and channel user hooks, the Java class mappings and the post mappings using a Java IDE. This requires the corresponding Java server running in the debug mode. This mode can be enabled by starting the Java Server via the command line using a batch runServer.bat (or .sh). Edit it and uncomment line beginning by **REM SET debug** (when working with Windows version otherwise use ported settings for UNIX systems). This will enable remote debugging on the pre-configured port 48174. Any Java IDE supporting remote debugging is now able to connect to this port and debug your Java classes used in the new Java-based provisioning workflows.

See the DirX Identity Use Case Document "Java Programming" for more information.

5. Transferring Custom Target System Templates

The full target system configuration with provisioning workflows consists of the provisioning domain specific configuration and the connectivity configuration. The DirX Identity transport workflows support automated configuration transfer from the provisioning or connectivity configuration. It is necessary to transfer connectivity and provisioning configuration in two separate steps.

You can create custom target system templates in a specific provisioning domain. If you intend to use these templates also in a different provisioning domain, it is necessary to configure the appropriate transport workflow. This allows you to correctly reconfigure the provisioning domain name. See documentation for more details.

The connectivity workflow configuration can be shared between more provisioning domains. This allows you to use simple collection mechanism for the configuration transfer. You can also use the transport workflows if desired.

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

Eviden is a registered trademark © Copyright 2026, Atos Group – All rights reserved.

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.