EVIDEN

Identity and Access Management

DirX Identity

Jaspersoft Reports

Version 8.10.14, Edition March 2026

Table of Contents

Copyright ii
Preface 1
DirX Identity Documentation Set 2
Notation Conventions 4
1. Overview 6
2. Templates and Input Files 7
  2.1. Report Templates 7
  2.2. Parameters 8
  2.3. Subreports 8
  2.4. Report Configuration 9
  2.5. Nationalization 10
  2.6. Template Fields and [Virtual] Attributes 11
    2.6.1. Utility Methods 12
    2.6.2. firstElementOrEmptyStringFrom 12
    2.6.3. firstElementOrNullFromList 13
    2.6.4. toStringOrNullFromObject 13
    2.6.5. printObjectList 13
    2.6.6. printObjectArray 13
    2.6.7. createLocale 13
    2.6.8. formattedTime 13
    2.6.9. formattedTimeFromList 13
    2.6.10. formatDateTime 14
    2.6.11. whyDetailsToString 14
    2.6.12. getFirstValueFromDn 14
    2.6.13. convertDn2Path 14
    2.6.14. parseCommonName 14
    2.6.15. matchPattern 14
3. Developing Reports 15
  3.1. Templates in the File System 15
  3.2. Report Configuration File 16
  3.3. Control File 16
  3.4. Running Reports 16
  3.5. Report Template for Testing 17
  3.6. Importing Report Artifacts to LDAP 17
    3.6.1. Report Entries and Artifacts with Import Tool 18
    3.6.2. Create Report Entry manually 18
    3.6.3. Styles 18
    3.6.4. Images 19
    3.6.5. Nationalization texts (i18n) 19

4. Upgrading from V8.9 20
5. Helpful Resources 21
Legal Remarks 23

Preface

This document describes how to develop and generate reports using JasperReports® library.

It consists of the following chapters:

- Chapter 1 provides an overview of the features.
- Chapter 2 describes the LDAP entries and files that make up a full report definition.
- Chapter 3 explains how to develop a report.
- Chapter 4 describes how to upgrade from a previous version.
- Chapter 5 provides some helpful links to Jaspersoft documentation.

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

DirX Identity uses TIBCO JasperReports library, “the world’s most popular open source reporting engine” [https://community.jaspersoft.com/project/jasperreports-library] for generating reports on DirX Identity domain entries. It’s the same library that DirX Audit uses for generating its reports. The library can produce pixel-perfect documents including charts of different types and export them in a variety of document formats including HTML, PDF, Excel, CSV, OpenOffice and Word.

For the report design, you can use TIBCO Jaspersoft® Studio. It presents a graphical user interface and supports you in the layout of visual components, using lots of chart types (most important: bar and pie charts), integrating images and maps, writing complex expressions and integrating subreports to display, for example, the list of roles a user has assigned.

The JasperReports library provides a set of built-in Java functions for evaluating object attributes and producing nice output, for example, for date formats. DirX Identity provides additional utility methods intended to handle situations specific to DirX Identity entries and attributes.

The product supports internationalization. You can display texts and labels in any language and print, for example, dates in language specific format.

The JasperReports library and similar reporting tools are mainly built and described for producing reports on data stored in relational SQL databases. DirX Identity provides a custom data source implementation that lets the Jasper reporting engine fill the reports from DirX Identity Service layer objects (SvcUser, etc.) and especially leverages the use of “virtual attributes”; for example, to retrieve the roles of a user by requesting the attribute “roles_assigned”.

You can produce a report from DirX Identity Web Center, DirX Identity Manager, a DirX Identity workflow and during report design with a command line tool. Unfortunately, it is not possible to run a report from within Jaspersoft Studio: we were not able to solve the class loader problems related to the storage LDAP URLs.

The report and subreport templates are expected to be stored in LDAP entries, the same entries and folders where the classic XSLT-based report definitions are stored. Also, the other artifacts like the referenced images, the styles, and the internationalization messages are held in LDAP. For designing and testing a new report, all the artifacts can be held in local files. When finished, command line tools help you to import them to LDAP.

Find pdf samples of generated reports in folder **Additions/SampleReports** of the product installation media.

2. Templates and Input Files

Artifacts that make up a complete report definition need to be stored in LDAP entries. For compatibility reasons and for supporting report development, all or a part of them can be stored in the file system. See chapter ‘Developing Reports’ for more details.

In the DirX Identity LDAP domain, the Jasper-based report templates are stored along with the XSLT-based reports. So, the default reports delivered out-of-the-box are in

*Domain***/Configuration/Reports/Default**

For better management and easier overview, the Jasper-based reports are stored in the subfolder **Jasper-based Reports**. Unlike the XSLT-based reports, there are no Jasper-based reports specific for target system types.

Artifacts that make up a complete report definition are:

- *Report template* or *report output format* - A report template prescribes the structure and layout of a report. The report template is in XML format and stored in an LDAP entry. In the report design phase, it can also be stored in a file. The file needs the suffix **.jrxml**.
- *Images* – A report template may refer to images, for example, a company logo. These entries are stored in LDAP in the subfolder *Domain***/Configuration/Reports/Default/_Images**.
- *Style templates* – The product provides a couple of style templates, for example, for blue, green or yellow. You select them at the beginning of a report template. These entries are stored in LDAP in the subfolder *Domain***/Configuration/Reports/Default/_StyleTemplate**.
- *Internationalization (i18n)* – The keys and their translations to the respective language are stored in LDAP entries in the subfolder *Domain**/Configuration/Reports/Default/Nationalization*. The section ‘**Error! Reference source not found.Error! Reference source not found.**’ gives more details on the LDAP entries and the format.
- *DirX Identity specific report configuration* – An XML document that defines the data to be filled into the report, the output type (PDF, XLS, etc), the i18n bundle name and the output file name. The configuration is stored in the same LDAP entry as the report template. When used in design phase, it can be stored in an XML file.

2.1. Report Templates

A report template contains all the information about the structure and the aspects of the document that will be generated when the data is provided. This information contains the position and content of various text or graphic elements, their appearance, the custom calculations, the data grouping and data manipulations that should be performed when the data are generated, and so on.

For details on a report template, see the Jaspersoft documentation, especially the chapter “Report Template Structure” and the subsequent ones in [4] (see “Helpful Resources” in this document).

A report template is stored in LDAP in a report entry of object class **dxrReport**. Identity Manager presents it in its *Format* tab and stores it in the **dxrFormat** LDAP attribute. This is the same kind of entry and attribute that holds the output format of XSLT-based reports.

Subreport templates are also stored in LDAP entries of object class **dxrReport**. They are distinguished from the main template by the value **jasperSubReport** in their LDAP attribute **dxrType**. In subreport entries, the configuration (attribute **dxrConfig**) is ignored and can be left empty. The subreport entries MUST be stored in the same folder as their main templates.

Please pay attention to the artifacts described in the next sections.

2.2. Parameters

Parameters are defined at the beginning of a report template. They can be output to the report document and are referred to by $P{*parameter_name*}.

Some parameters are set from the DirX Identity report generator at report production time, including SearchBase, SearchScope, SearchFilter, REPORT_LOCALE and all the properties listed in the <*consumer*> section of the report configuration.

Other important parameters refer to the document template: DIRX_STYLE_TEMPLATE_DIR (leave this unchanged) and DIRX_STYLE_TEMPLATE_FILE. Change the value of this file parameter if you want to exchange the template style; for example, from **DXT71_Styles_blue** to **DXT71_Styles_orange**.

2.3. Subreports

Subreports are typically used when printing a variable list of information in a main report. Examples are the roles or groups assigned to a user.

A subreport is referred from its containing main template by an XML element <**subreport**>. Its <**subreportExpression**> needs to contain the name of the subreport template. In DirX Identity, this name must be defined as a parameter and is filled by the DirX Identity report generator at report production time. The name MUST match exactly the RDN of the LDAP subreport entry.

For example, the subreport for a user’s roles:

The parameter is defined as

```
<parameter name="Users_with_all_properties_Roles"
class="net.sf.jasperreports.engine.JasperReport"/>
```

... and referred in <subreport> as:

```
<subreportExpression>
<![CDATA[$P{Users_with_all_properties_Roles}]]>
</subreportExpression>
```

The LDAP entry must be stored in the same folder as the main template and named as the parameter; in our example, it needs to be:

```
cn=Users_with_all_properties_Roles,cn=Users_with_all_properties,cn=Us
ers,…
```

Some parameters should be passed to the subreport: REPORT_LOCALE, DIRX_STYLE_TEMPLATE_DIR, DIRX_STYLE_TEMPLATE_FILE, REPORT_RESOURCE_BUNDLE.

2.4. Report Configuration

A report configuration is an XML document that defines

- The data to be filled into the report
- The output type (PDF, XLS, etc.)
- The i18n bundle name
- The output file name
- The style template
- The images folders

The configuration is stored in the same LDAP entry as the report template in the LDAP attribute dxrConfig. The configuration is nearly the same as the one used for XSLT-based reports.

A Jasper-based report is identified by the XML attribute type="jasper" in the top XML element <report>.

The <producer> section contains the search parameters that identify the set of entries that are to be filled into the report. In its sub-element <search>, these are the attributes base, filter, scope, attributes, sizelimit and order. They are mapped directly to the corresponding attributes of an LDAP search.

Note: Due to the way in which DirX Identity Storage components behave, the list of LDAP attributes need not necessarily be complete. But it supports performance if they are. If an attribute is requested in the report template and is not within the list of search attributes, a DirX Identity Storage object performs an extra LDAP read.

The output format is specified by an XML element of the output type name in uppercase. For PDF output, the element must be <PDF>, for Excel output it must be <XLS> or <XLSX>.

Its sub-element <FILE><consumer> contains sub-elements <property> for setting various parameters.

- output file name:

```
<property name="file" value="Users_with_all_properties.pdf"/>
```

- resource bundle name (see the section “Nationalization“)

```
<property name="i18nBundle" value="messages"/>.
```

- Folder and name of style template LDAP entry (or file)

```
<property name="DIRX_STYLE_TEMPLATE_DIR"
value="STYLE_TEMPLATE_DIR_LDAP_DEFAULT:"/>.
<property name="DIRX_STYLE_TEMPLATE_FILE"
value="DXT71_Styles_blue"/>.
```

  If you want to use custom styles, place them under folder “*cn=_StyleTemplate, cn=Reports,cn=Customer Extensions,cn=Configuration,cn=<domain name>*” and configure template directory

```
<property name="DIRX_STYLE_TEMPLATE_DIR"
value="DIRX_STYLE_TEMPLATE_DIR_CUSTOM:"/>.
```

- Identifiers for default and custom image containers. Don’t change them.

  If you want to use custom images, place them in folder“cn=_Images, cn=Reports,cn=Customer Extensions,cn=Configuration,cn=<domain name>”.

2.5. Nationalization

Reports can be localized by leveraging the Java internationalization support. Format and naming of the message properties must follow the Java conventions for internationalization (see https://www.oracle.com/technetwork/java/javase/tech/intl-139810.html). For example, the English text for the key ‘*sn*’ must be denoted as:

```
sn=Last name
```

The translated key can be referred to from the report template by the placeholder $R; for example, **$R{sn}**.

The report engine first searches the appropriate message texts in LDAP. If it doesn't find them, it searches in the file system.

In LDAP it searches below the folder "*cn=Configuration,cn=<domain name>*". The default message texts are delivered below folder

"*cn=Nationalization,cn=Jasper Based Reports,cn=Default,cn=Reports,cn=Configuration,cn=<domain name>*".

Custom messages should be located below folder

"*cn=Reports,cn=Customer Extensions,cn=Configuration,cn=<domain name>*".

The attribute *cn* (displayed in Identity Manager as *Name*) of a nationalization entry must match the configuration property **i18nBundle**. Messages for a different language, but same i18nBundle have to be placed into different folders, for example "*cn=messages,cn=de*, …". The language is stored in the attribute *dxrLanguage*, displayed in Identity Manager in field *Language*. If the report engine doesn't find an entry with the requested language, it takes English as default.

The message texts are stored as one long string in attribute *dxrMessage*. This allows you to easily copy all the messages from a simple text file to the LDAP entry.

If not found in LDAP, the report engine searches the message properties files in the installation's reports folder (*install_path**/reports*) in the subfolder **i18n**. For a configured i18Bundle 'messages' the default file name is **messages.properties**. Language-specific files suffix the bundle name with a locale-specific identifier. For example, the file name is **messages_en.properties** for English and **messages_de.properties** for German.

2.6. Template Fields and [Virtual] Attributes

In the report template, field names identify the properties of the data source entries from which the information is taken. For example, the following XML element specifies the surname property with a Java value type String:

```
<field name="sn" class="java.lang.String"/>
```

Each attribute to be printed in the output document must be declared by such a <field> element. When reading this property from the data source, the Jasper report engine expects exactly the configured return type and stops processing the report if it is different.

A field is referred to from the template by the **$F** placeholder, for example **$F{sn}**.

The DirX Identity service layer classes provide a number of "virtual attributes". They are not real LDAP attributes that can be found in the LDAP schema, but are supported by individual classes in their method *getProperty(<name of attribute>)*. They are used for various purposes, especially to implicitly read associated LDAP entries from LDAP. As an example, the SvcUser class retrieves the user-role assignments of its user when it is called with *getProperty(roles.assigned)*.

These virtual attributes can be leveraged in the report templates and are very helpful when including associated entries of an entry selected by the search in the report configuration. The associated document *virtualAttributes.pdf* lists DirX Identity service layer classes and the virtual attributes of attribute prefixes they support.

For example: the class SvcUser passes *getProperty* requests for attribute names starting with “roles.” to a controller class SvcUserRoleAssCtrl, which returns for attributes such as “roles.assigned” the role assignments of the user.

Note these very important points:

- In the report templates, replace the “.” in virtual attribute names with “_”. For example, use “roles_assigned” rather than “roles.assigned”. The reason is that the Jasper report engine interprets this attribute so that it first tries to get the property “roles”, expects an object and then gets the property “assigned” from it. This action fails because the Service layer class does not return anything for the property “roles”. Behind the scenes, the DirX Identity custom data source transforms back ‘_’ to ‘.’ and passes the correct attribute name “roles.assigned” to the DirX Identity class.
- Virtual attributes are always returned as a list even if they contain only one value. See the section “Utility Methods” for information on how you can easily extract the first value from this list.

2.6.1. Utility Methods

To simplify template definition, you can use utility methods delivered with the product. They leverage the Jaspersoft feature to call methods of Java classes from within a report.

First, you should define a parameter as an alias for the full class name of the utility class. This allows you to use the short alias rather the long class name later in the report.

```
<parameter name="Utils"
class="siemens.dxm.storage.report.jasper.Utils"/>
```

Then in a text field you can access the method entering the alias followed by the method name, separated by ‘.’ like so:

```
$P{Utils}.firstElementOrEmptyStringFrom($F{manager.cn})
```

This example calls the method *firstElementOrEmptyStringFrom* passing the value of the virtual attribute **manager.cn**.

Here is the list of utility methods for the class **siemens.dxm.storage.report.jasper.Utils**:

2.6.2. firstElementOrEmptyStringFrom

The method *firstElementOrEmptyStringFrom* expects as the only parameter a list of objects. It returns either an empty string if the list is null or empty or the first element in the list.

2.6.3. firstElementOrNullFromList

The method *firstElementOrNullFromList* expects as the only parameter a list of objects. It returns either null if the list is null or empty or the first element in the list.

2.6.4. toStringOrNullFromObject

The method *toStringOrNullFromObject* expects as parameter an object and returns null if the object is null or returns the stringified value of the object: object.toString.

2.6.5. printObjectList

This method expects a list of objects and returns their stringified, comma-separated values.

2.6.6. printObjectArray

This method expects an array of objects and returns their stringified, comma-separated values.

2.6.7. createLocale

This method expects 3 string parameters for language, country and variant and returns the corresponding Java **Locale**. Default is Locale.UK.

2.6.8. formattedTime

The method *formattedTime* expects the following parameters:

1. A GeneralizedTime object.
2. The format string as used in Java class *SimpleDateFormat*. It gives the output pattern telling which part of the date and time should be included.
3. The locale affecting the output format.

The method returns the formatted date time string for the given time or null if the object is missing.

2.6.9. formattedTimeFromList

The method *formattedTimeFromList* expects the following parameters:

1. A list of GeneralizedTime objects.
2. The format string as used in Java class **java.text.SimpleDateFormat**. It gives the output pattern telling which part of the date and time should be included.
3. The locale affecting the output format.

The method returns the formatted date time string for the first element of the input list or null if the list is empty.

2.6.10. formatDateTime

This method expects the following parameters:

1. The format string as used in Java class **java.text.SimpleDateFormat**. It gives the output pattern telling which part of the date and time should be included.
2. The locale affecting the output format.
3. A **java.sq.Timestamp** to be formatted.

The method returns the formatted date time string for the given time or null if the timestamp is missing.

2.6.11. whyDetailsToString

This method expects a Java **Set** of strings and returns their values as comma-separated list.

2.6.12. getFirstValueFromDn

This method expects a DN string as an input parameter and returns the first RDN (relative DN); that is, the entry's name or null if the DN is missing or in the wrong format.

2.6.13. convertDn2Path

This method expects a DN string and returns a string with the path notation as known from Unix file paths. For example: a DN "cn=users,cn=My-Company" would be returned as "My-Company/users".

2.6.14. parseCommonName

This method expects a DN string and returns the leading *cn* RDN.

2.6.15. matchPattern

This method expects the following string parameters:

1. The string for which the pattern is to be applied.
2. A regular expression.

It returns the result of applying **java.util.regex.Matcher** on a pattern created from the regular expression on the input string.

3. Developing Reports

In the file system, all the report data are stored in the following folder:

install_path/**reports**

The files are distributed to subfolders according to their type. See the following list on which subfolder contains which types of files. Please follow these best practices when designing your own reports.

The report templates must be stored in XML format in files with the suffix "**.jrxml**". They can be edited with any XML editor. We recommend using TIBCO Jaspersoft Studio. It's a free, graphical, Eclipse-based report designer that allows you to design complex JasperReports definitions easily and quickly. See the references [1] and [2] in the chapter "Helpful Resources" for its home page and user guide.

For production, the templates their configuration (search base, filter, etc.), and the other artifacts are taken from LDAP. For development, it is much easier to work with templates in files. Jaspersoft Studio primarily works with files and cannot use templates in LDAP.

DirX Identity supports this use case:

- All the report artifacts can be stored in files.
- A batch file allows for generating the report from the command-line passing options in a control file.

When you have finished report development with Jaspersoft Studio, you have to import the necessary artifacts to LDAP. A couple of CLI tools is supporting you for that.

See the next sections for more details.

3.1. Templates in the File System

You can store the templates in file folders. Out-of-the-box, the installation already contains a report template in the subfolder **reports** of the installation. The subfolder reflects the structure of the reports in LDAP:

reports/Users/Users with all properties

Subreports must be placed in the same folder with exactly the names used in the main template. For example, if the main template refers the subreport with:

Users_with_all_properties_Roles

the subreport template must be stored in the file:

Users_with_all_properties_Roles.jrxml

3.2. Report Configuration File

A configuration file must be supplied for configuring the entries to be filled into the report and other parameters (as described in the section “Report Configuration”).

The file name suffix should be “**.xml**”, the standard name in the samples is **jspContent.xml**. We recommend storing it in the same folder as the templates. The content is exactly the same as in the attribute **dxrConfig** (displayed in the Content tab) of the LDAP report entry.

3.3. Control File

You need a control file for generating a report in the file system. It is the same format as used in the report generation workflows. For each report delivered in the reports folder, there is a corresponding file **control.ini** in the same subfolder as the report templates.

Besides general information like the operation (generateReport), address and bind credentials to access the DirX Identity LDAP domain and logging levels, it especially sets values for the following keys:

- **reportFolder** – The folder that contains the report control, configuration and definition files.
- **report.input.file** - The file with the main report template. The path to the file can be absolute or relative to the folder where the report is located.
- **reportCtxFile** – The absolute or relative path to the configuration file; in the samples: **jspContent.xml**.
- **report.filename** – The file where the filled report is written. For example, **Users_with_all_properties.pdf**.
- **report.outputFormat** – The output format for the report: PDF, XLSX, etc.
- **viewReport** – Whether (**true**) the report engine opens a pop-up window with the report output or (**false**) writes it into the output file.
- **report.sizelimit** – The maximum number of entries to be written into the report. This parameter helps to limit the LDAP search.
- **report.name** – The DN of the report configuration entry in LDAP. This parameter is mandatory but its value is ignored when the parameter **report.input.file** is set (not commented).
- **reportsFolder** – If set, the folder that contains the sub-folders with styles, images and i18n. Per default it is *install_path**/reports*.

If you want the job to take the report from LDAP, comment out the line for **reportCtxFile** and make sure to have the correct DN in **report.name**. As a sample, take the file **controlLdap.ini**.

3.4. Running Reports

To generate the report, use the file **genReport.bat**. It is in the folder **reports**.

Change the last line in the file to select the appropriate control file.

Normally, run the batch file from the **reports** folder. Per default, it expects that the message properties files and the styles are in the subfolder **reports** of the installation. The samples also use relative file paths and are below the reports folder.

If you want to design and test in another folder, you can either configure absolute file paths for the properties reportsPath and / or reportPath or even store the batch file and your configuration in a different folder. In the batch file you see that the Java class path is set relative to the installation folder. This assumes DirX Identity is installed on your test system.

3.5. Report Template for Testing

The product provides a very simple report template and a corresponding subreport. Its function is to test and experiment with domain entries and their attributes, especially testing the values of the virtual attributes.

The templates, the configuration and the control file are provided in folder **tests** beneath **reports**.

3.6. Importing Report Artifacts to LDAP

If you are ready with the report design in files, you should transfer them to LDAP so that they can be used in the normal way, for example, from Identity Manager and Web Center.

The scripts for importing styles, images and message texts are in folder

install_path/**reports/importTools**.

Each script uses its own configuration file, which have most of the configuration properties in common:

- User, password, host, and port for accessing the Identity domain.
- Default: if true, the imported entries are put to the report default folder *Domain*/**Configuration/Reports/Default**. This flag is normally commented and then the imported entries go into the **Customer Extensions** folder**.**

The import tools create an LDAP entry per file or replace a corresponding LDAP entry, if it already exists.

There are two ways to import report definitions to LDAP:

- Use the tool *runImportReports* and import a whole folder with report definitions, styles, images, and nationalization texts.
- Create the LDAP entries with report and sub-report definitions manually and import the other artifacts by running the respective import tool.

3.6.1. Report Entries and Artifacts with Import Tool

The fastest way to import all artifacts that make up a report is to use the **importReports** tool.

The script

runImportReports.[**bat** | **sh**]

imports report definitions, image icons, styles, and nationalization texts in a selected folder to LDAP into folder

*Domain***/Configuration/Customer Extensions/Reports**

Configure the import folder in the properties file *install_path***/reports/reportImport.properties**. Set the property **diskFolder** to the desired folder path.

Before you can select the report in the User Interface, you must define the appropriate object type(s) in the field **Types** of the main report entry: User, Role, etc.

3.6.2. Create Report Entry manually

In LDAP, create a folder for the new report and create a new report entry. Copy and paste the report template from the **jrxml** file to the report entry's Format tab. Copy the configuration from the configuration file (**jspContent.xml**) to the Content tab.

Do not forget to set the object type(s) in the field **Types** of this report entry: User, Role, etc.

For each subreport, create another report entry in the same folder as the main report. Make sure that the name of the report (LDAP RDN) matches the parameter name used in the report template.

You need to set the type of the subreport entry to subreport. At the time of this writing, this task must be performed in DirX Identity Manager's Data View. From the context menu of the entry, select **Go to DataView** and change attribute **dxrType** to **jasperSubReport**.

3.6.3. Styles

The script

runImportStyles.[**bat** | **sh**]

imports style definitions to LDAP, per default into folder

*Domain***/Configuration/Customer Extensions/Reports/_StyleTemplate**

If your style files are not in folder *install_path**/reports/_StyleTemplate*: in the properties file **styleImport.properties** uncomment the line with diskFolder and set the folder or file name.

3.6.4. Images

The script

runImportImages.[**bat** | **sh**]

imports image icons to LDAP, per default into folder

Domain/Configuration/**Customer Extensions/Reports/_Images**

If your image files are not in folder *install_path***/reports/img**: in the properties file **imgImport.properties** uncomment the line with diskFolder and set the folder or file name.

3.6.5. Nationalization texts (i18n)

The script

runImportI18n.[**bat** | **sh**]

imports nationalized message texts to LDAP, per default into folder

Domain/Configuration/**Customer Extensions/Reports/Nationalization**

If your message files are not in folder *install_path***/reports/i18n**: in the properties file **i18nImport.properties** uncomment the line with diskFolder and set the folder or file name.

4. Upgrading from V8.9

Starting with V8.10, images, styles and nationalized texts (i18n) are stored in LDAP. Therefore, the configuration for the reports must be extended. In the XML element <consumer> the following sub-elements must be added:

```
<property name="DIRX_IMAGE_DIR" value="IMAGE_DIR_LDAP_DEFAULT:"/>
<property name="DIRX_IMAGE_DIR_CUSTOM" value=
"IMAGE_DIR_LDAP_CUSTOM:"/>
<property name="DIRX_STYLE_TEMPLATE_DIR"
value="STYLE_TEMPLATE_DIR_LDAP_DEFAULT:"/>
<property name="DIRX_STYLE_TEMPLATE_DIR_CUSTOM"
value="STYLE_TEMPLATE_DIR_LDAP_CUSTOM:"/>
```

You can do that manually and edit the Content tab of each report or use the CLI tool to do that automatically.

Edit login credentials and LDAP address in the configuration file **migrateConfig.properties** and run the script

runMigrateConfigurations.[**bat** | **sh**]

This will update the reports in the customer extensions folder. If you customized reports in the default folder, you need to set the property **default** to **true**.

5. Helpful Resources

List of Jaspersoft documents: https://community.jaspersoft.com/documentation

[1] Jaspersoft Studio Home page: https://community.jaspersoft.com/project/jaspersoft-studio

[2] Jaspersoft Studio User Guide: https://community.jaspersoft.com/documentation/tibco-jaspersoft-studio-user-guide/v720/getting-started-jaspersoft-studio-0

[3] JasperReports library home page: https://community.jaspersoft.com/project/jasperreports-library

[4] Jaspersoft Reports Library Ultimate Guide: https://community.jaspersoft.com/documentation/jasperreports-library-ultimate-guide;.

[5] Report samples: https://community.jaspersoft.com/wiki/jasperreports-library-samples

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