EVIDEN

Identity and Access Management

DirX Audit

Administration Guide

Version 9.0, Edition September 2025

Table of Contents

Preface

This manual provides information how to administer DirX Audit. It consists of the following chapters:

- Chapter 1 describes how to manage DirX Audit Manager Classic.
- Chapter 2 describes how to manage DirX Audit Manager and REST API.
- Chapter 3 describes how to manage DirX Audit Message Broker.
- Chapter 4 describes how to manage DirX Audit Server.
- Chapter 5 describes how to manage DirX Audit databases.
- Chapter 6 describes how to configure DirX Audit collectors.
- Chapter 7 provides information about DirX Audit Server error handling.
- Chapter 8 provides information about DirX Audit in a multiple tenant environment.
- Chapter 9 provides information about the Audit message logical schema.
- Chapter 10 provides information about several implementations of authorization policy enforcement points (PEP) that DirX Audit supports.
- Chapter 11 describes how to configure the generator for populating OLAP fact tables (KPI generator / Fact Population Tool).
- Chapter 12 provides information on configuring and running synchronization jobs for importing history entries.
- Chapter 13 describes how to configure the scheduled job that updates History Database entries.
- Chapter 14 provides information about the History Database schemas.
- Chapter 15 provides useful hints on tuning database performance in your environment.
- Chapter 16 provides information about the fact tables that are the foundation for DirX Audit Manager Classic's Dashboard components.
- Chapter 17 describes how to set log levels for DirX Audit applications.
- Chapter 18 describes how to manage DirX Audit cryptographic material.
- Chapter 19 describes how to monitor DirX Audit with Java Management Extensions (JMX).
- Chapter 20 describes how to monitor DirX Audit with reports.

DirX Audit Documentation Set

DirX Audit provides a powerful set of documentation that helps you configure your audit server and its applications.

The DirX Audit document set consists of the following manuals:

- *DirX Audit Introduction*. Use this book to obtain a description of DirX Audit architecture and components.
- *DirX Audit Tutorial*. Use this book to get familiar quickly with your DirX Audit installation.
- *DirX Audit Administration Guide*. Use this book to understand the basic tasks of DirX Audit connectivity administration.
- *DirX Audit Manager Classic Guide*. Use this book to obtain a description of the DirX Audit Manager Classic user interface provided with DirX Audit.
- *DirX Audit Manager Guide*. Use this book to obtain a description of the DirX Audit Manager user interface provided with DirX Audit.
- *DirX Audit Command Line Interface Guide*. Use this book to obtain a description of the command line interface provided with DirX Audit.
- *DirX Audit Customization Guide*. Use this book to customize your DirX Audit environment.
- *DirX Audit History Synchronization Guide*. Use this book to obtain information about the DirX Audit History synchronization jobs.
- *DirX Audit Best Practices*. Use this book to obtain guidelines and tips for avoiding common mistakes and improving the experience of a DirX Audit installation.
- *DirX Audit Installation Guide*. Use this book to install DirX Audit.
- *DirX Audit Migration Guide*. Use this book to migrate DirX Audit from previous versions.
- *DirX Audit Release Notes*. Use this book to understand the features and limitations of the current release.
- *DirX Audit History of Changes*. Use this book to understand the features of previous releases.

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.

install_path
The exact name of the root of the directory where DirX Audit programs and files are installed. The default installation directory is *userID_home_directory*/**DirX Audit** on UNIX systems and **C:\Program Files\DirX\Audit** 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*.

install_media
The exact path where the DirX Audit installation media is located.

1. Managing DirX Audit Manager Classic

This chapter provides information about how to manage DirX Audit Manager Classic. It consists of the following sections:

- Running the DirX Audit Manager Classic service
- Configuring LDAP authentication
- Configuring a secure HTTP connection
- Configuring a secure LDAP connection
- Configuring Windows authentication using the Kerberos login module
- Configuring Single Sign-on (SSO) Web authentication using SPNEGO / Kerberos

1.1. Running the DirX Audit Manager Classic Service

The DirX Audit Manager Classic application is based on the Apache Tomcat technology.

To start and stop DirX Audit Manager Classic, use the service provided by Apache Tomcat. This service must be configured externally.

You can check whether the application is running in your Internet browser. Type the following URL:

http://*hostname***:***port***/AuditManager/?tenant=***tenantID*
https://*hostname***:***port***/AuditManager/?tenant=***tenantID*

where *hostname* specifies the server address, *port* the server port number and *tenantID* the identifier of a tenant generated during the tenant configuration process. The default port for HTTP is **8080** and **8443** for HTTPS.

Examples:

`http://localhost:8080/AuditManager/?tenant=8cd425f7-adcd-4c89-8e24-90160a51b428`

`https://localhost:8443/AuditManager/?tenant=8cd425f7-adcd-4c89-8e24-90160a51b428`

1.2. Configuring LDAP Authentication

DirX Audit Manager Classic can authenticate users on an external directory server over the LDAP protocol. The authentication is fully configurable, and you can choose from a broad range of directory services. (See the sections “Audit Manager Classic Authentication” in the Core Configuration and the “Authentication Configuration” in the Tenant Configuration in the chapter “Configuring DirX Audit” in the *DirX Audit Installation Guide* for details.)

The roles (Auditors, AuditAdmins and RestrictedAuditors) are evaluated in DirX Audit Manager Classic when a user accesses the Dashboard, Audit analysis, Reports and History views. (See the *DirX Audit Manager Classic Guide* for details.)

Here is a sample configuration for a DirX Identity domain:

Search base for users
`cn=Users,cn=My-Company`

Search filter for users
`(&(objectClass=dxrUser)(%s))`

User target
`cn`

Search base for groups
`cn=Groups,cn=DirXmetaRole,cn=TargetSystems,cn=My-Company`

Search filter for groups
`(&(objectclass=dxrTargetSystemGroup)(uniqueMember=%d))`

Group target
`cn`

List of auditor groups
`cn=Auditors,cn=Groups,cn=DirXmetaRole,cn=TargetSystems,cn=My-Company`

List of audit administrator groups
`cn=AuditAdmins,cn=Groups,cn=DirXmetaRole,cn=TargetSystems,cn=My-Company`

List of restricted auditor groups
`cn=RestrictedAuditors,cn=Groups,cn=DirXmetaRole,cn=TargetSystems,cn=My-Company`

User identification attribute
`dxrUID`

User mail attribute
`mail`

Here is a sample configuration for an Active Directory domain:

Search base for users
`CN=Users,DC=dxt,DC=my-company,DC=com`

Search filter for users
`(&(objectClass=user)(%s))`

User target
`sAMAccountName`

Search base for groups
`OU=Groups,DC=dxt,DC=my-company,DC=com`

Search filter for groups
`(&(objectclass=group)(member=%d))`

Group target
`cn`

List of auditor groups
`CN=Auditors,OU=Groups,DC=dxt,DC=my-company,DC=com`

List of audit administrator groups
`CN=AuditAdmins,OU=Groups,DC=dxt,DC=my-company,DC=com`

List of restricted auditor groups
`CN=RestrictedAuditors,OU=Groups,DC=dxt,DC=my-company,DC=com`

User identification attribute
`employeeNumber`

User mail attribute
`mail`

1.3. Configuring a Secure HTTP Connection

We strongly recommend that you run the DirX Audit Manager Classic application via the HTTPS protocol. See the Tomcat documentation for details. For example, http://tomcat.apache.org/tomcat-11.0-doc/ssl-howto.html.

1.4. Configuring a Secure LDAP Connection

DirX Audit Manager Classic can authenticate users on an external directory server over the LDAP protocol. We strongly recommend that you secure this connection. For LDAP to work over an SSL connection, the SSL certificate of the LDAP server must be added to the trusted certificate store (trust store) on the manager container.
See the sections "Authentication Configuration", "Server LDAP Collector for DirX Identity Format" and "Configuring LDAPS" in the chapter "Configuring DirX Audit" in the *DirX Audit Installation Guide* for details.

1.5. Windows Authentication Using the Kerberos Login Module

DirX Audit Manager Classic can authenticate users with their windows user name and password.
To enable this feature, check **Windows authentication** in the Configuration Wizard.

Users are authenticated according the Kerberos configuration as configured in the core configuration step in the "Audit Manager Classic Authentication" dialog. (See the **Kerberos file** option.) If the Kerberos configuration file is missing the Key Distribution Center (KDC) is taken from the DNS configuration. The user is authenticated against this KDC.

For authentication, the standard DirX Audit Manager Classic login page is used:

https://*hostname***:***port***/AuditManager/?tenant=***tenantID*

Enter your user name in **Name** in the format *Kerberos REALM**username*, for example `MY-Company.COM\user01`. If you omit *Kerberos REALM* then the default realm is used for authentication.

Enter your domain password in **Password**.

1.5.1. Kerberos Configuration File

The Kerberos configuration file contains the Kerberos configuration information including the locations of KDCs and admin servers for the Kerberos realms of interest. On Windows systems, its default location is **C:\Windows\krb5.ini**. On Unix systems, its default location is **/etc/krb5.conf**.

Here is a sample configuration file:

```
[libdefaults]
default_realm = MY-COMPANY.COM
default_tkt_enctypes = aes256-cts aes128-cts rc4-hmac des3-cbc-sha1
des-cbc-md5 des-cbc-crc
default_tgs_enctypes = aes256-cts aes128-cts rc4-hmac des3-cbc-sha1
des-cbc-md5 des-cbc-crc
permitted_enctypes = aes256-cts aes128-cts rc4-hmac des3-cbc-sha1
des-cbc-md5 des-cbc-crc
default_checksum = rsa-md5
kdc_timesync = 0
kdc_default_options = 0x40000010
clockskew = 300
check_delegate = 0
ccache_type = 3
kdc_timeout = 60000

[realms]
MY-COMPANY.COM = {
    kdc = DXA-SERVER-01.my-company.com:88
}

[domain_realm]
my-company.com= MY-COMPANY.COM
.my-company.com = MY-COMPANY.COM
```

1.6. Configuring SSO Web Authentication Using SPNEGO / Kerberos

DirX Audit Manager Classic can authenticate users through the SPNEGO mechanism using Kerberos authentication protocol. Users can authenticate without entering their usernames/passwords.

The Kerberos authentication is fully configurable. To enable this feature, check **Windows SSO** in the Tenant Configuration Wizard for a defined tenant.

There are two required configuration items:

- **Keytab location** – the location of the **keytab** file that contains the service principal-specific information.
- **Service Principal Name** (SPN) – defines a unique service principal name registered in the domain.

For authentication, the DirX Audit Manager Classic URL is used:

https://*hostname***:***port***/AuditManager/?tenant=***tenantID*

If authentication was not successful, you are redirected to the standard DirX Audit Manager Classic login page.

The next sections describe how to create these items and configure your Internet browser for SSO authentication.

1.6.1. Generating the Keytab File

Use the **ktpass** tool to generate the **keytab** file that contains the service principal-specific name and the keys required for the Kerberos ticket validation. If the validation is successful, the user is authenticated (but not yet authorized).

To generate the **keytab** file, run the following command in the command line as an administrator:

ktpass -princ HTTP/*fully_qualified_Audit_manager_classic_hostname***@***AD_name*
-mapuser *username* **-out** *filename* **keytab -pass** *user's_password* **-crypto All**

For example:
`ktpass -princ HTTP/tomcat.my-company.com@MY-COMPANY.COM -mapuser tomcat -out tomcat.keytab -pass dxt -crypto All`

(the example assumes that **tomcat** is a user existing in Active Directory).

The **keytab** file is then created and can be found in the current working directory of the command line.

1.6.2. Defining the Service Principal Name

The Service Principal Name (SPN) is a unique service principal name registered in a domain. It is mapped to a user registered in Active Directory. The creation of the user in Active Directory is not described here.

The SPN format for a Web application is as follows:

HTTP/*fully_qualified_hostname***@***AD_name*

For example:
`HTTP/tomcat.my-company.com@MY-COMPANY.COM`

The Kerberos realm name is equivalent to the name of the Active Directory.

1.6.3. Configuring the Internet Browser for Windows SSO Authentication

When using Windows authentication, the Internet browser must be configured as described here.

1.6.3.1. Configuring Microsoft Edge and Google Chrome

To configure the Microsoft Edge and Google Chrome browsers:

1. Press the Windows key + **R** to open the **Run** command box. Type **inetcpl.cpl** and then press **Enter**.
2. Navigate to the **Security** tab.
3. In the **Local intranet** window, enter the web address of the host name where DirX Audit Manager Classic is installed into **Add this web site to the zone**.
4. In the **Internet Options** window, click the **Advanced** tab and scroll to **Security settings**. Ensure that the **Enable Integrated Windows Authentication** (requires restart) box is selected.
5. Click **OK**.
6. Restart the Internet Explorer browser to activate this configuration.

1.6.3.2. Configuring Mozilla Firefox

To configure the Firefox browser:

1. Start Firefox.
2. In the **address** field, enter **about:config**.
3. In the search filter, enter **network.n**.
4. Double-click on **network.negotiate-authn.trusted-uris**. This preference lists the URIs that are permitted to engage in Simple and Protected GSS-API Negotiation Mechanism (SPNEGO) authentication with the browser. Enter the Audit Manager Classic installation URL (**http://***fully_qualified_name_of_Audit_Manager_classic_host* or **https://***fully_qualified_name_of_Audit_Manager_classic_host* when using a secure connection).

2. Managing DirX Audit Manager and REST API

This chapter provides information about how to manage DirX Audit Manager. It consists of the following sections:

- Running the DirX Audit Manager service
- Configuring LDAP authentication
- Configuring a secure HTTP connection
- Configuring a secure LDAP connection
- Configuring session timeouts
- Using Swagger user interface

2.1. Running the DirX Audit Manager Service

The DirX Audit Manager application is based on the Apache Tomcat technology.

To start and stop DirX Audit Manager, use the service provided by Apache Tomcat. This service must be configured externally.

You can check whether the application is running in your Internet browser. Type the following URL:

http://*hostname***:***port***/audit-manager-***tenantID*
https://*hostname***:***port***/audit-manager-***tenantID*

where *hostname* specifies the server address, *port* the server port number and *tenantID* the identifier of a tenant generated during the tenant configuration process. The default port for HTTP is **8080** and **8443** for HTTPS.

Examples:

`http://localhost:8080/audit-manager-8cd425f7-adcd-4c89-8e24-90160a51b428`

`https://localhost:8443/audit-manager-8cd425f7-adcd-4c89-8e24-90160a51b428`

2.2. Configuring LDAP Authentication

DirX Audit Manager can authenticate users on an external directory server over the LDAP protocol. The authentication is fully configurable, and you can choose from a broad range of directory services. (See the sections "REST Service Authentication Configuration" in the Tenant Configuration in the chapter "Configuring DirX Audit" in the *DirX Audit Installation Guide* for details.)

The roles (Auditors, AuditAdmins and RestrictedAuditors) are evaluated in DirX Audit Manager when a user accesses the Audit analysis, Reports and History views. (See the *DirX Audit Manager Guide* for details.)

Here is a sample configuration for a DirX Identity domain:

Search base for users
`cn=Users,cn=My-Company`

Search filter for users
`(&(objectClass=dxrUser)(%s))`

User target
`cn`

Search base for groups
`cn=Groups,cn=DirXmetaRole,cn=TargetSystems,cn=My-Company`

Search filter for groups
`(&(objectclass=dxrTargetSystemGroup)(uniqueMember=%d))`

Group target
`cn`

List of auditor groups
`cn=Auditors,cn=Groups,cn=DirXmetaRole,cn=TargetSystems,cn=My-Company`

List of audit administrator groups
`cn=AuditAdmins,cn=Groups,cn=DirXmetaRole,cn=TargetSystems,cn=My-Company`

List of restricted auditor groups
`cn=RestrictedAuditors,cn=Groups,cn=DirXmetaRole,cn=TargetSystems,cn=My-Company`

User identification attribute
`dxrUID`

User mail attribute
`mail`

Here is a sample configuration for an Active Directory domain:

Search base for users
`CN=Users,DC=dxt,DC=my-company,DC=com`

Search filter for users
`(&(objectClass=user)(%s))`

User target
`sAMAccountName`

Search base for groups
`OU=Groups,DC=dxt,DC=my-company,DC=com`

Search filter for groups
`(&(objectclass=group)(member=%d))`

Group target
`cn`

List of auditor groups
`CN=Auditors,OU=Groups,DC=dxt,DC=my-company,DC=com`

List of audit administrator groups
`CN=AuditAdmins,OU=Groups,DC=dxt,DC=my-company,DC=com`

List of restricted auditor groups
`CN=RestrictedAuditors,OU=Groups,DC=dxt,DC=my-company,DC=com`

User identification attribute
`employeeNumber`

User mail attribute
`mail`

2.3. Configuring OIDC Authentication

DirX Audit Manager can authenticate users via Authorization server supporting OpenID Connect (OIDC).

Sample configuration for DirX Access authorization server

OpenID Configuration
`https://my-server.my-company.example:10115/oauth-provider/.well-known/openid-configuration`

Issuer
`https://my-server.my-company.example:10115/oauth-provider`

JWK Set URI
`https://my-server.my-company.example:10115/oauth-provider/.well-known/jwks.json`

Authorization endpoint
`https://my-server.my-company.example:10115/oauth-provider/authz`

Token endpoint
`https://my-server.my-company.example:10115/oauth-provider/token`

Client ID
`MyCompanyOpenIdClientPUBLIC`

Redirect URI
`https://localhost:30444/web-manager-1c523d49-bcf9-40b7-973d-93894ab0bb24/dirx-dxt-app-manager`

List of auditor roles
`Intranet Employee,Intranet Manager Accounting`

List of restricted auditor roles
`Extranet Employee`

List of audit administrator roles
`Administrator`

Roles
`roles`

ID
`sub`

Name
`name`

Email
`email`

Other
`audience:aud`
`expiration:exp`

Mapping of JWT payload claims

Here is a sample of JSON Web Token (JWT) payload issued by DirX Access authorization server:

```
{
    "sub": "mL0Yf7wkl5e8r9dxf6we5c46sd2r3C5JM.",
    "ou": "Finances",
    "roles": [
        "Intranet Manager Accounting",
        "User"
    ],
    "givenName": "John",
    "iss": "https://my-server.my-company.example:10115/oauth-
provider",
    "o": "My-Company",
    "aud": "MyCompanyOpenIdClientPUBLIC",
    "surname": "Cook",
```

```
    "auth_time": 1755853458,
    "loginName": "John Cook",
    "name": "John Cook",
    "exp": 1755857059,
    "iat": 1755853459,
    "email": "john_cook@atos.net"
}
```

DirX Audit Server expects claims to be presented in this format. Names of claims should correspond with names mapping configured in DirX Audit Configurator in step REST Service Authentication Configuration in section **Claims mapping**. (See the sections "REST Service Authentication Configuration" in the Tenant Configuration in the chapter "Configuring DirX Audit" in the *DirX Audit Installation Guide* for details.)

All claims should contain string value or array of string values. Currently, nested claims are not supported in DirX Audit Server.

Mandatory claims expected by DirX Audit Server:

- `subject` - unique identifier of the user (**sub** in sample JWT payload)
- `roles` - array of roles assigned to the user (**roles** in sample JWT payload)

Claims that are not mandatory but are used in DirX Audit Manager UI:

- `name` - full name of the user (**name** in sample JWT payload)
- `email` - email of the user (**email** in sample JWT payload)

Following claims are necessary for proper functionality of DirX Audit Reports export. Names of these claims are not configurable:

- `organization` - organization of the user (**o** in sample JWT payload)
- `organizational unit` - organizational unit of the user (**ou** in sample JWT payload)

Corresponding configuration of claims mapping in DirX Audit Configurator for the sample JWT payload above would look like following

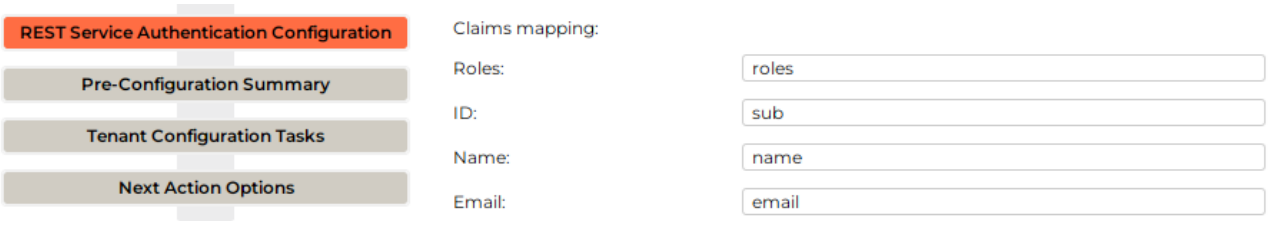

Figure 1. Configuration of Claims mapping

For example, if the claim containing roles should be called **dirxauditroles** in the JWT payload, the configuration of claims mapping would look like following

```
"dirxauditroles": [
    "Intranet Manager Accounting",
    "User"
]
```

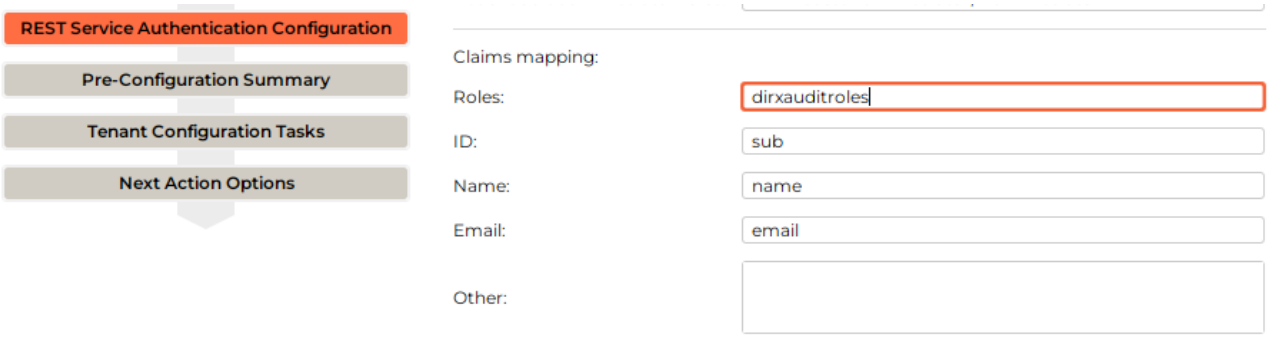

Figure 2. Custom configuration of claim name holding roles

Configuration of scopes in DirX Audit Manager

By default, 3 scopes are supported in OIDC authentication - openid, email and profile. These scopes can be modified in DirX Audit Manager configuration file **app-config.json** in section *authentication - oAuth - scope* (*install_path***web\audit-manager-**<*tenantId*>**\plugins\dirx-dxt-app-manager\assets\config\app-config.json**). Custom scopes can be added here manually. Modification and adding of scopes to configuration of Swagger UI is currently not supported (See the sections "Using Swagger UI" in section **Login** in this guid).

Redirect in DirX Audit Manager and Swagger UI

For OIDC authentication flow to work properly in DirX Audit Manager, make sure that redirect URI in format configured by attribute **Redirect URI** (See the sections "REST Service Authentication Configuration" in the Tenant Configuration in the chapter "Configuring DirX Audit" in the *DirX Audit Installation Guide* for details.) is listed in Authorization server in Client metadata (defined by **Client ID** configured in DirX Audit Configurator).

For OIDC authentication flow to work properly in Swagger UI, make sure that redirect URI in format **https://***hostname***:***port***/Tenants/***tenantID***/api/audit/swagger-ui/oauth2-redirect.html** is configured in Authorization server Client metadata (defined by **Client ID** configured in DirX Audit Configurator).

2.4. Configuring a Secure HTTP Connection

We strongly recommend that you run the DirX Audit Manager application via the HTTPS protocol. See the Tomcat documentation for details.
For example, http://tomcat.apache.org/tomcat-11.0-doc/ssl-howto.html.

2.5. Configuring a Secure LDAP Connection

DirX Audit Manager can authenticate users on an external directory server over the LDAP

protocol. We strongly recommend that you secure this connection. For LDAP to work over an SSL connection, the SSL certificate of the LDAP server must be added to the trusted certificate store (trust store) on the manager container.
See the sections "REST Service Authentication Configuration", "Server LDAP Collector for DirX Identity Format" and "Configuring LDAPS" in the chapter "Configuring DirX Audit" in the *DirX Audit Installation Guide* for details.

2.6. Configuring Session Timeouts

DirX Audit Manager uses the session timeout settings to manage user sessions effectively. We recommend configuring these settings to balance security and user convenience. Default timeout of session is set to one hour and default maximum age of session cookie is set to 10 hours. For more details, see the section "REST Service Authentication Configuration" in the *DirX Audit Installation Guide*.

2.7. Using Swagger UI

Swagger is tool for accessing and documentation of REST API endpoints. It shows all available endpoints and provides a way to use them outside of Audit Manager web application.

To access REST API use URL: **https://***hostname***:***port***/Tenants/***tenantID***/api/audit/swagger-ui/index.html**

Figure 3. DirX Audit RESTful API

Under the title **DirX Audit RESTful API** is link to API documentation in JSON format. To access API docs directly use URL: **https://***hostname***:***port***/Tenants/***tenantID***/api/audit/v3/api-docs**

2.7.1. Disabling Swagger UI

To disable Swagger UI, set the property `enable_swagger` to `false` in the section `[server_container.rest]` of corresponding tenant `configuration.cfg` file. This will prevent Swagger UI from being accessible.

This property is set to `true` by default, which means that Swagger UI is enabled.

The access to the API documentation in JSON format is not affected by this property and remains accessible.

2.7.2. Configuring Execution URL

In section **Servers**, URL path should be shown with current tenant ID. This path will be used to send requests when using endpoints in Swagger UI. In case of need, the dropbox can be expanded and different URL can be configured with setting of protocol, host, port number and tenant ID.

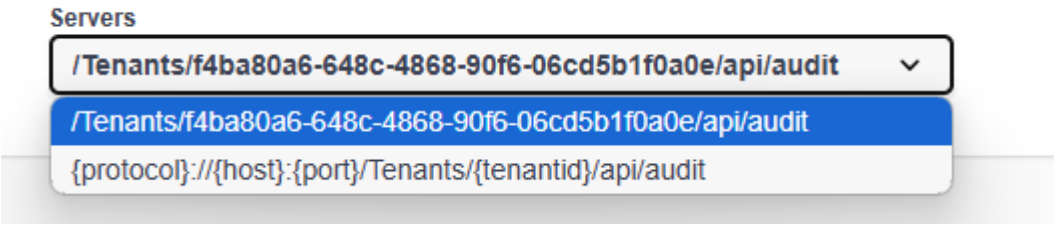

Figure 4. Server configuration

2.7.3. Login

To the right from Servers section is **Authorize** button. Use this button to authorize user.

Figure 5. Authorize button

- LDAP authorization
  - LDAP username and password must be filled in

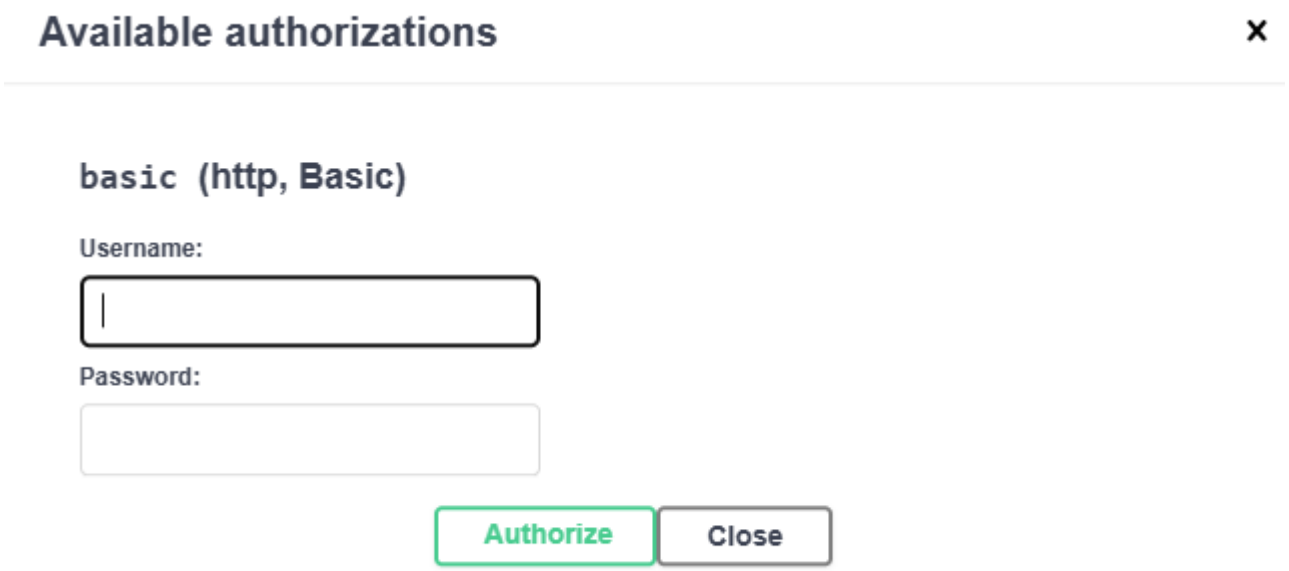

Figure 6. Basic authentication

- OIDC authorization
  - token authentication – requires filling in the token value

Figure 7. Token authentication

  - oAuth2 authentication – redirects to authorization server for user to be authorized and redirects back to Swagger UI
    For this method to work make sure values are prefilled for *Authorization URL*, *Token URL*, *Flow* and *Client ID*.

Authorization URL, *Token URL* and *Client ID* are configured in tenant Configurator in step "REST Service Authentication Configuration",
Client secret can stay empty as the method uses authorization code with PKCE.
Scopes *openid*, *email* and *profile* should be all selected for proper user management in Audit Manager.

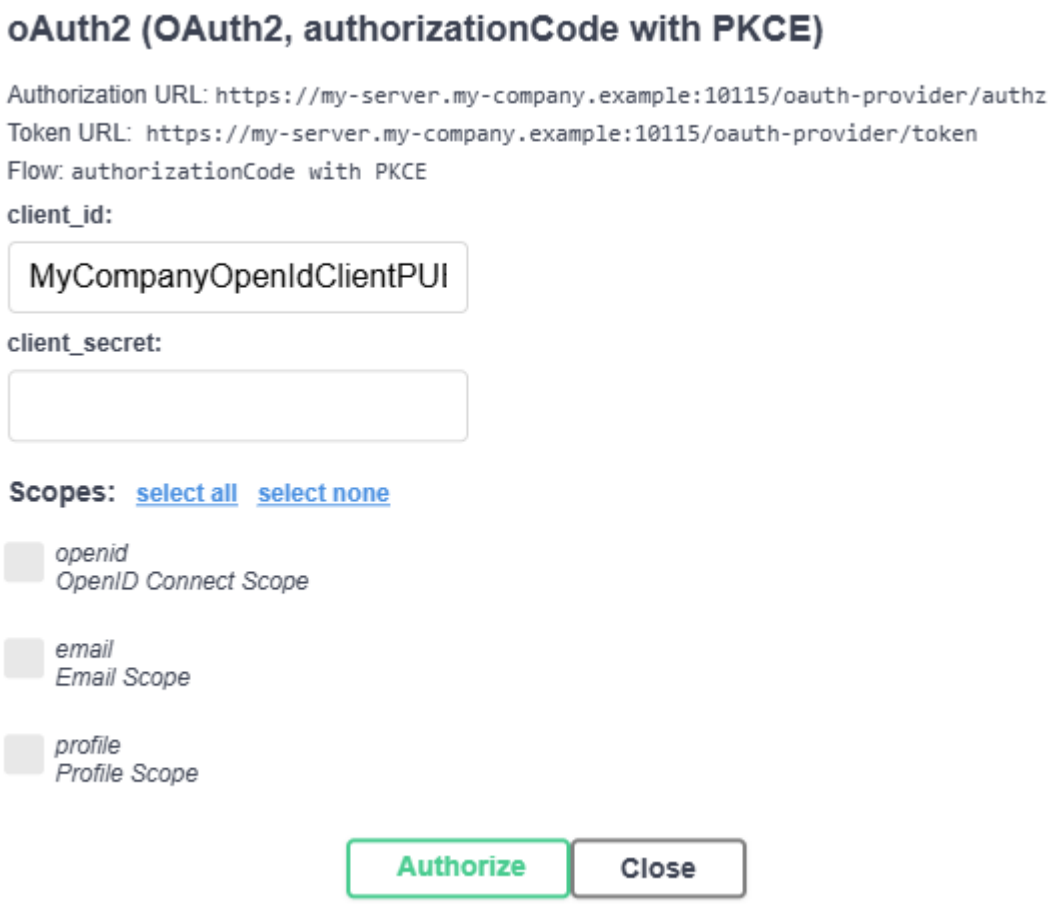

Figure 8. oAuth2 authentication

Once authenticated, all REST API endpoints shown in Swagger UI should be usable to the extend of authorized user permissions.

2.7.4. Logout

- LDAP authentication
  - execute **/logout** endpoint to ensure that security context and authentication object of user are cleaned on server side - without this the previous user will be used even after using Logout button provided by Swagger UI
  - use **Logout button** provided by Swagger UI to clean the credentials

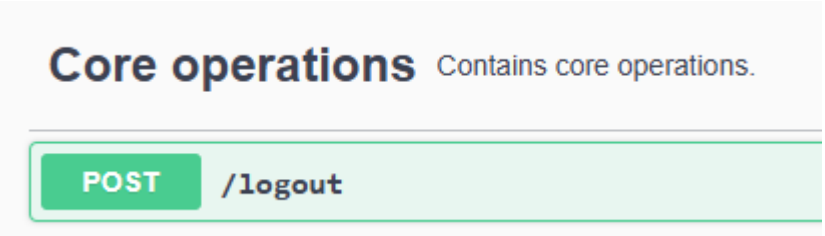

Figure 9. Endpoint to perform logout on server side

- OIDC authentication
  - use **Logout button** provided by Swagger UI

Figure 10. Swagger UI Logout button

3. Managing DirX Audit Message Broker

This chapter provides information on how to manage the DirX Audit Message Broker. It consists of the following sections:

- Running the DirX Audit Message Broker service
- Securing the Message Broker
- Monitoring the Message Broker

3.1. Running the Message Broker Service

The DirX Audit Message Broker is based on the Apache ActiveMQ technology. To start and stop DirX Audit Message Broker, use the DirX Audit Message Broker service (dirx-audit-messagebroker on UNIX). (See the section "Message Broker System Service" in "Configuring DirX Audit" in the *DirX Audit Installation Guide* for details.)

If the DirX Audit Message Broker is restarted, the DirX Audit Server reconnects automatically within a minute. However, other connected servers may also need a restart to renew their connection to the DirX Audit Message Broker; for example, DirX Access Servers.

3.2. Securing the Message Broker

Securing the DirX Audit Message Broker includes securing its account passwords and using a secure HTTP connection. The next sections provide more detail about these tasks.

3.2.1. Configuring the Accounts

DirX Audit Message Broker is deployed with pre-configured authentication and authorization rules. Please set the account passwords for the following accounts in the Message Broker Core and Tenant Configuration:

Administrative credentials, in the Core Configuration

- system (system)
- webconsole admin (admin)
- webconsole user (user)

User credentials, in the Tenant Configuration

- dxt-*tenantID*-reader
- dxt-*tenantID*-writer

Propagate these changes as follows:

- dxt-*tenantID*-reader: Modify the configuration of the JMS collectors in the DirX Audit Server.
- dxt-*tenantID*-writer: Modify the configuration of the DirX Identity JMS-Audit Handler Plug-in and DirX Access JMS-Audit Handler Plug-in.

See the sections "Message Broker Administration" and "Common JMS Collector Credentials" in chapter "Configuring DirX Audit" in the *DirX Audit Installation Guide* for details.

3.2.2. Using a Secure HTTP Connection

The DirX Audit Message Broker runs the HTTPS protocol by default. You can consult the Apache ActiveMQ documentation for additional information, http://activemq.apache.org/how-do-i-use-ssl.html.
See also the section "Preparing the Message Broker" in chapter "Preparing Truststores and Keystores for SSL Configuration" in the "Installation Configurations" chapter of the *DirX Audit Installation Guide*.

3.3. Monitoring the Message Broker

You can monitor the traffic of the DirX Audit Message Broker. Open your Internet browser and type the URL for the Apache ActiveMQ Console:

https://*host***:***port***/admin/queues.jsp**

where *host* specifies the server address and *port* the server port number; for example:
`https://localhost:30662/admin/queues.jsp`

DirX Audit uses the predefined port number **30662**.

The Apache ActiveMQ Console displays a table containing all registered message queues.

Use the following URL to display all active connections:

https://*host***:***port***/admin/connections.jsp**

The Apache ActiveMQ Console displays a table containing all registered connections.

To access the Apache ActiveMQ Console, you need the credentials that you have set up in the Core Configuration Wizard. You can use both the webconsole **admin** account and the **user** account to access the console.

4. Managing DirX Audit Server

This chapter provides information on how to manage the DirX Audit Server. It consists of the following sections:

- About the DirX Audit Server Components
- Running the DirX Audit Server Service

4.1. About the DirX Audit Server Components

DirX Audit Server (also called the Server) hosts the following component types:

- Collectors to collect audit messages from audit sources.
- Transformers to transform audit messages from their original data format to audit messages in the DirX Audit format.
- A digest producer to generate audit events by calculating the digest fields for operation, type and detail.
- A tag producer to generate message and event tags for each audit message and its audit events.
- A Persistence Service Unit to write audit messages including audit events and tags into the relational database.
- An error-handling unit to manage errors in collected audit messages.
- A scheduler to run scheduled reports, context records calculation, history DB update, and fact population on a regular basis.
- A scheduler to run purge jobs for history entries or audit messages.
- A scheduler to run history synchronization and other jobs on a regular basis.
- A RESTful API services to provide access for DirX Audit Manager to contents of DirX Audit Database and configuration.

A separate instance of DirX Audit Server is used for every tenant starting with version 7.2.

The collectors read audit messages from their specific audit source and pass them to the appropriate transformer. The transformed messages are then passed to the persistence unit, which adds them to the DirX Audit Database and creates the audit events and tags for each message. Messages that cannot be transformed or stored are passed to the error-handling unit, which stores the failed messages into files. These components are deployed in the form of Apache Camel components and routes.

For more details on these components and their configuration, see the chapters in this guide on database connectivity, collectors, digest and tag producers, fact population and error handling.

4.2. Running the DirX Audit Server Service

The DirX Audit Server is based on the Spring Boot technology. A separate instance of DirX Audit Server is used for every tenant starting with version 7.2.

To start and stop DirX Audit Server for a specific tenant, use the **DirX Audit Server** *tenant_name* (**dirx-audit-server-***tenantID* on UNIX) service.

Also note that the server will immediately implement any changes in deployed packages and routes in the *install_path***/server_container/tenants/***tenantID***/deploy** folder (for example, changes of the routes scheduler or deployed collectors) while changes in the server configuration (taken from *install_path***/conf/tenants/***tenantID***/configuration.cfg**) will be implemented only after DirX Audit Server for a specific tenant is restarted.

The server also uses the JDBC driver file located in the *install_path***/lib** folder. Microsoft JDBC Driver for SQL Server is provided with the standard DirX Audit installation. If you want to connect an Oracle Database, the Oracle JDBC driver should be placed in the *install_path***/lib** folder for the DirX Audit Server, the DirX Audit Manager Classic and tools. See the section "Oracle Database JDBC Driver Installation" in the "Installation Configurations" in the *DirX Audit Installation Guide* for instructions on how to install it.

See the *DirX Audit Release Notes* for more information about supported drivers.

Also see the sections "Server Scheduled Jobs" and "Application Container Configuration" in "Configuring DirX Audit" in the *DirX Audit Installation Guide* for further configuration details.

4.3. REST API

REST API is published at base URI: **https://***hostname***:***port***/Tenants/***tenantID***/api/audit/**

Rest API provides access for DirX Audit Manager to contents of installed databases, security context and installation information.

- Core REST API

  Provides information about logged-in user and method to remove security context of the user when logging out.

- Configuration REST API

  Provides information about DirX Audit configuration, such as licensed components.

- I18n REST API

  Provides access to internationalization (i18n) resources, such as language bundles and assets.

- Data Database REST API

  Provides access to Data Database content - Audit Messages and Audit Events.

- Configuration Database REST API

  Provides access to Configuration Database content - Audit Analysis Views, History Views, Reports and Reports components.

- History Database REST API

  Provides access to History Database content. This API is only available with valid History database license.

Access to the REST API

To access REST API via Swagger UI see more details in the section "Using Swagger UI" in chapter "Managing DirX Audit Manager and REST API" of this guide.

Swagger and OpenAPI endpoints are available without logging in. So are GET methods of I18n REST API. To access the rest of the endpoints, user needs to be authenticated using one of provided authentication methods. User needs to have appropriate permissions to access all endpoints apart from the ones mentioned before.

Required Audit roles are **Audit Administrator**, **Auditor** and **Restricted Auditor**. Users with roles Audit Administrator and Auditor have access to all endpoints. Restricted Auditor has access only to endpoints of Core and Configuration Reports REST APIs.

Provided authentication methods are:

- Basic authentication against LDAP server with session management enabled

  Basic authentication is only used when user is being logged in. During login, session is created which is used for the rest of the requests. Session validity is time restricted. See more in section "Session management" below.

- oAuth2 with OIDC

  When using DirX Audit Manager or Swagger UI to log in, user is redirected to the OIDC authorization server to be authenticated. After successful authentication, user is redirected back.

Session management

For managing sessions, default Spring framework session management is used. When user is authenticated and authorized, a session is created and stored in embedded Tomcat container. JSESSIONID cookie is also created and sent to the client for future use. Default idle validity of session is 1 hour and maximum age of session is 10 hours. This can be configured in the Tenant Configurator. See the section "REST Service Authentication Configuration" in the Tenant Configuration in the chapter "Configuring DirX Audit" in the *DirX Audit Installation Guide* for details.

History REST API

History REST API is only available in installation with valid History DB license.

Pagination

Pagination is realised using SCIM standard. On query, the response body contains number of total results, number of items per page and start index. Certain page and size of it can be set when querying the endpoint. If not set, default values are used. Default page size is 1000 and default start index is 1.

Caching

Endpoints which load big data sets (such as Audit Messages or Audit Events) cache the results for a limited time. Entries are held in cache for a maximum period of 24 hours. Some entries, like Related Events in Audit Messages endpoints are stored in cache based on interval of context records job. Job calculating context records runs by default every 5 minutes, therefore the data may change and so Related Events are stored in cache only for 5 minutes.

Caching can be disabled globally in DirX Audit Configurator (See "REST Service Configuration" in the Tenant Configuration in the chapter "Configuring DirX Audit" in the *DirX Audit Installation Guide* for details.) or locally as endpoint parameter.

5. Managing DirX Audit Databases

DirX Audit stores configuration data, audit messages and history entries in a relational database. The configuration data, audit messages and history entries can be stored in the same database or in separate databases provided that the database server type is the same for the data DB containing audit messages and history DB containing history entries. Each tenant has its own one or more databases. If you want to configure more than one tenant, you must prepare and manage as many databases for as many tenants as you want to set up.

DirX Audit supports the following relational database management systems:

- SQL Server
- Oracle Database

The next sections describe how to:

- Set up DirX Audit Database management
- Manage DirX Audit Database connectivity
- Manage SQL Server or Oracle Database
- Manage audit messages data
- Manage history entries data

5.1. Setting up Database Management

To set up DirX Audit Database management:

1. Choose a relational database management server.
2. Choose a connector / driver.
3. Create the technical account(s).
4. Create the database(s).
5. Create the database objects – tables, views and indexes.

The next sections describe how to perform these tasks.

5.1.1. Choosing a Relational Database Management Server

You can choose from the list of supported relational database management servers. You can make an individual decision for the audit messages database (Data DB) and for the configuration database (Config DB) but the database type for the audit messages database and history entries database (History DB) must be the same.

5.1.2. Choosing a Connector / Driver

A connector / driver for the SQL Server is pre-installed with DirX Audit Manager Classic and DirX Audit Server. We recommend that you do not change these settings.

A connector / driver for Oracle Database must be installed separately. See the section "Oracle Database JDBC Driver Installation" in "Installation Configurations" in the *DirX Audit Installation Guide* for instructions on how to install it.

5.1.3. Creating the Technical Accounts

Create one, two or three technical accounts for connecting to the database(s) by following the instructions in the relational database management server documentation.

Note that they need the access rights to create tables, indexed / materialized views, indexes and procedures and write records. On Oracle Database, the account must also be allowed to create and execute triggers and jobs and work with full-text structures.

When using Oracle Database, the account should have at least the following privileges:

CREATE_TABLE
CREATE_INDEXTYPE
CREATE_TYPE
CREATE_VIEW
CREATE_MATERIALIZED_VIEW
CREATE_SEQUENCE
CREATE_PROCEDURE
CREATE_JOB

and should have the following roles:

CONNECT
RESOURCE
CTXAPP

5.1.4. Creating the Database(s)

Create one, two or three databases to store your audit messages (Data DB), configuration data (Config DB) and history entries (History DB). Make the technical account(s) – an owner of the database(s). Use the relational database management server documentation to perform this task.

Follow the following rules when naming databases:

- The database names must not contain the dash sign. You can use the underscore sign instead.
- The database names should be upper case.

5.1.5. Creating the Database Objects – Tables, Views and Indexes

The Tenant Configuration Wizard can create Data DB objects for audit messages, Config DB objects for configuration data and History DB objects for history entries including tables, views and indexes automatically when configuring the databases and validating their contents. You can run SQL scripts located on the installation media to create the set of tables, views and indexes manually. The provided indexes support DirX Audit Manager and DirX Audit Manager Classic Audit analysis filters and predefined reports distributed with the product. Use the following scripts with your database administration tool in the order given here:

- For SQL Server:
  install_media/**Additions/Scripts/MSSQL/create_data_tables.sql**
  install_media/**Additions/Scripts/MSSQL/create_data_indexes.sql**
  install_media/**Additions/Scripts/MSSQL/create_data_procedures.sql**
  install_media/**Additions/Scripts/MSSQL/create_config.sql**
  install_media/**Additions/Scripts/MSSQL/create_config_indexes.sql**
  install_media/**Additions/Scripts/MSSQL/create_history_oltp_tables.sql**
  install_media/**Additions/Scripts/MSSQL/create_history_oltp_indexes.sql**
  install_media/**Additions/Scripts/MSSQL/create_history_olap_tables.sql**
  install_media/**Additions/Scripts/MSSQL/create_history_olap_indexes.sql**
  install_media/**Additions/Scripts/MSSQL/create_history_olap_views.sql**
  install_media/**Additions/Scripts/MSSQL/create_history_procedures.sql**
- For Oracle Database:
  install_media/**Additions/Scripts/Oracle/create_data_tables.sql**
  install_media/**Additions/Scripts/Oracle/create_data_indexes.sql**
  install_media/**Additions/Scripts/Oracle/create_data_procedures.sql**
  install_media/**Additions/Scripts/Oracle/create_config.sql**
  install_media/**Additions/Scripts/Oracle/create_config_indexes.sql**
  install_media/**Additions/Scripts/Oracle/create_history_oltp_tables.sql**
  install_media/**Additions/Scripts/Oracle/create_history_oltp_indexes.sql**
  install_media/**Additions/Scripts/Oracle/create_history_olap_tables.sql**
  install_media/**Additions/Scripts/Oracle/create_history_olap_indexes.sql**
  install_media/**Additions/Scripts/Oracle/create_history_olap_views.sql**
  install_media/**Additions/Scripts/Oracle/create_history_procedures.sql**
  install_media/**Additions/Scripts/Oracle/create_hibernate_sequence.sql**

The last script has to be run in all used databases: Data DB, Config DB and History DB.

If you are upgrading DirX Audit, please follow instructions in the *DirX Audit Migration Guide*.

Determine whether additional indexes should be created when designing new filters in the DirX Audit Manager Audit analysis or reports. Query processing is highly dependent on the database indexes. Use the relational database management server documentation to perform this task.

If you selected to use the full-text search in the database configuration, you must also create the full-text database structures. Use the script **create_data_fulltext.sql** in the correct folder base on the database server type to create the full-text structures and **create_data_jobs.sql** to set up a job for optimizing it, Oracle Database only.

For Oracle Database, the full-text search index is synchronized on each commit. This configuration causes fragmentation and degradation of the index over time. Therefore, a job is created to optimize it once a day. See the script **create_data_jobs.sql**. If you encounter performance or space problems with this configuration, you should change the configuration of the full-text index. For example, synchronize every hour and optimize or rebuild on a weekend. Consult with your database administrator to determine the best approach.

The created materialized views on Oracle Database are refreshed by default once a day at 5 A.M. See the section "Managing Oracle Databases" for details.

5.2. Managing DirX Audit Database Connectivity

You can configure database connectivity with the Tenant Configuration Wizard. You can configure the following connections:

- DirX Audit connectivity to the database that contains the audit messages (Data DB). See the section "Data DB Configuration" in "Configuring DirX Audit" in the *DirX Audit Installation Guide* for details.
- DirX Audit connectivity to the database that contains the configuration data (Config DB). See the section "Config DB Configuration" in "Configuring DirX Audit" in the *DirX Audit Installation Guide* for details.
- DirX Audit connectivity to the database that contains the history entries (History DB). See the section "History DB Configuration" in "Configuring DirX Audit" in the *DirX Audit Installation Guide* for details.

5.3. Managing SQL Server

SQL Server has the following pre-requisites:

- The database server must have TCP/IP connection enabled.
- The port must be set explicitly.
- The Collation option must be set explicitly for databases.
- The default language for the technical account(s) must be English.

5.4. Managing Oracle Databases

When you are using Oracle Database, be aware that the Database Content dialog's Oracle Text option must be checked when the database is created using dbca.

When you are using Oracle Database, perform the following step before running the software:

- Create the DXT_LOGINTRG trigger, if it does not already exist: run the script *install_media***\Additions\Scripts\Oracle\create_trigger_dxt_logintrg.sql** in your Oracle Database management tool. Be sure to set the user/schema name properly in the script.

When you are using Oracle Database, be aware that materialized views for the History Database are refreshed by default at 5 A.M. each day. The views are used for the History view and risk-related dashboard components in the DirX Audit Manager and for reports. If you wish to refresh the materialized views manually, you can call the **DXT_history_view_refresh** procedure. The creation script for this procedure is available at *install_media***\Additions\Scripts\Oracle\create_history_procedures.sql**. Use the script to create the procedure if it does not already exist.

If you want to run the materialized views refresh on a different schedule, drop the views, modify the *install_media***\Additions\Scripts\Oracle\create_history_olap_views.sql** script and then run it to recreate the views. Alternatively, you can alter the views.

5.5. Managing Audit Messages Data

In the first approach, customers tend to audit everything to be sure that nothing is lost for later exploration. The result is a fast growing DirX Audit Database.

We strongly recommend reducing the created audit data as much as possible to the amount that is really needed. There are many strategies on how to achieve this reduction. The number of created messages is highly dependent on the configured source systems, so it makes the most sense to optimize the source systems.

The following sections provide some hints and describe some data management strategies.

5.5.1. DirX Identity Audit Messages Data

This section provides information about typical audit messages originated in DirX Identity and strategies for controlling them.

5.5.1.1. Typical Number and Size of Audit Messages

Practical experience with a medium customer installation of about 20,000 identities that runs about 1,000 request workflows per day and that uses real-time and event-based workflows intensively provided the following numbers:

- Audit messages per day: about 50,000 (the size of one message is about 10 kB)

- Audit messages per month: about 1.5 million
- Resulting DB size per month: about 15 GB

For three months (90 days), about 45 GB are required. Purging this amount of data for long-time storage (see the section “Purging Parts of the Database”) would result in about 2 GB of data.

A rule of thumb is that you need about 100 GB of disk space for about 10,000 identities for the database and about 3–5 GB in purged form. These numbers can vary a lot depending on the customer scenario that is run, the dynamic activity in the system and the audit configuration in DirX Identity.

5.5.1.2. Strategies for Controlling the Number and Size of Audit Messages

To control the number and size of audit messages, adjust the following settings:

- Activate audit policies only for object types that are really needed.
- Reduce the number of identifying attributes in activated audit policies. Use attributes with short values wherever possible. We recommend using unique identifiers like employee numbers or pseudonyms. Note that these attributes are part of every audit message that is produced for the object type.
- Reduce the number of audited attributes in activated audit policies. Use attributes with short values wherever possible.
- Decide whether you really need system signature for audit messages. Using it almost doubles the message size.
- Decide whether you really need client signature for audit messages because this requires additional space, too.
- Decide whether you really need to store original audit messages into the DirX Audit Database.

5.5.2. DirX Access Audit Messages Data

This section provides information about typical audit messages originated in DirX Access and strategies for controlling them.

5.5.2.1. Typical Number and Size of Audit Messages

Practical experience with a medium-size customer installation of about 20,000 identities that runs web single sign-on for several web applications provided the following numbers:

- Audit messages per day: about 40,000 (message size is about 20 kB).
- Audit messages per month: about 1.2 million.
- Resulting DB size per month: about 24 GB, of which 75% represents the original message delivered from DirX Access to DirX Audit.

For three months (90 days), about 75 GB are required. Purging this amount of data for long-time storage (see the section "Purging Parts of the Database") would result in about 5 GB of data.

A rule of thumb is that you need about 200 GB of disk space for about 10,000 identities for the database and about 5–10 GB in purged form. These numbers can vary a lot depending on the customer scenario that is run, the dynamic activity in the system and the audit configuration in DirX Access.

5.5.2.2. Strategies for Controlling the Number and Size of Audit Messages

To control the number and size of audit messages, adjust the following settings:

- Reduce the number of identifying attributes in activated audit policies. Use attributes with short values wherever possible. Note that these attributes are part of every audit message that is produced for this object type.
- Decide whether you really need to store original audit messages into the DirX Audit Database.

5.5.3. Purging Parts of the Database

If the size of the DirX Audit Database is about to reach the system limits, you should purge a set of messages. Use the DirX Audit Database maintenance tool (DirX Audit DB tool) to perform this task. A regular system task can be scheduled when using the tool with relative time parameters. See "Using the DirX Audit Tools" in the *DirX Audit Command Line Interface Guide* for details.

Data exported with this utility is written in compressed XML files. Compared to the content in the database, this action reduces the size to about 3 to 5 percent.

Purged data is structured in year, month, and daily folders. Thus you can easily restore parts of this data to your standard database or to another one for specific analysis. If the audit messages restored from the archive have never been covered by fact population, you need to run also the fact population tool with a time range that contains the restored messages.

5.5.4. Identifying Duplicated Audit Messages

The **Identification – UID** (unique identifier) field is required and is constrained to be unique. If the value is missing when the audit message is collected and to be persisted, DirX Audit Server persistent unit generates a new value using the java.util.UUID class and calling its randomUUID() method. The generated **Identification – UID** value is extended with a configurable prefix, **dxt_** by default. You can set a different prefix value; see the section "Customizing the Identification – UID Prefix" for details.

The **Identification – UID** value can be missing accidentally in audit messages collected from DirX Identity or DirX Access products, but this always indicates a failure in the audited system and should be reported to product support along with the original message. On the other hand, when auditing a third-party system, the **Identification – UID** can be missing intentionally because the audited system cannot produce any unique identifier value for audit messages. In these cases, a new unique identifier value is always generated. Please

pay special attention when designing your system to guarantee that the audit message is not collected repeatedly. This action leads to duplication of the audit message in the DirX Audit Database with a different **Identification – UID** value. Other fields are identical.

You can search for potential duplications by running a SQL query over the DirX Audit Database.

```
select
 am1.IDENTIFICATION_UID 'AM1_IDENTIFICATION_UID'
 , am2.IDENTIFICATION_UID 'AM2_IDENTIFICATION_UID'
from
 DAT_AUDITMESSAGES am1
 join DAT_AUDITMESSAGES am2 on
  am1.IDENTIFICATION_WHEN = am2.IDENTIFICATION_WHEN
   and am1.IDENTIFICATION_SOURCE = am2.IDENTIFICATION_SOURCE
   and am1.IDENTIFICATION_UID < am2.IDENTIFICATION_UID
where
 am1.IDENTIFICATION_UID like 'dxt_%'
  and am2.IDENTIFICATION_UID like 'dxt_%'
```

You can extend the join condition with a test on equality of values in additional columns like:
IDENTIFICATION_CATEGORY,
IDENTIFICATION_CAUSE,
IDENTIFICATION_OP,
IDENTIFICATION_OUTCOME or
IDENTIFICATION_TYPE.

You can then delete duplicated audit messages with a query like the following:

```
delete from
 DAT_AUDITMESSAGES
where
 IDENTIFICATION_UID = '{PROVIDE_SPECIFIC_VALUE_HERE}'
```

This query cascades deletion to the rows in related tables.

5.5.5. Customizing the Identification – UID Prefix

You can change the **Identification – UID** prefix by setting the property for it in the *install_path*/**conf/configuration.cfg** file. This file is created and later updated by the Core Configuration Wizard during the first configuration.

The property key for the UID prefix is **uid_prefix** in the **[general]** section. The Core Configuration Wizard does not set this particular property in this file. You need to add it manually.

If the value is not set in the file, the default value **dxt_** is used.

5.6. Managing History Entries Data

When you are going to use the History Database, plan enough disk space. Here are some numbers that help you to calculate the space needed for your environment. They were observed with SQL Server.

Tests were run with N * 10,000 entries, each entry with 24 small attributes, 28 link attributes and 1 large attribute (description). The results show that the consumed space increases nearly linearly and that about 180 MB are required per 10,000 entries. Changing 10 attributes (6 small and 4 link attributes) adds another 30 MB per 10,000 entries.

An observation from a project confirms that a history entry-related data can consume several 10s kB in a database.

5.6.1. Calculating Foreign Keys

History database data must be processed to be available in views. During this procedure, foreign keys are calculated. A scheduled job running on the DirX Audit Server updates data only for the last several days. If you want to update older data, you can run the SQL queries delivered in *install_media***/Additions/Scripts/common/updatehistdb** manually. Remove the section that contains a placeholder that restricts the execution for the several last days from the where clause or optionally extend manually the existing **History DB update** scheduled job time range parameter **jobdatamap.arg_membership_validfrom** in the **route-dxt-scheduler-updatehistdb.xml** file.

6. Configuring DirX Audit Collectors

DirX Audit contains the following types of generic collectors:

- JMS collectors
- File collectors

These collectors support a proprietary generic audit message format initially modeled on the *RFC 3881 Security Audit and Access Accountability Message XML* format. You can find more information about the RFC 3881 standard at the URL http://www.faqs.org/rfcs/rfc3881.html. The audit message format was significantly modified between DirX Audit V2.0 and DirX Audit V3.0.

DirX Audit also provides product-specific collectors that collect audit messages from the following products:

- DirX Identity
- DirX Access

The next sections describe how to configure these collectors.

For details on specific input validation and processing in File and JMS collectors, see the section "Specific input validation and processing in File and JMS collectors" in "Managing DirX Audit Server Error Handling".

6.1. Configuring Generic Collectors

The generic collectors can read audit messages in the DirX Audit format. The DirX Audit XML schema file AuditMessages.xsd uses the XML namespace "urn:com:siemens:dxt:persistence:schemadb:2.0". The next sections describe how to configure the generic collectors.

The following figure provides an overview of the data flow of the generic collectors:

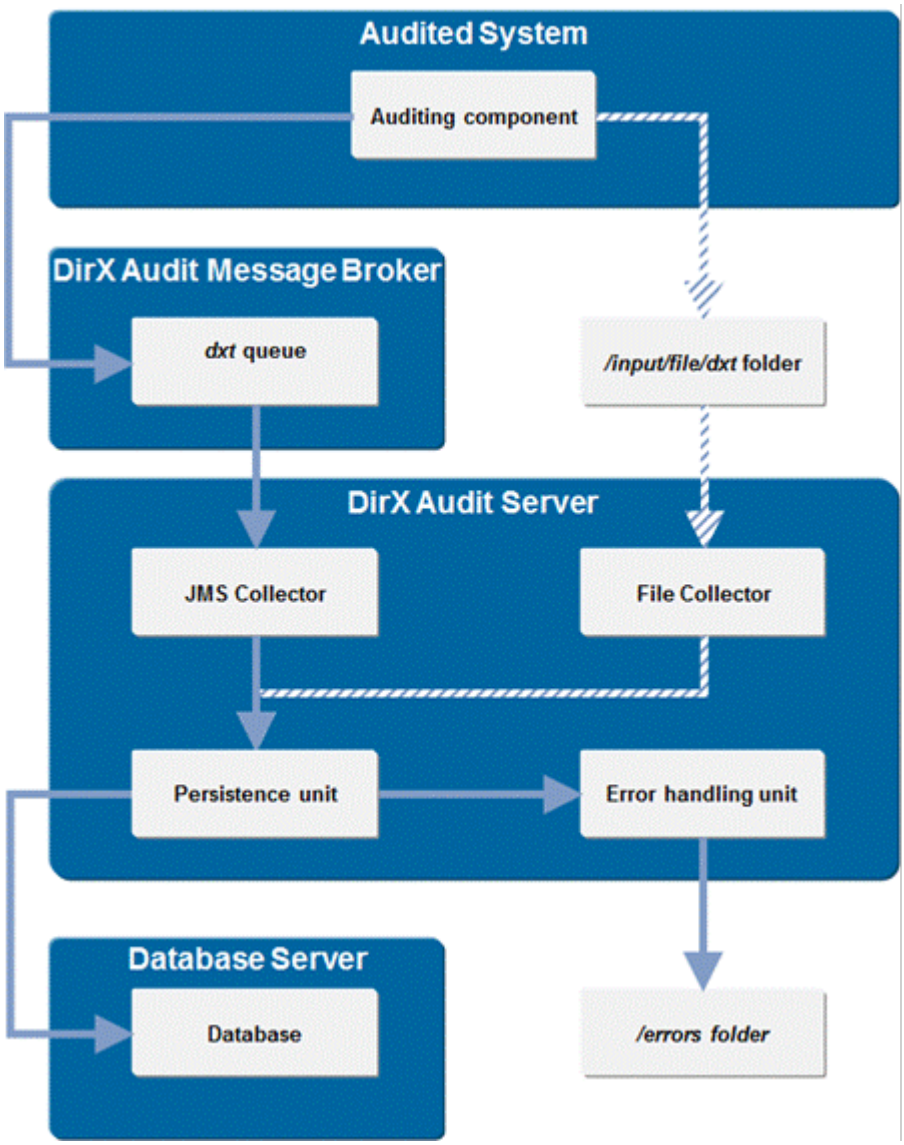

Figure 11. Data Flows when Using Generic Collectors

6.1.1. Configuring the Generic JMS Collector

For details on configuring generic JMS collector, see the section "Server JMS Collector for DirX Audit Format" in "Configuring DirX Audit" in the *DirX Audit Installation Guide*.

6.1.2. Configuring the Generic File Collector

For details on configuring the generic file collector, see the section "Server File Collector for DirX Audit Format" in "Configuring DirX Audit" in the *DirX Audit Installation Guide*.

6.2. Configuring DirX Identity Collectors

DirX Identity provides audit messages in several locations:

- In the dxrHistory attribute of LDAP entries; these messages are called “history audit messages”. DirX Identity workflows can optionally export these audit messages to files.
- In JMS messages produced by the JMS-Audit Handler hosted in the IdS-J and covering workflow events; these messages are called “workflow audit messages”.
- In files produced by the File Audit Handler hosted in the IdS-J covering workflow events; these messages are called “workflow audit messages”.

The JMS and the File Audit Handler should be used alternatively.

DirX Audit provides collectors to read DirX Identity audit messages from the following sources:

- Directory services via LDAP
- JMS message queues
- Files

The following figure provides an overview of the DirX Identity collectors:

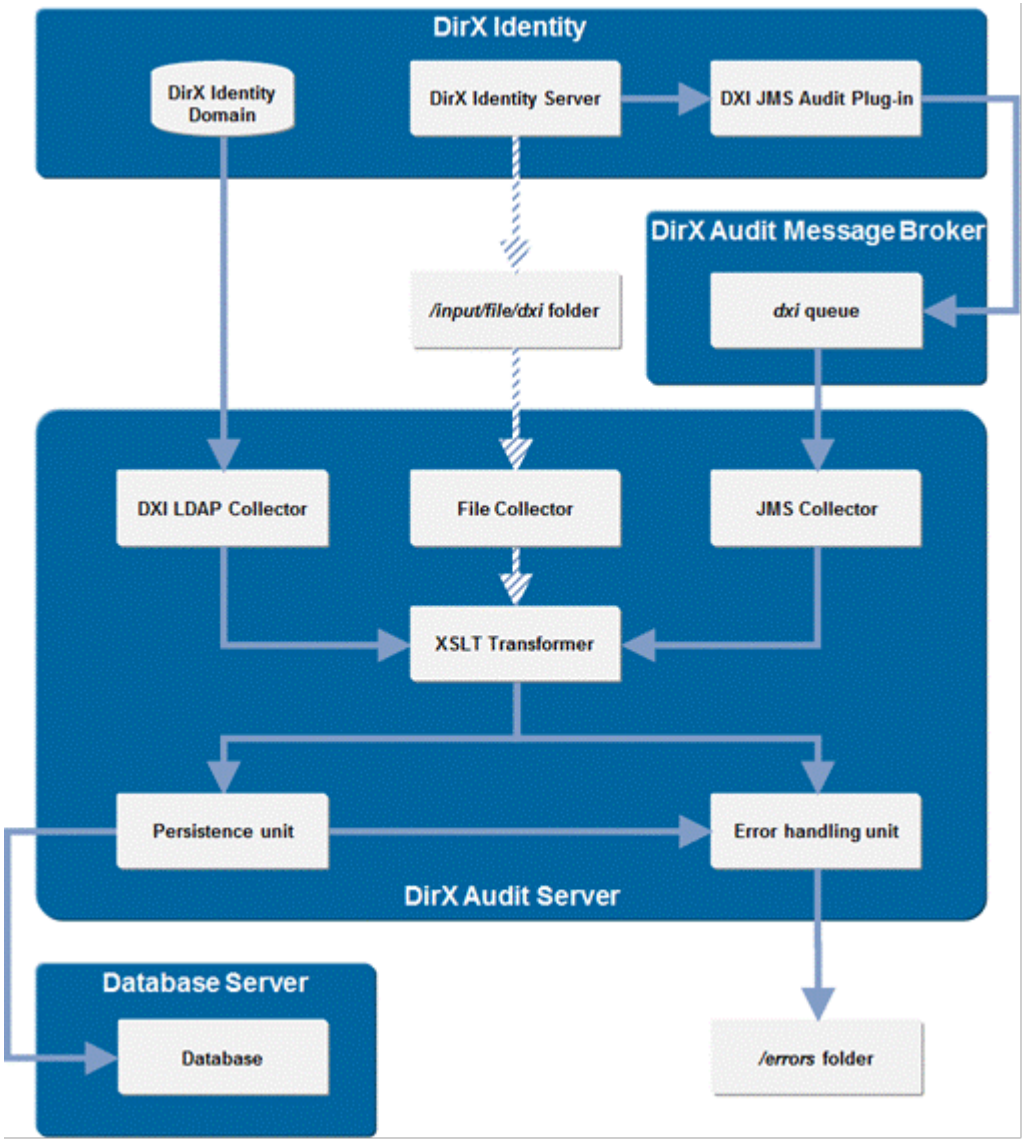

Figure 12. Data Flows when Using DirX Identity Collectors

6.2.1. About the DirX Identity JMS-Audit Handler Plug-in

DirX Identity can automatically deliver workflow audit messages to a message queue using the JMS-Audit Handler plug-in delivered with DirX Audit. See the chapter "Installing the DirX Identity JMS-Audit Handler Plug-in" in the *DirX Audit Installation Guide* for instructions on how to install the DirX Identity JMS-Audit Handler plug-in.

6.2.2. Configuring DirX Identity LDAP Collectors

For details on configuring DirX Identity LDAP collectors, see the section "Server LDAP Collector for DirX Identity Format" in "Configuring DirX Audit" in the *DirX Audit Installation Guide*.

6.2.3. Configuring DirX Identity JMS Collectors

For details on configuring DirX Identity JMS collectors, see the section "Server JMS Collector for DirX Identity Format" in "Configuring DirX Audit" in the *DirX Audit Installation Guide*.

6.2.4. Configuring DirX Identity File Collectors

For details on configuring DirX Identity file collectors, see the section "Server File Collector for DirX Identity Format" in "Configuring DirX Audit" in the *DirX Audit Installation Guide*.

6.3. Configuring the DirX Access JMS Collector

DirX Access produces audit messages using its proprietary format. The DirX Access JMS-Audit Handler Plug-in transforms the data into DirX Audit format and sends it to a message queue. The next sections describe how to configure the DirX Access JMS collector.

The following figure provides an overview of the DirX Access JMS collector data flow:

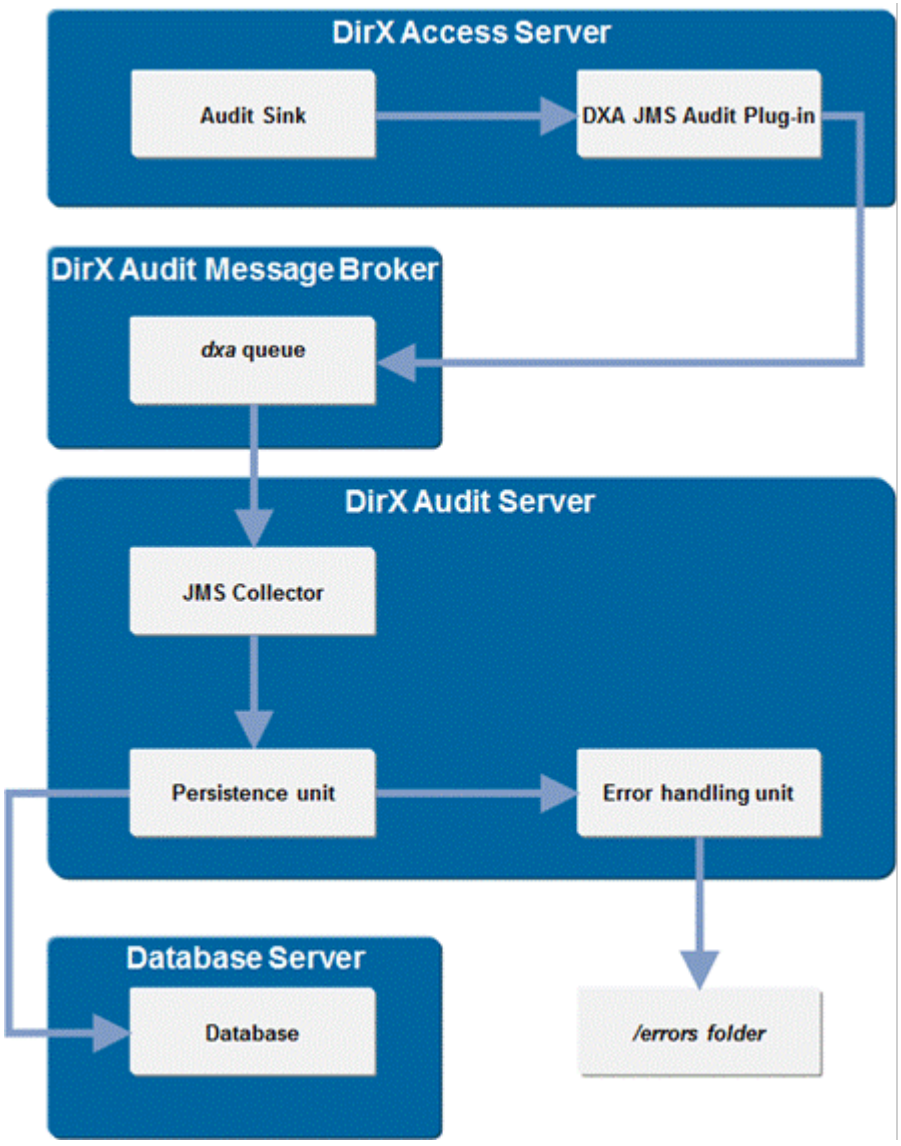

Figure 13. Data Flows when Using the DirX Access JMS Collector

6.3.1. About the DirX Access JMS-Audit Handler Plug-in

The DirX Access JMS-Audit Handler plug-in allows DirX Access to use the DirX Audit JMS collector to track DirX Access audit events. See the chapter "Installing DirX Access JMS-Audit Handler Plug-in" in the *DirX Audit Installation Guide* for information on how to install the DirX Access JMS-Audit Handler plug-in.

6.3.2. Configuring the DirX Access JMS Collector

For details on configuring the DirX Access JMS collector, see the section "Server JMS Collector for DirX Access Format" in "Configuring DirX Audit" in the *DirX Audit Installation Guide*.

7. Managing DirX Audit Server Error Handling

This chapter provides information on error handling inside DirX Audit Server, including:

- How DirX Audit Server error handling works
- Error message structure
- Configuring DirX Audit Server error handling
- Recoverable error handling
- Non-recoverable error handling
- Specific input validation and processing in File and JMS collectors

The standard notation for the Java datetime formatter pattern is used for the datetime pattern values used in folder and file names in this section.

7.1. About DirX Audit Server Error Handling

A set of collector routes are created and run for every tenant. Also a separate instance of DirX Audit Server is used for every tenant starting with version 7.2. Each collector route has the following parts that are connected by the routing mechanisms:

- An input endpoint (collector) that collects incoming records from external system.
- An optional transformer that transforms the external record format into DirX Audit message if needed.
- A Persistence unit that creates the final audit message, extends it with digest and dimensions and stores it into the Database.

An error can occur at any of these steps when processing an incoming record. DirX Audit Server makes sure that if such an error occurs, the relevant record is not lost but is either correctly processed or stored with additional information into error storage for later automatic or manual processing.

The incoming records can contain one or more target messages (for example, when using the file or the LDAP collector). If an error occurs when processing such a record, the contained messages are cut to single messages by the splitter component and are processed one by one (so that only the ones that are causing errors are stored in error storage when needed).

The record is stored into error storage if an error occurs and it can’t be resolved by the DirX Audit Server.

The error storage component stores errors into files. For every error, it stores both the original record and the information about the error (type, exception and related properties).

The record(s) are kept in the source system and should be picked up again later by the collector (on the next poll) in the following cases:

- If the error occurs during initial reading in the collector (before passing to the transformer or the Persistence unit)
- If the error is not handled correctly by the Persistence unit or the error storage component

If the error occurs later in the transformer or Persistence unit (but is handled by them or the error storage component), the incoming record is removed from the source system and is either correctly reprocessed by the DirX Audit Server or stored into the error storage.

Two main types of error can occur during processing:

- A recoverable error – an error occurring on the Persistence unit caused by a temporary problem with the database (for example, a network connectivity problem). These errors are expected to be corrected in a short time period. The DirX Audit Server tries to reprocess the affected messages twice with a short delay. If it isn't successful, the message is stored in a special folder and it's reprocessed later automatically. Error files are stored in type folder: **100-recoverable**.
- A non-recoverable error – an error caused by the record content (for example, invalid format or missing required fields). These records can't be processed and are stored as files in the error storage. These errors must be analyzed and processed manually. Error files are stored in a type folder (if not one of the sub-types below): **200-nonrecoverable**.

There are two sub-types that are stored in separate folders:

- Duplicate – The incoming record was already stored in the database. These errors can be ignored. An increasing number of stored duplicates can indicate other problems in the installed systems (e.g. frequent network interruptions) and should be analyzed by the administrator. Error files are stored in type folder: **300-duplicates**.
- XML processing error – The error occurred when parsing or transforming the incoming record. These errors should be analyzed, and a correction should be applied on the source or in the transformation. Error files are stored in the type folder: **250-nonrecoverable-xml**.

7.2. Error Message Structure

The base folder is configurable in the Tenant Configuration Wizard. All subfolders (for every error type and sub-type) within this folder are created automatically when first needed. The following type subfolders can be created:

100-recoverable

200-nonrecoverable

250-nonrecoverable-xml

300-duplicates

When an error is being stored, additional subfolders are created within the corresponding type subfolder – based on the current date. Two levels are created: the first is based on current year and month (in the format *yyyy_MM*) and second based on current date (in the format _dd_). For example, **2020_06/02** (for June 2nd). For each error, two files are created in the final folder: a **content** file (zipped) containing the original record (file name format *yyyyMMdd_HHmmssSSS-source_components_info*.**content.zip**) and an **info** file containing additional error information (file name format *yyyyMMdd_HHmmssSSS-source_components_info*.**info**).

Here is an example of the full paths of the two files (for a duplicate coming through the DirX Identity File collector):

```
300-duplicates/2020_06/02/20200602_094739922-dxi_file-pers.info

300-duplicates/2020_06/02/20200602_094739922-dxi_file-pers.content.zip
```

7.3. Configuring DirX Audit Server Error Handling

For information on how to configure DirX Audit error handling, see the section "Server Error Handling" in "Configuring DirX Audit" in the *DirX Audit Installation Guide*.

7.4. Error Handling for Recoverable Errors

If a recoverable error is stored in error storage, it is automatically processed again at least three times. If any of these attempts is successful (the stored record is persisted into the database) the stored error is removed along with the corresponding information file and attempt information (if created before).

If the attempt is not successful (an error occurs), the stored error is kept in the folder and the information about the processing error is stored in an additional file (a file name with the suffix **.redelivery.properties**). Only one additional file is kept (the older ones are rewritten if they exist).

There are two jobs scheduled by default for this processing with different schedules and scopes:

- Daily recovery processor (named **recovery_1**) – scheduled to run every day at 7pm with scope of last two days.
- Weekly recovery processor (named **recovery_2**) – scheduled to run every Sunday at 5am with scope of last seven days.

Each job tries to redeliver all errors stored within the given scope. Every stored error is tried once during a single job execution.

The jobs configuration can be modified manually by editing configuration files. Either the default file can be modified (then all tenants will use the changed configuration) or the tenant-specific configuration file can be modified (then only this tenant will use the changed configuration). The tenant-specific values overwrite the values from the default file.

The location of the default file is:
install_path/**conf/defaults/tenant/configuration.cfg**

The location of the tenant-specific configuration file is:
install_path/**conf/tenants/***tenant_id*/**configuration.cfg**

The sections relevant for the two jobs are (the jobs are simply numbered as 1 and 2 and differ only in schedule and scope and are otherwise functionally equal):

```
[server.apps.error_handling.recovery_1]
enabled = true
cron_expression = 0+0+19+?+*+*
scope_days = 2

[server.apps.error_handling.recovery_2]
enabled = true
cron_expression = 0+0+5+?+*+SUN
scope_days = 7
```

The following options can be set:

- **enabled** – Boolean value – whether (**true**) or not (**false**) the job will run.
- **cron_expression** – String value – a CRON-like expression for the job schedule.
- **scope_days** – Integer value – the number of past days to check for stored errors. For example, a value of 2 means today and two days before (yesterday and the day before yesterday).

You must reconfigure server jobs for the given tenant using the Tenant Configuration Wizard to apply the changed values (or restart the DirX Audit Server service for a given tenant).

The administrator should check all stored errors that remain in the recoverable errors folder after the configured maximum days scope for either one of the two jobs (by default, eight days). Such stored errors will no longer be processed automatically and should be analyzed and processed manually.

7.5. Error Handling for Non-recoverable Errors

All stored non-recoverable errors are not processed automatically and the administrator should analyze them manually.

Duplicates do not need to be processed since they are already in the database. If the number of stored duplicates keeps growing, the entire system should be checked since it indicates that there are other problems (for example, frequent network disconnections).

Stored XML errors should be always checked since these records are missing in the database. The typical reason can be wrong format (duplicated XML header, invalid XML syntax, collector type mismatch), invalid content (missing required elements/attributes) or an error in the transformation template (in case of different source format; for example, DirX Identity records). Once the problem is fixed, the stored files should be reprocessed manually, for example, using the corresponding File collector.

7.6. Specific Input Validation and Processing in File and JMS Collectors

File and JMS collectors perform additional validation before or during processing of input data. They have different ways of performing input validation and specific processing of the input.

File collectors will not try to process files that do not pass the basic validation checks. Error handling will not be used for such input files and those files will stay in input folder.

If the file passes all validation checks it is processed further. The file content is not processed at once. It is cut into groups of a given number or records (by default 10). These groups are then processed separately in sequence. Any error that happens during this cutting process will cause the whole file to be processed by error handling and stored into an error storage. It might happen that some of the file content have been already stored successfully into the database before this error occurred.

JMS collectors perform the basic validation check after the message is read. In the event that the content is not valid error handling will be processed. Such messages will be stored in an error storage.

The following sections provide more details on these checks.

7.6.1. File Collector Input Validation

The File collector performs the following validation checks before reading and later when processing a file from the input folder:

1. File name and file size reprocessing check – The File collector stores the name and size of every file that has been processed. If you later put into the input folder a file with exactly the same name and size it will not be processed, and the file will stay in the input folder. No log message is written into the log file unless a specific logger is set to DEBUG level. The list of processed files is kept in memory (by default the last 1000 processed files). The list is cleared when DirX Audit Server is restarted, or the collector route redeployed.
2. The file content is checked for basic conformance to the XML format expected for the given source product, for example DirX Identity. The file will not be processed if the content is not a valid XML or does not contain the expected root element and first sub-element. No log message is written into the log file, unless a specific logger is set to DEBUG level. If the file does not pass that check it will stay in the input folder.

3. The file name is checked against the configured file name mask, configured in the Tenant Configuration Wizard for the given File collector. Files that do not match the given mask are not being processed. No log entry is created in this case.

7.6.2. JMS Collector Input Validation

The JMS collector performs the following validation checks after reading a message from the input queue:

1. The message content is checked for basic conformance to the XML format expected for the given source product, for example DirX Identity. The message will be processed by error handling and stored into an error storage if the message content is not a valid XML or does not contain the expected root element and first sub-element. The message will be always removed from the input queue.

8. Managing a Multi-tenant Environment

This chapter provides information on how to manage the multi-tenant environment supported by DirX Audit.

8.1. About Multi-tenancy

DirX Identity can provide identity management for multiple customers, organizations or separate entities (tenants). It is implemented with one DirX Identity installation hosting several domains: one tenant per domain.

DirX Audit also provides for **multi-tenancy**: the ability to configure and operate multiple tenants, which are configuration entities that represent sets of audit database connection settings, authentication and authorization settings, lists of audit sources and server task configurations as well as DirX Audit Message Broker users and queues, all of which remain completely separate from each other within the DirX Audit installation while providing access for the group of authorized tenant specific users to the configured tenant-specific resources. A separate instance of DirX Audit Server and its system service is used for every tenant starting with version 7.2.

Only a single DirX Audit installation is necessary and it guarantees that users and administrators authorized for a particular tenant cannot view or access the data (audit events and history entries) contained in a different tenant. Tenants can practically shadow the structure of existing DirX Identity domains; that is, specific users from individual domains will have access to domain-specific audit events and history entries extracted from the specific DirX Identity domain and stored in tenant-specific DirX Audit Database processed by tenant-specific DirX Audit Server tasks.

The following figure illustrates the DirX Audit multi-tenant architecture.

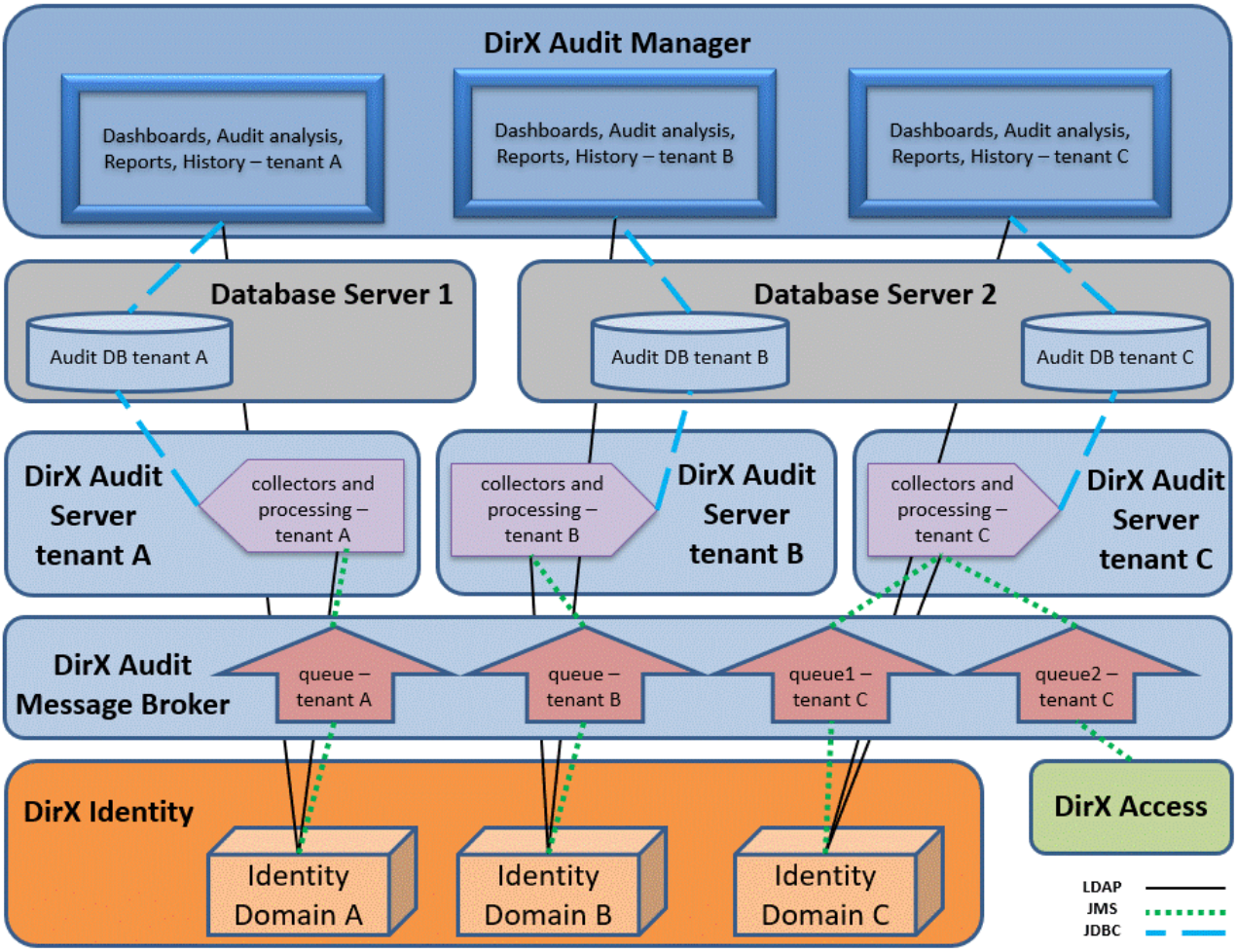

Figure 14. DirX Audit Tenant Architecture

8.2. Configuring and Using DirX Audit Multi-tenancy

This section provides information about DirX Audit's multi-tenancy features and how multi-tenancy impacts DirX Audit components like DirX Audit Manager and DirX Audit Manager Classic, DirX Audit Server processes, DirX Audit Message Broker and DirX Audit Tools.

8.2.1. Setting up Multiple Tenants

DirX Audit multi-tenancy has the following design features:

- A separate, dedicated Tenant Configuration Wizard is used to create, configure, modify and remove individual tenants. The Core Configuration Wizard configures the global settings. There are also two Start Menu shortcuts to launch the respective configuration wizards; however, it is also possible to seamlessly continue with the tenant configuration once the core configuration is finished. Also see the section "Using the Configuration Wizard for the Tenant Configuration" in the "Configuring DirX Audit" in the *DirX Audit Installation Guide* for further configuration details.
- Individual tenant configuration files are stored in their respective subfolders in the *install_path*/**conf/tenants/** folder, while the global settings are stored in the *install_path*/**conf/configuration.cfg** file.

- The tenant name is configurable, chosen by the user when a new tenant is created, while the tenant identifier is assigned automatically to ensure its uniqueness and cannot be changed by the user.
- The tenant configuration must be unique: no tenant can share the same DirX Audit Database, DirX Audit Server file collector folder and LDAP collector settings or DirX Audit Message Broker JMS queue with another tenant to prevent the mixing of data originating from different sources, domains or organizations. Only the DirX Audit Manager authentication settings can be shared between multiple tenants, but a Tenant Configuration Wizard warning will be displayed in these cases.

8.2.2. Using DirX Audit Manager Classic in a Multi-Tenant Environment

The multi-tenant environment has the following effect on DirX Audit Manager Classic:

- The tenant identifier must be specified in the URL and the respective tenant name is also displayed in the Login page if there are multiple configured tenants. For details, see the section "Logging In" in "Using the DirX Audit Manager Classic" in the *DirX Audit Manager Classic Guide*.
- Once the user logs into the DirX Audit Manager Classic, the tenant name is also displayed next to the username following the “@” separator if there are multiple configured tenants.
- All DirX Audit Manager Classic tabs - Dashboard, Audit analysis, Reports and History - display only data from relevant tenant-specific DirX Audit databases.
- The authentication and authorization settings are also tenant-specific, which means that only authorized users existing in configured tenant-specific authentication source systems (for example, specific DirX Identity domains) can view or access the data of the given tenant.

8.2.3. Using DirX Audit Manager in a Multi-Tenant Environment

The multi-tenant environment has the following effect on DirX Audit Manager:

- The tenant identifier must be specified in the URL. For details, see the section "Logging In" in the "Using the DirX Audit Manager" in the *DirX Audit Manager Guide*.
- All DirX Audit Manager tabs - Audit Analysis, History and Reports - display only data from relevant tenant-specific DirX Audit databases.
- The authentication and authorization settings are also tenant-specific, which means that only authorized users existing in configured tenant-specific authentication source systems (for example, specific DirX Identity domains) can view or access the data of the given tenant.
- Each DirX Audit Manager application uses the REST API of the corresponding DirX Audit Server of the same tenant. The REST API of that DirX Audit Server instance must be accessible from the client web browser where the DirX Audit Manager is accessed. As the REST API is required for the DirX Audit Manager to access the tenant-specific data and configuration, the DirX Audit Server instance service must be started and running before the DirX Audit Manager can be used to access the tenant-specific data (the Tomcat service hosting the DirX Audit Manager application should be configured to

start after the server service).

8.2.4. DirX Audit Server Processes in a Multi-Tenant Environment

A separate instance of DirX Audit Server is used for every tenant starting with version 7.2. DirX Audit Server operates with its specific DirX Audit Message Broker queues, connected DirX Identity domain, file collector folders, and DirX Audit databases.

DirX Audit Server instance also provides REST API for the DirX Audit Manager to access the tenant-specific data and configuration. The REST API of the DirX Audit Server instance must be accessible from the client web browser where the DirX Audit Manager is accessed. The DirX Audit Server instance service must be started and running before the DirX Audit Manager can be used to access the tenant-specific data (the Tomcat service hosting that DirX Audit Manager application should be configured to start after the server service).

8.2.5. DirX Audit Message Broker Operations in a Multi-Tenant Environment

The multi-tenant environment has the following effect on the DirX Audit Message Broker operation:

- The DirX Audit Message Broker configuration creates special queues for each collector and tenant that include the tenant identifier. There are also tenant-specific users with read and write access rights created for each tenant, each of them including the tenant identifier.
  For further configuration details, see the section "Common JMS Collector Credentials" and all Server JMS Collector sections in the "Using the Configuration Wizard for the Tenant Configuration" in the *DirX Audit Installation Guide*.
- When a tenant is removed, its queues will remain active without an associated consumer, such as a server endpoint, until they are deleted manually.

8.2.6. Using DirX Audit Tools in a Multi-Tenant Environment

The multi-tenant environment has the following effect on the DirX Audit Tools:

- Tool users must specify the tenant whose configuration, including the database connection setting, should be used to access the database to be processed by the tool. See the chapter "Using the DirX Audit Tools" in the *DirX Audit Command Line Interface Guide* for the command syntax description.

9. Understanding the Audit Message Logical Schema

This chapter describes the logical audit message layout as presented at the user interface level. There is also a section that describes the mapping of the message logical schema to the database schema, which consists of several tables.

The database schema is proprietary but is modeled on the RFC 3881 standard (Security Audit and Access Accountability Message XML) with some extensions. You can find more information about this standard at http://www.faqs.org/rfcs/rfc3881.html.

A message in DirX Audit consists of five sections:

Identification section – identification of the audit message with some basic information.

Where From section – location where the message originated.

Who section – a user, hardware device or software process that originated the event (subject or active participant).

What section (can occur multiple times) – object of the message (passive participant).

Original Message section – the entire original message that the audited system provided.

The following figure illustrates this layout and lists the attributes in each section:

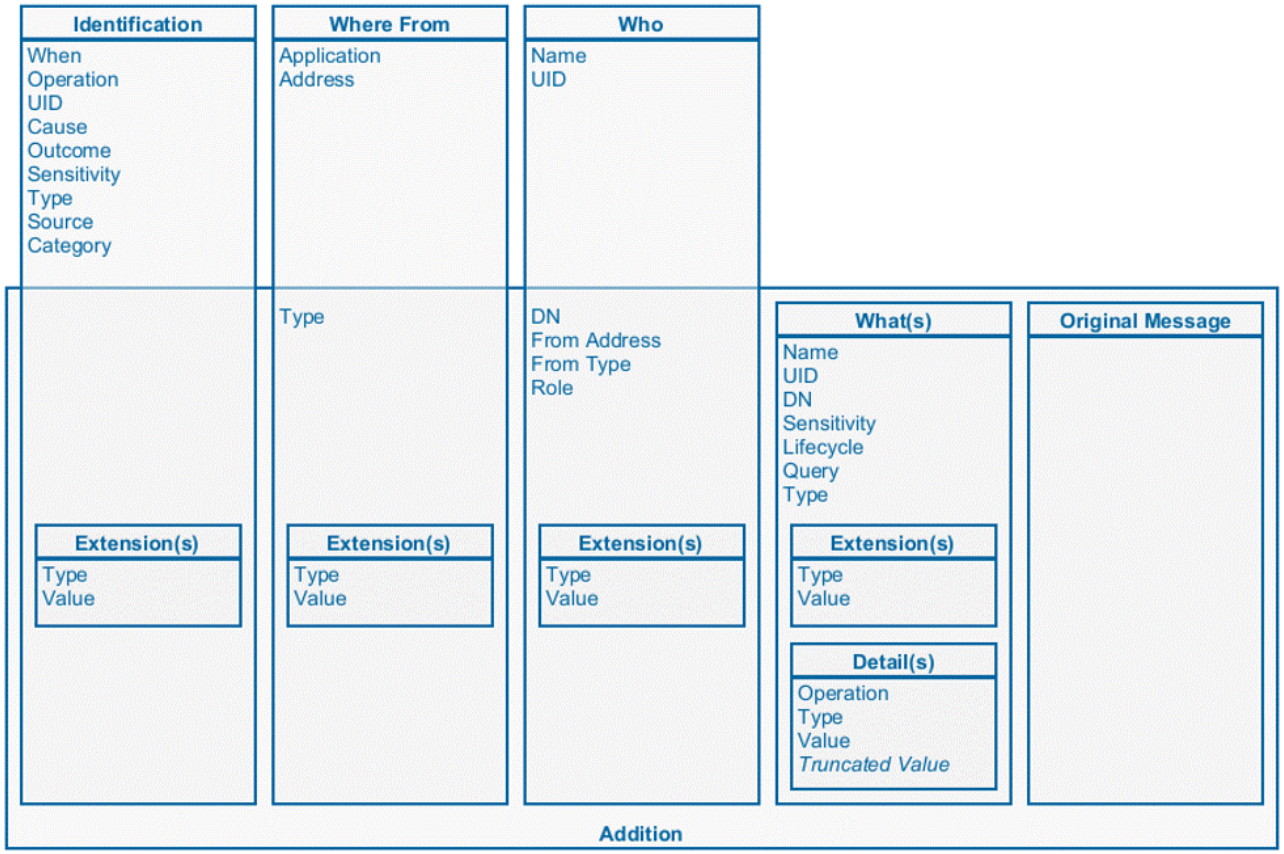

Figure 15. Audit Message Logical Schema

The next sections describe the attributes of these sections and how they are used by applications like DirX Identity or DirX Access.

This chapter also describes the mapping of the DirX Audit message logical schema to the database schema.

9.1. About the Identification Section

The Identification section describes the message itself with some basic information. The attributes of this section are:

When – (required) the universal coordinated time (UTC) when the event originated.

- Access – delivers a date and time stamp.
- Identity – delivers a date and time stamp.

Operation – (optional) an indicator for the type of action performed during the message (**C** – Create, **R** – Read, **U** – Update, **D** – Delete, **E** – Execute).

- Access – uses all codes.
- Identity – uses all codes besides R (Read).

UID – (optional) a unique identifier of the message. If this value is not supplied, DirX Audit Server calculates one.

- Access – delivers a unique identifier per message.
- Identity – delivers a unique identifier per message. This identifier can be part of the **Cause** field or the **Identification – Extension – Value** field, where **Identification – Extension – Type** is equal to **IDENTIFICATION_CAUSE**, of other messages to indicate relationship.

Cause – (optional) a unique identifier of the message that triggered this message.

- Access – the identifier of a user session.
- Identity – the identifier of the parent message, if any.

Outcome – (required) whether the message succeeded (0) or failed (4, 8, 12).

- Access – delivers 0 for OK, 4 for Minor Failure, 8 for Serious Failure and 12 for Major Failure.
- Identity – delivers 0 for OK and 8 for all other values.

Sensitivity – (optional) an arbitrary string value indicating the sensitivity of the message.

- Access – not set by default.
- Identity – not set by default.

Type – (optional) the type of the audited message.

- Access – delivers additional information in this field: more values are possible.
- Identity – delivers additional information in this field: manual, on event, on request, on schedule.

Source – (optional) an identification of the audited system.

- Access – delivers additional information in this field: DirX Access.
- Identity – delivers additional information in this field: DirX Identity.

Category – (optional) the category of the audited message.

- Access – delivers additional information in this field: Authentication, Authorization, Configuration, Federation, Policy, ServerLifecycle, Sso, User and so on.
- Identity – delivers additional information in this field: Object, Assignment, Workflow and so on.

9.1.1. About the Identification – Extension Section

This section can occur multiple times. It is an extensible structure containing type – value pairs.
The attributes of this section are:

Type – (required) a type of the extension.

Value – (required) a value of the extension.

9.2. About the Where From Section

The Where From section describes the location at which the message originated.
The attributes of this section are:

Application – (optional) the name of the application where the message originated.

- Access – delivers the component name.
- Identity – delivers the component name.

Address – (required) the address of the audit source (for example, the LDAP server).

- Access – the physical (TCP/IP) or logical (DNS) address of the machine where the message occurred.
- Identity – the physical (TCP/IP) or logical (DNS) address of the machine where the message occurred.

Type – (optional) the type of the audit source.

- Access – not used.
- Identity – not used.

9.2.1. About the Where From – Extension Section

This section can occur multiple times. It is an extensible structure containing type – value pairs. The attributes of this section are:

Type – (required) a type of the extension.

Value – (required) a value of the extension.

9.3. About the Who Section

The Who section provides information about the actor who triggered the message. In the RFC 3881 standard, this section is called the **Active Participant**. The attributes of this section are:

Name – (required) the human-meaningful name of the subject.

- Access – the human-readable name of the user or other meaningful information.
- Identity – the name of the subject. You can set up this information for the service layer via audit policies if you set the Name property.

UID – (optional) the unique identifier of the subject.

- Access – the subject identifier.
- Identity – the dxrUID attribute of the subject. This value is an auto-generated number in DirX Identity.

DN – (optional) the structured identifier of the subject.

- Access – the distinguished name (DN) of the subject who originated the message.
- Identity – the distinguished name (DN) of the subject who originated the message.

From Address – (optional) the logical network location of the application activity.

- Access – for authentication messages, the address at which the authentication was performed.
- Identity – the address of the client application (when available).

From Type – (optional) an identifier for the type of network access point (1 – Machine Name, 2 – IP Address).

- Access – a value according to the From Address field or the fixed value **0**.
- Identity – a value according to the From Address field or the fixed value **0**.

Role – (optional) the role the subject played when performing the message, as assigned in role-based access control security.

- Access – a list of the user's roles.
- Identity – not used.

9.3.1. About the Who – Extension Section

This section can occur multiple times. It is an extensible structure containing type – value pairs.
The attributes of this section are:

Type – (required) a type of the extension.

Value – (required) a value of the extension.

- Access – information such as credential information.
- Identity – You can set up this information for the service layer via audit policies. It is recommended to follow personal data protection rules and use pseudonyms. Examples: `cn=D638471, employeeNumber=1058347296`.

9.4. About the What Section

The What section provides information about the changed object(s) of the message. In the RFC 3881 standard, this section is called the **Participant Object**. The attributes of this section are:

Name – (required) an instance-specific descriptor of the audited object (for example, a person's identifier).

- Access – information about the resource the user tried to access (for example, a URL).
- Identity – the name of the object. You can set up this information for the service layer via audit policies if you set the Name property.

UID – (optional) the unique identifier of the object.

- Access – not used.
- Identity – the dxrUID attribute of the subject. This value is an auto-generated number in DirX Identity.

DN – (optional) the structured identifier of the object.

- Access – the fixed value **N/A**.
- Identity – the distinguished name (DN) of the object.

Sensitivity – (optional) the policy-defined sensitivity for the object, such as VIP, high importance objects, or similar topics.

- Access – not used.
- Identity – not used.

Lifecycle – (optional) an identifier for the object's life-cycle stage.

- Access – not used.
- Identity – information about the change operation on the "what" object that relates its life-cycle in the Identity Store. For example, when a user-role assignment is removed, the operation for the entire message is "Delete". However, there are four "what" objects associated with this operation: the workflow, the user, the role, and the assignment. The Lifecycle value for "assignment" is Delete (the assignment was removed from the Identity Store), while the value for workflow, object and user is "Info" (their life-cycles in the Identity Store were not affected).

Query – (optional) the current query for a query-type object.

- Access – not used.
- Identity – not used.

Type – (required) a type identifier of the contained object.

- Access – the type of the object.
- Identity – the type of the object. You can set up this information for the service layer via audit policies if you set the **Audit Type** property.

9.4.1. About the What – Extension Section

This section can occur multiple times. It contains additional identifying information in type-value pairs. The attributes of this section are:

Type – (required) a type of the extension.

Value – (required) a value of the extension.

- Access – not used.
- Identity – You can set up this information for the service layer via audit policies. It is recommended to follow personal data protection rules and use pseudonyms. Examples: `cn=D638471, employeeNumber=1058347296`.

9.4.2. About the What – Detail Section

The section can occur multiple times. It contains specific details about what was accessed on the object. Operation, type and value show exactly what was changed on the object.

- Access – all detail information.
- Identity – all detail information.

The attributes of this section are:

Operation – (optional) the operation applied when the object was changed.

Type – (required) the type on what the change was applied.

Value – (optional) the value applied when the object was changed.

9.5. About the Database Schema

The DirX Audit message logical schema is mapped to the database schema. The following table describes how message attributes are mapped to tables and their columns in the database schema.

Section **Attribute**	**Table** **Column**
Identification - When	DAT_AUDITMESSAGES IDENTIFICATION_WHEN
Identification - Operation	DAT_AUDITMESSAGES IDENTIFICATION_OP
Identification - UID	DAT_AUDITMESSAGES IDENTIFICATION_UID
Identification - Cause	DAT_AUDITMESSAGES IDENTIFICATION_CAUSE
Identification - Outcome	DAT_AUDITMESSAGES IDENTIFICATION_OUTCOME
Identification - Sensitivity	DAT_AUDITMESSAGES IDENTIFICATION_SENSITIVITY
Identification - Type	DAT_AUDITMESSAGES IDENTIFICATION_TYPE
Identification - Source	DAT_AUDITMESSAGES IDENTIFICATION_SOURCE
Identification - Category	DAT_AUDITMESSAGES IDENTIFICATION_CATEGORY
Identification - Extension - Type	DAT_IDENTIFICATION_EXTENSIONS TYPE
Identification - Extension - Value	DAT_IDENTIFICATION_EXTENSIONS VAL
Where From - Application	DAT_AUDITMESSAGES WHEREFROM_APPLICATION
Where From - Address	DAT_AUDITMESSAGES WHEREFROM_ADDRESS

Section Attribute	Table Column
Where From - Type	DAT_AUDITMESSAGE_ADDITIONS WHEREFROM_TYPE
Where From - Extension - Type	DAT_WHEREFROM_EXTENSIONS TYPE
Where From - Extension - Value	DAT_WHEREFROM_EXTENSIONS VAL
Who - Name	DAT_AUDITMESSAGES WHO_NAME
Who - UID	DAT_AUDITMESSAGES WHO_UID
Who - DN	DAT_AUDITMESSAGE_ADDITIONS WHO_DN
Who - From Address	DAT_AUDITMESSAGE_ADDITIONS WHO_FROMADDRESS
Who - From Type	DAT_AUDITMESSAGE_ADDITIONS WHO_FROMTYPE
Who - Role	DAT_AUDITMESSAGE_ADDITIONS WHO_ROLE
Who - Extension - Type	DAT_WHO_EXTENSIONS TYPE
Who - Extension - Value	DAT_WHO_EXTENSIONS VAL
What - Name	DAT_WHATS WHAT_NAME
What - UID	DAT_WHATS WHAT_UID
What - DN	DAT_WHATS WHAT_DN
What - Sensitivity	DAT_WHATS WHAT_SENSITIVITY
What - Lifecycle	DAT_WHATS WHAT_LIFECYCLE
What - Query	DAT_WHATS WHAT_QUERY
What - Type	DAT_WHATS WHAT_TYPE
What - Extension - Type	DAT_WHAT_EXTENSIONS TYPE

Section Attribute	Table Column
What - Extension - Value	DAT_WHAT_EXTENSIONS VAL
What - Detail - Operation	DAT_WHAT_DETAILS OP
What - Detail - Type	DAT_WHAT_DETAILS TYPE
What - Detail - Value	DAT_WHAT_DETAILS VAL
Original Message	DAT_ORIGINALMESSAGES ORIGINALMESSAGE

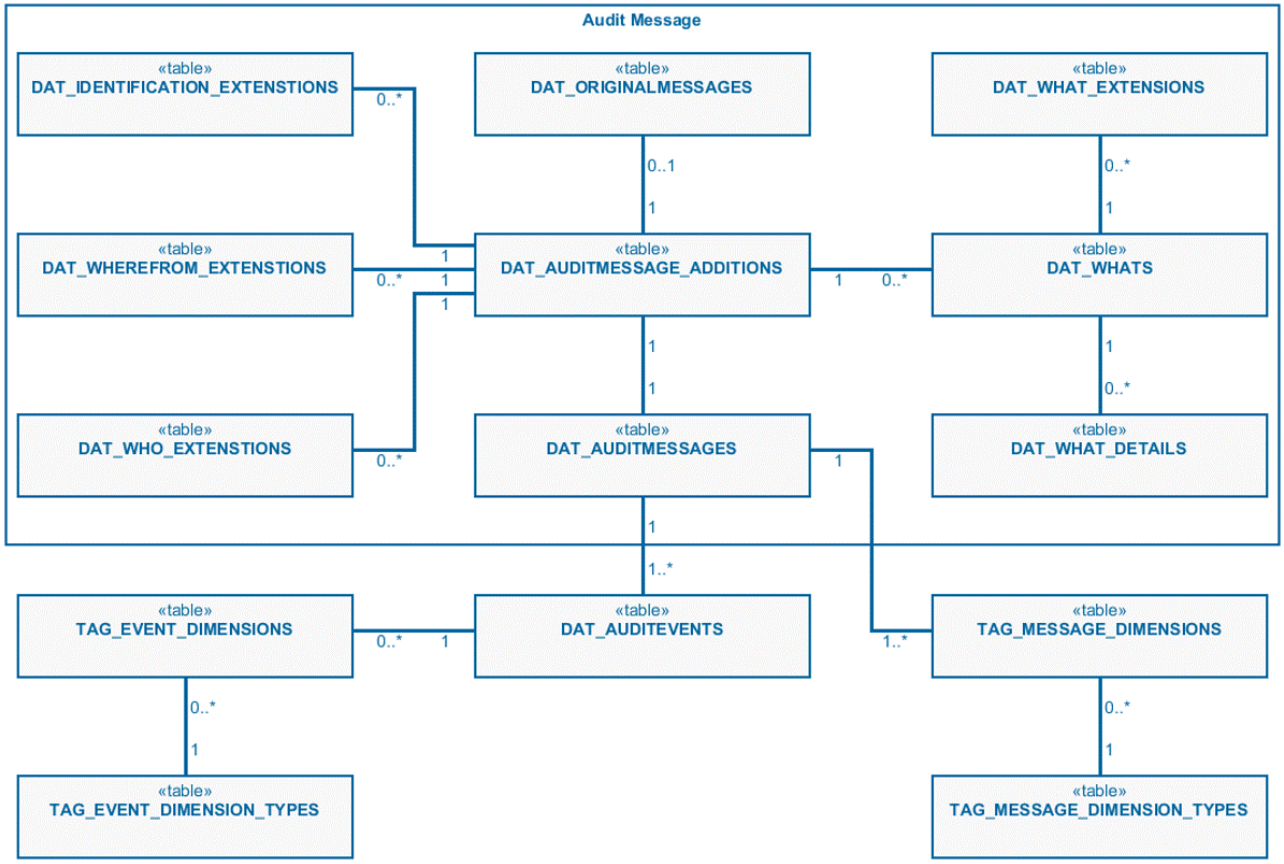

Figure 16. DirX Audit Message Database Schema

9.6. About the Audit Event

The audit event contains data digested from the audit message on the logical operation, type and details. For example, it can provide information that the audit message represents an Add Assignment (Operation) on the User to Role (Type) and identifications of both user and role (Details).

The following table describes how audit event attributes are mapped to the table and its columns in the database schema.

Attribute	Table.Column
Operation	DAT_AUDITEVENTS.OP
Type	DAT_AUDITEVENTS.TYPE
Detail	DAT_AUDITEVENTS.DETAIL

9.7. About the Context Record

The context record contains data on the causing event for most audit events. In particular, it holds names of requesters and approvers in approval activities like Request Add Assignment and Request Add Object.

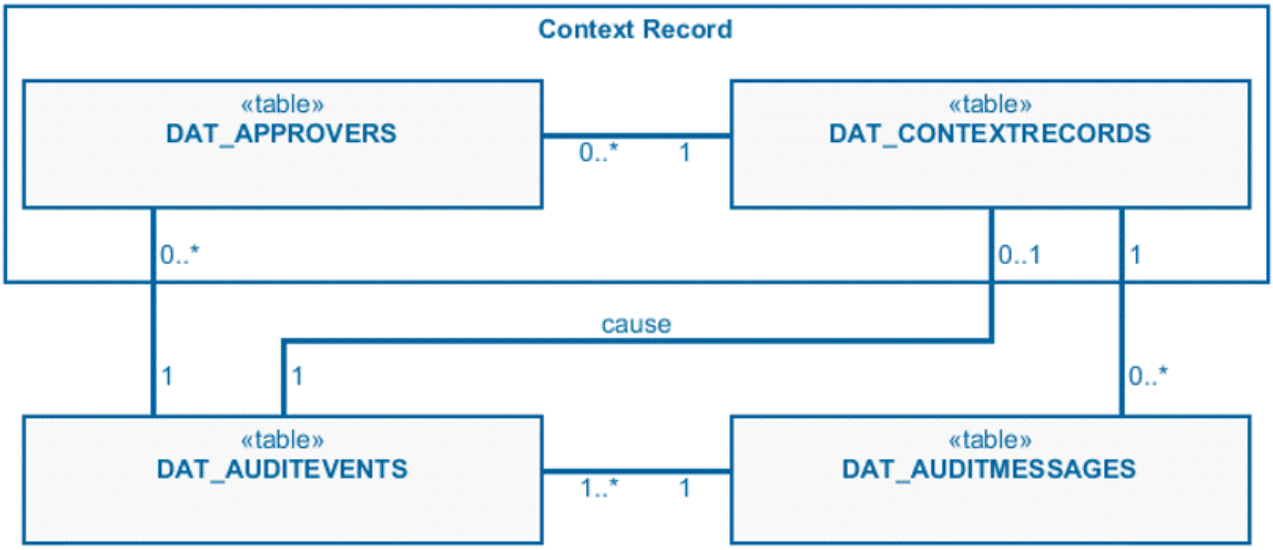

Figure 17. DirX Audit Message Context Record

The following table describes how context record attributes are mapped to tables and their columns in the database schema.

Attribute	Table.Column
Requestor - Name	DAT_CONTEXTRECORDS.REQUESTOR_NAME
Requestor - UID	DAT_CONTEXTRECORDS.REQUESTOR_UID
User - Name	DAT_CONTEXTRECORDS.USER_NAME
User - UID	DAT_CONTEXTRECORDS.USER_UID
Cause - Operation	DAT_CONTEXTRECORDS.CAUSE_OP
Cause - Type	DAT_CONTEXTRECORDS.CAUSE_TYPE
Cause - Detail	DAT_CONTEXTRECORDS.CAUSE_DETAIL
Cause - Rules	DAT_CONTEXTRECORDS.CAUSE_RULES
Approver - Name	DAT_APPROVERS.APPROVER_NAME

10. Authorization

DirX Audit provides authorization on several levels. It supports several application roles and it also allows configuring an authorization Policy Enforcement Point (PEP) to intersect reading of audit messages.

This chapter provides information on how to manage authorization. It describes how to:

- Manage application roles
- Manage authorization policy enforcement points (PEPs)

10.1. Managing Application Roles

DirX Audit supports the following application roles in DirX Audit Manager and REST services:

- **Audit administrator** – this role has full rights for an assigned tenant and full access to all DirX Audit Manager features. Audit administrators can especially create public views that are visible for every common auditor.
- **Auditor** – this role is a common tenant auditor with access to all DirX Audit Manager features. An Auditor can use public or product views and create private versions from these views that can be further modified. Private views and reports are only visible for the auditor who created them.
- **Restricted auditor** – this auditor role has restricted access rights. A restricted auditor can only run and schedule some reports, but cannot use Audit Analysis and History views in DirX Audit Manager. In the Report view, a restricted auditor can see only reports that are tagged as *Restricted*. These reports typically require a parameter such as the organization or organizational unit. It is taken from the attributes of the auditor, which are retrieved during authentication from the associated LDAP directory or from the ID token provided by the OIDC provider (depending on the authentication method defined in the configuration for the tenant). The value of this parameter is used to further restrict the result of the SQL query, so that the report shows only audit events or history entries related to the auditor's organization or organizational unit. Such restricted reports will fail with an error if those required parameters are not provided by the authentication process.

DirX Audit supports the following application roles in DirX Audit Manager Classic:

- **Audit administrator** – this role has full rights for an assigned tenant and can especially create public Audit Analysis filters and Dashboard components that are visible for every common auditor.
- **Auditor** – this role is a common tenant auditor with access to all features of DirX Audit Manager Classic. An Auditor can use public Dashboard components and filters and create private versions from these public items that can be further modified. Private components and reports are only visible for the auditor who created them.
- **Restricted auditor** – this auditor role has restricted access rights. A restricted auditor can only run and schedule some reports, but cannot use Dashboard, Audit Analysis, and

History views in DirX Audit Manager Classic. In the Report view, a restricted auditor can see only reports that are tagged as *Restricted*. These reports typically require a parameter such as the organization or organizational unit. It is taken from the attributes of the auditor, which are retrieved during authentication from the associated LDAP directory. The value of this parameter is used to further restrict the result of the SQL query, so that the report shows only audit events or history entries related to the auditor's organization or organizational unit.

For details on report definitions, see the chapter "Customizing Reports" in the *DirX Audit Customization Guide*. Especially note that the tag *Restricted* corresponds to the key **report.tag.restricted**. For example, see the report definition **EvnImportedUsersApplication_restricted** and its SQL query defined with the **EvnImportedUsersApplication** report.

10.2. Managing Authorization PEPs

Authorization PEPs are responsible for user authorization to read audit events stored in the DirX Audit Database.

Authorization is applied only in Audit analysis view in Audit Manager Classic and in legacy events report and context events reports configured in Audit Manager Classic. All other views in Audit Manager Classic, Audit Manager and REST API do not apply authorization described in this chapter.

The following authorization PEP implementations are currently provided:

- **Empty PEP** – no authorization is enforced. All users can access all audit events.
- **DirX Access PEP** – PEP connected to DirX Access Server and based on policies configured there.

It can be configured with the Tenant Configuration Wizard. See "Authorization Configuration" in "Using the Configuration Wizard for the Tenant Configuration" in "Configuring DirX Audit" in the *DirX Audit Installation Guide* for details on configuring Empty PEP and DirX Access PEP.

10.2.1. PEP Processing

The DirX Access PEP assumes a policy decision point (PDP), which implements XACML 2.0 requests and responses (urn:oasis:names:tc:xacml:2.0:context:schema:os).

Before a query request is performed, the PEP sends a XACML request to the PDP. The request contains all known subject attributes including the group memberships, the resource and the action.

If the response contains a decision "**deny**", a security exception is returned, which causes DirX Audit not to perform any query.

If the response contains a decision "**permit**", the PEP investigates the response for obligations. If there is no obligation, DirX Audit can perform the query unchanged.

If the XACML response contains one or more obligation elements, they are considered to represent query constraints. Query constraints are expected to be the values of the XML sub-elements <AttributeAssignment> with AttributeId="urn:com:siemens:dxt:xacml:extendQuery:and". The PEP extends the query with these obligations by adding each query constraint as an additional condition in the "AND" conjunction to the original query.

DirX Audit then performs the changed query and presents the result without any further involvement of the PEP.

For further details on the XACML request, the policies and the obligations, see the following sections.

10.2.2. XACML Context Request

The PEP sends a XACML 2.0 context request according the schema urn:oasis:names:tc:xacml:2.0:context:schema:os. It contains a <Subject>, a <Resource> and an <Action> element.

The <Subject> contains a list of <Attribute> elements containing the known attributes of the authenticated user. The PEP uses the following format for the AttributeIds:

- urn:oasis:names:tc:xacml:1.0:subject:subject-id: contains the distinguished name (DN) of the user.
- urn:oasis:names:tc:xacml:2.0:subject:role: contains the group DNs of which the LDAP user is a member.
- urn:com:dirxcloud:audit:xacml:application-role: contains the DirX Audit application role (for example, Auditor or Audit admininstrator).
- All other LDAP AttributeIDs start with the prefix "urn:siemens:com:dirxaudit:ldap:" followed by the LDAP attribute name. The following example denotes the "**ou**" attribute: urn:siemens:com:dirxaudit:ldap:ou.

The <Resource> element simply contains an <Attribute> element with AttributeId=” urn:oasis:names:tc:xacml:1.0:resource:resource-id”
and the database name as content value of the element <AttributeValue>.

The <Action> element contains an <Attribute> element with AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
and "**query**" as content value of the element <AttributeValue>.

Note that all attribute values are considered to be strings.

For a sample of a context request, see the **Additions/Data/SamplePolicies** folder on the product installation media.

10.2.3. Access Policies with Obligations

The access policies used for the DirX Access PEP are supposed to adhere to the XACML 2.0 schema "urn:oasis:names:tc:xacml:2.0:policy:schema:os". When deployed with DirX Access, you can edit the policies using the DirX Access Manager by navigating to Policies → ABAC policies.

For policy sample files, see the **Additions/Data/SamplePolicies** folder on the product installation media.

The <Obligations> sub-element of an access policy is particularly important. It is optional and contains several <Obligation> sub-elements. The DirX Audit PEPs evaluate them only on "**permit**" decisions.

In this case, they assume sub-elements <AttributeAssignment> with an AttributeId="urn:com:siemens:dxt:xacml:extendQuery:and"
and the content value as the query constraint. It is in XML format and adheres to the schema
"http://dxt.siemens.com/manager/configuration" and is the same format that DirX Audit Manager uses internally to represent its queries. Since this query constraint is the content of an XML element, the special XML markup characters must be escaped.

11. Managing Fact and Dimension Tables

This chapter describes how to create and maintain the OLAP tables with facts and dimensions, which are the sources for all the Dashboard components and also some reports in DirX Audit Manager and DirX Audit Manager Classic. DirX Audit provides two mechanisms for this purpose:

- A fact population job that the DirX Audit Server runs periodically
- A fact population tool that can be run on demand

The next sections describe these elements and also describe the fact population configuration files on which these elements rely.

11.1. Managing the Scheduled Fact Population Job

The DirX Audit Server can execute scheduled jobs, including a job for the regular population of OLAP cubes. This job obtains the fact and dimension tables from configuration files, creates tables if necessary and then updates them.

The **Fact population** job is part of DirX Audit Server. The job typically runs once per day and updates the facts for the previous several days. The **Fact population** job is dependent on **History DB Update** job, which populates History DB tables necessary for calculating correct values. Therefore, it is recommended to schedule **History DB Update** job before the **Fact population** job. See the section "Defining the History DB Update Job" in the chapter "Managing History Database Tables" of this guide for details.

In order to save time, the **Fact population** job typically should not calculate the facts for the entire database. Therefore, it supports configuration options that tell it how far into the past it should go. Based on this configuration, the **Fact population** job searches for audit messages and history entries created in the last several days, updates the dimension tables for newly created dimension types and values, calculates the facts and dimensions and overrides the appropriate rows of all fact tables.

11.1.1. Defining the Schedule

If you want to change the fact population schedule, you only need to start the Tenant Configuration Wizard for the respective tenant and choose the "Scheduled Jobs Configuration". Finishing the Tenant Configuration Wizard modifies the respective tenant route XML definition file. If the DirX Audit Server for the tenant is running, the job is rescheduled immediately. For more details, see the section "Scheduled Jobs Configuration" in the *DirX Audit Installation Guide*.

It is also possible to modify the route manually in the file located at *install_path*/**server_container/tenants/***tenantID*/**deploy/routes/route-dxt-scheduler-factpopulation.xml**.

Note that when manual modifications are made to this file, it must be copied to some other directory outside of the tenant route deployment folder, deleted from the tenant route deployment folder, and finally copied back into the tenant route deployment folder. If the file is not deleted from the folder and then copied back in, the modifications will not be reflected in the job scheduling.

The tenant route XML file contains several processing directives, but the most important part is the route element: **route-dxt-scheduler-factpopulation**.

To create a scheduled job, you need to specify the URI of a component that is able to process it. DirX Audit Server contains such a component – an Apache Camel endpoint – that accepts a new schedule at **dxt-scheduler://groupName/jobName?jobParameters**. In this way, you can create a new scheduled job.

11.1.1.1. Time Factors

It is important to consider the effects of daylight savings time on job's operation. For example, in the Central European time zone, you go from 1:59:59.999 to suddenly being 3:00:00.000 for spring forward and from 2:59:59.999 to being 2:00:00.000 for autumn back. If you schedule your job between 2:00:00.000 a.m. and 2:59:59.999 a.m. inclusive, it will not run on the daylight-saving date in spring and run twice on the daylight-saving date in autumn. Therefor it is highly recommended to schedule your job out of this range. An example below schedules the job to run daily at 3:01:00 a.m.

As some dimension and fact tables are dependent on fresh data calculated during **History DB Update** job, it is recommended to schedule **Fact population** job after **History DB Update** job.

11.1.1.2. Defining a Trigger

To start a job in DirX Audit at certain time, the Quartz Scheduler, which is a Java variant of the Linux **cron** tool, is used. The **cron** expression string defines the regular schedule in terms of seconds, minutes, hours, day-of-month, month, day-of-week and year (optional). The format should be well-known from the UNIX tool. For a detailed tutorial on **cron** expressions, see the following Quartz tutorial: http://www.quartz-scheduler.org/documentation/2.3.1-SNAPSHOT/tutorials/tutorial-lesson-06.html.

For example, to run fact population daily at 4:01 AM, the following expression is sufficient:

```
0 1 4 * * ?
```

However, because the **cron** parameter is a part of the URI and it does not allow spaces, it is necessary to rewrite the **cron** expression like this:

```
cron=0+1+4+*+*+?&amp;
```

Note the "& amp;" part of trigger. It is a URI parameter separator "&" encoded in UTF-8 and is necessary if there are other parameters that follow the cron parameter in the URI.

11.1.1.3. Defining a Job

A typical job definition for scheduled XML-based fact population uses the “relative date” job class:

```
…
<route
 id="route-dxt-scheduler-factpopulation">
 <from
  uri="dxt-scheduler://DXT-6bc24196-327d-441c-8d66-3633ee6b887b-
jobs/PopulateFactTablesRelDatesJob?cron=0+0+5+?+*+*&amp;jobClass=solu
tions.dirx.audit.common.jobs.quartzjobs.PopulateFactTablesRelDatesJob
&amp;jobdatamap.tenant_id=6bc24196-327d-441c-8d66-
3633ee6b887b&amp;jobdatamap.arg_relative_date_from=RAW(TD-
5)&amp;jobdatamap.arg_relative_date_to=RAW(TD)&amp;jobdatamap.arg_num
ber_of_days_iteration=14" />
 <log
  loggingLevel="INFO"

logName="solutions.dirx.audit.server.routes.scheduler.factpopulation.
Route"
  message="Job 6bc24196-327d-441c-8d66-3633ee6b887b -
PopulateFactTablesRelDatesJob has finished." />
</route>
…
```

As already mentioned, the main part is the route element, more specifically, the **from** element and its attribute URI. There are two main parameters that are mandatory for all scheduled jobs. The **cron** parameter specified according to the rules described here and the **jobClass** parameter that represents the Java class handling job. Make sure that the **jobClass** is correct. It needs to be an existing class of the DirX Audit Server’s Fact Population binding component. The following parameters are also important:

- **jobdatamap.tenant_id** – defines the tenant identifier.
- **jobdatamap.arg_relative_date_from** – defines the start date for the XML-based fact population and must conform to the relative date expression described in “Using Relative Date Expression”.
- **jobdatamap.arg_relative_date_to** – defines the end date for the XML-based fact population and must conform to the relative date expression described in “Using Relative Date Expression”.
- **param_VALIDFROM** – defines the start date for the script-based fact population.

Additional parameters include:

- **jobdatamap.arg_timezone_id**
- **jobdatamap.arg_locale_country**
- **jobdatamap.arg_locale_language**
- **jobdatamap.arg_number_of_days_iteration**

The additional parameters are all optional and define the time zone, locale and the number of days per iteration. They are used for setting dates and iterating filling scripts. By default, the Java Virtual Machine default settings are used.

We recommend performing changes – either manually or by running the Tenant Configuration Wizard – with a running instance of the DirX Audit Server service for a specific tenant. If you want to change the data time scope specified by time parameters, you must modify the parameters manually. After you are finished with the (re)configuration of a job, you need to save the tenant route XML file in a different directory, delete it from the tenant route deployment directory, and then copy the saved file back to the tenant deployment directory. Modifying the file and saving the changes is not sufficient to load the changes; the file itself must be deleted from the route deployment folder and then copied back in for the changes to be triggered on the server.

11.1.1.4. Using Relative Date Expression

The following string defines the regular expression for the relative date:

```
^(TD|TW|TM|TY)(+|-[1-9][0-9]*)?$
```

The enumeration **TD**, **TW**, **TM** and **TY** stand for:

- **TD**: this day / today – the current day.
- **TW**: this week.
- **TM**: this month.
- **TY**: this year.

The optional suffixes *-n* specify a shift into the past relative to the **T*** expression.

Examples:

TD: For from-date: beginning of the current day.
TD: For to-date: end of the current day.

TD-1: For from-date: beginning of yesterday.
TD-1: For to-date: end of yesterday.

TW-1: For from-date: beginning of the last week. Depending on the locale this might be Sunday or Monday.
TW-1: For to date: end of the last week (Saturday or Sunday).

11.2. Running the Fact Population Tool

There are cases where scheduled fact population is not enough; for example:

- When you have exported old audit messages to files, re-imported them and want to have facts for them.

DirX Audit provides the **db_fact_population** command line tool to address these use cases. The tool accepts a start and an end date and calculates the facts for this time range. See the chapter "Using the DirX Audit Tools" in the *DirX Audit Command Line Interface Guide* for usage information on this tool.

Fact and dimension tables are created when the population runs for the first time. DirX Audit Manager can issue alerts about non-existing fact and dimension tables. To create the tables immediately after DirX Audit installation, run the fact population tool. For more details, see the chapter "Using the DirX Audit Tools" in the *DirX Audit Command Line Interface Guide*.

11.3. Fact Population Configuration Files

Both the scheduled and on-demand fact population services use the following fact and dimension table configuration files, which are all contained in the *install_path*/**conf/tenants/***tenantID*/**fact-configuration** folder when the tenant-specific customization is defined. In the *install_path*/**conf/fact-configuration** folder when the core customization is defined. And lastly, in the *install_path*/**conf/defaults/fact-configuration** folder where the default definition is available:

- **confFactTables.xml** – fact tables configuration file.
- **confFacts.xml** – facts configuration file.
- **confDimensions.xml** – dimensions configuration file.

The Data DB and History DB (if licensed) must be set for the same DB server type.

See the section "Customizing Fact and Dimension Tables" in the *DirX Audit Customization Guide* for details on the structure of these files and how to add and disable fact tables, facts or dimensions. The tenant-specific configuration is described there as well.

Note that both the scheduled and on-demand fact population services create the fact and dimension tables according to the configuration in these files when necessary. They also add fact and dimension columns to existing tables when missing, but they don't delete existing columns from tables when they are disabled or removed from the configuration.

12. History Database Synchronization

The new DirX Audit feature for importing entries from DirX Identity domain into the DirX Audit History Database is used starting with DirX Audit version 7.2. The DirX Audit History synchronization jobs are used instead of the synchronization workflows running on a DirX Identity Java-based Server (IdS-J) used in previous versions.

For instructions on how to configure and run DirX Audit History synchronization jobs, see the *DirX Audit History Synchronization Guide*.

13. Managing History Database Tables

This chapter describes how to configure the scheduled History Database (History DB) update job, which computes additional information from stored history values. Its main function is to calculate foreign keys from the stored DN values in imported history entries to obtain relational database references to History DB tables. In addition, it performs some other tasks and can also be extended.

The History DB update job runs as a scheduled job in the DirX Audit Server and is enabled automatically when the license for the History DB feature is valid and DirX Audit Server is installed and properly configured. Each configured tenant has its own scheduled job and a separate instance of DirX Audit Server is used for every tenant starting with version 7.2.

If you want to change the History DB update job schedule, you only need to start the Tenant Configuration Wizard for the respective tenant and choose the "Scheduled Jobs Configuration".
After finishing the configuration, if the DirX Audit Server is running, the job is rescheduled immediately.
For more details, see the section "Scheduled Jobs Configuration" in the *DirX Audit Installation Guide*.

The job scheduling and details for the History DB update job can also be configured manually in the file *install_path*/**server_container/tenants/***tenantID*/**deploy/routes/route-dxt-scheduler-updatehistdb.xml**. This file has the same structure and syntax as the scheduled fact population service definition file. See the section "Defining the Schedule" in the chapter "Managing Fact and Dimension Tables" of this guide for details. After modifying the file, copy it to a different directory outside of the tenant route deployment folder, delete the file from the tenant route deployment folder, and then copy the saved file back into the tenant route deployment folder. This procedure ensures that the modifications are registered with the server and reflected in the job scheduling.

The next sections describe how to define the job and how to express relative dates in the SQL script files.

13.1. Defining the History DB Update Job

You must define the job and (optionally) provide the SQL script files that contain the additional statements to execute. There are several update tasks hard-coded in the DirX Audit Server for updating the DB. The configurable SQL files are used to extend and customize the History DB update job. The SQL files are executed prior to the hard-coded actions.

You can limit the number of updated entries by setting the filter for **VALID_FROM** fields of the updated entries. The general limit can be set in the **jobdatamap.arg_update_validfrom_condition** argument. This argument must be a simple condition (included in the where clause). This filter will be used in the hardcoded actions and can be referenced also in the SQL files (using the **%ARG_VALIDFROM_COND%** placeholder). No filter will be used if this argument is not given or is empty (all suitable entries will be updated).

By default, VALID_FROM filter is set to 7 days to the past, meaning relevant entries are updated and fields are calculated. When setting the filter for VALID_FROM, it is recommended to set the filter same as or earlier as VALIDFROM in the Fact population job. Due to Fact population job being dependent on History DB Update job. See the section "Defining the Schedule" in the chapter "Managing Fact and Dimension Tables" of this guide for details.

You can supply multiple SQL script files separated by commas in the **jobdatamap.arg_sql_files** parameter.

Note, that you can add or customize these SQL files for all and/or specific tenant as described in the General Customization chapter. Examples are available in chapter Customizing SQL scripts.

A licensed History Database installation provides configured versions of all the necessary files. You can also find all SQL script files at the following locations on the installation media:

- *install_media*/**Additions/Scripts/common/updatehistdb/insert_DIM_HST_GEN_DATETIME.dxtreldatesql.txt**
- *install_media*/**Additions/Scripts/common/updatehistdb/delete_HDB_ASSIGNMENT_CERTIFICATIONS_01.dxtreldatesql.txt**
- *install_media*/**Additions/Scripts/common/updatehistdb/insert_HDB_ASSIGNMENT_CERTIFICATIONS_02.dxtreldatesql.txt**
- *install_media*/**Additions/Scripts/common/updatehistdb/merge_HDB_CERTENTRIES_SUCCEEDED_01.dxtreldatesql.txt**
- *install_media*/**Additions/Scripts/common/updatehistdb/merge_HDB_MANUAL_ASSIGNMENTS_01.dxtreldatesql.txt**
- *install_media*/**Additions/Scripts/common/updatehistdb/delete_HDB_ASS_CERTIFICATIONS_12_01.dxtreldatesql.txt**
- *install_media*/**Additions/Scripts/common/updatehistdb/insert_HDB_ASS_CERTIFICATIONS_12_02.dxtreldatesql.txt**
- *install_media*/**Additions/Scripts/common/updatehistdb/delete_HDB_CERTCAMPAIGNS_01.dxtreldatesql.txt**
- *install_media*/**Additions/Scripts/common/updatehistdb/insert_HDB_CERTCAMPAIGNS_02.dxtreldatesql.txt**
- *install_media*/**Additions/Scripts/common/updatehistdb/delete_HDB_RSK_USR_CERTIFICATIONS_03_01.dxtreldatesql.txt**
- *install_media*/**Additions/Scripts/common/updatehistdb/insert_HDB_RSK_USR_CERTIFICATIONS_03_02.dxtreldatesql.txt**
- *install_media*/**Additions/Scripts/common/updatehistdb/delete_HDB_USR_CERTIFICATIONS_12_01.dxtreldatesql.txt**
- *install_media*/**Additions/Scripts/common/updatehistdb/insert_HDB_USR_CERTIFICATIONS_12_02.dxtreldatesql.txt**
- *install_media*/**Additions/Scripts/common/updatehistdb/delete_HDB_APPROVALS_01.dxtreldatesql.txt**

- *install_media*/**Additions/Scripts/common/updatehistdb/insert_HDB_APPROVALS_02.dxtreldatesql.txt**
- *install_media*/**Additions/Scripts/common/updatehistdb/update_HDB_IMPORTED_MEMBERSHIPS_01.dxtreldatesql.txt**
- *install_media*/**Additions/Scripts/common/updatehistdb/merge_HDB_IMPORTED_MEMBERSHIPS_02.dxtreldatesql.txt**
- *install_media*/**Additions/Scripts/common/updatehistdb/update_HST_LINK_ATTRS_IN_TIME.dxtreldatesql.txt**
- *install_media*/**Additions/Scripts/common/updatehistdb/update_HST_ROLEPARAMS_IN_TIME_PARAMENTRY.dxtreldatesql.txt**
- *install_media*/**Additions/Scripts/common/updatehistdb/update_HST_ROLEPARAMS_IN_TIME_PARAMVALUE.dxtreldatesql.txt**
- *install_media*/**Additions/Scripts/common/updatehistdb/insert_HDB_REQWF.txt**
- *install_media*/**Additions/Scripts/common/updatehistdb/insert_HDB_REQWF_ACTIVITIES.txt**
- *install_media*/**Additions/Scripts/common/updatehistdb/insert_HDB_REQWF_RESOURCES.txt**

There is one optional parameter **jobdatamap.arg_membership_validfrom**, which is only relevant to the membership calculation task of the History DB update job and represents the time at which these memberships will be recalculated.

Here is an example of a job definition in the file for the History DB update job that uses five SQL script files:

```
…
<route>
 <from
  uri="dxt-scheduler://DXT-71a75691-d28a-48ce-a542-6d6af7ece680-
jobs/UpdateHistoryDbJob?cron=0+0+1+?+*+*&amp;jobClass=com.dirxcloud.a
udit.common.jobs.quartzjobs.UpdateHistoryDbJob&amp;jobdatamap.tenant_
id=71a75691-d28a-48ce-a542-
6d6af7ece680&amp;jobdatamap.arg_sql_param_VALIDFROM=%REL_DATE_FROM(TD
-7)%&ampjobdatamap.arg_sql_files=update_HST_LINK_ATTRS_IN_TIME.dxtrel
datesql.txt,update_HST_ROLEPARAMS_IN_TIME_PARAMENTRY.dxtreldatesql.tx
t,update_HST_ROLEPARAMS_IN_TIME_PARAMVALUE.dxtreldatesql.txt,insert_H
DB_REQWFS.txt,insert_HDB_REQWF_RESOURCES.txt,insert_HDB_REQWF_ACTIVIT
IES.txt&amp;jobdatamap.arg_membership_validfrom=TD-30" />
 <log
  loggingLevel="INFO"

logName="com.dirxcloud.audit.server.routes.scheduler.updatehistdb.Rou
te"
  message="Job 4606b126-b70e-4c16-b271-faf8c4f9674d -
UpdateHistoryDbJob has finished." />
</route>
…
```

13.2. Expressing Relative Dates in SQL Files

The SQL files can use following special placeholders for expressing dates relative to current date and time of the particular execution:

- **%REL_DATE_FROM(***RELDATE_EXPR***)%** – to set the from-date. The date and time are shifted by the given offset and then set to the beginning of the given relative date period; for example, beginning of day, week, month.
- **%REL_DATE_TO(***RELDATE_EXPR***)%** – to set the to-date. The date and time are shifted by the given offset and then set to the end of the given relative date period; for example, end of day, week, month.
- **%ARG_VALIDFROM_COND%** – use the valid-from condition from the quartz definition XML file (see above). Please notice that this argument expands to a condition **including** the operator.

The *RELDATE_EXPR* uses the same syntax as the expressions for fact population. For details, see the section "Defining the Schedule" in the chapter "Managing Fact and Dimension Tables".

Examples of the relative date expressions include:

- **TD** – today
- **TD-1** – yesterday
- **TD-3** – three days ago
- **TW-1** – last week

These placeholders are replaced by the corresponding date-time value in the query as a java.sql.Timestamp value with configured calendar. There can be any number of these placeholders used. Each will be replaced by the correct date-time value during execution.

```
update HST_LINK_ATTRS_IN_TIME
set LNK_ENTRIES_ID = (
 select top 1
  HST_ENTRIES_IN_TIME.HST_ENTRIES_ID
 from
  HST_ENTRIES_IN_TIME
 where
  HST_LINK_ATTRS_IN_TIME.LNK_ENTRIES_ID is null
   and HST_LINK_ATTRS_IN_TIME.ATTRIBUTE_VALUE =
    HST_ENTRIES_IN_TIME.dn
   and HST_LINK_ATTRS_IN_TIME.VALID_TO is null
   and HST_ENTRIES_IN_TIME.VALID_TO is null
 order by
  HST_ENTRIES_IN_TIME.VALID_FROM desc
 )
where
 LNK_ENTRIES_ID is null
  and HST_LINK_ATTRS_IN_TIME.VALID_FROM >= %REL_DATE_FROM(TD-3)%
```

14. Understanding the History Database Schema

This chapter describes the History Database (History DB) schema including OLTP and OLAP database structures. The database schema is designed for extensibility to support customized IAM solutions.

14.1. About the History Database OLTP Schema

A history entry consists of the following data:

- **Entry type** – indicates the type of entry, such as AccessRight, Account, Activity Definition, Activity Instance, Assignment, Audit Policy, Business Object, Certification Assignment Change, Certification Campaign, Certification Entry, Certification Notification, Configuration Object, Delegation, Domain Object, Group, Permission, Policy, Role, Role Parameter, Target System, Target System Configuration Object, Ticket, User, Workflow Definition and Workflow Instance.
- **Entry** – contains identification attributes such as the unique identifiers and the name of the entry.
- **Small attribute values** – contain the attribute values that can be represented with primitive data types like string, number, date and time or boolean.
- **Link attribute values** – contain the attribute values that can be represented as references to other entries. It is usually a DN in a DirX Identity domain. Link attributes also contain user-to-privilege assignments, including assignment attributes and role parameter values.
- **Large attribute values** – contain the attribute values that can be represented with large object data types like images, XML structures, certificates, and so on.

Values are stored together with their date and time validity.

The following figure illustrates the History Database OLTP schema:

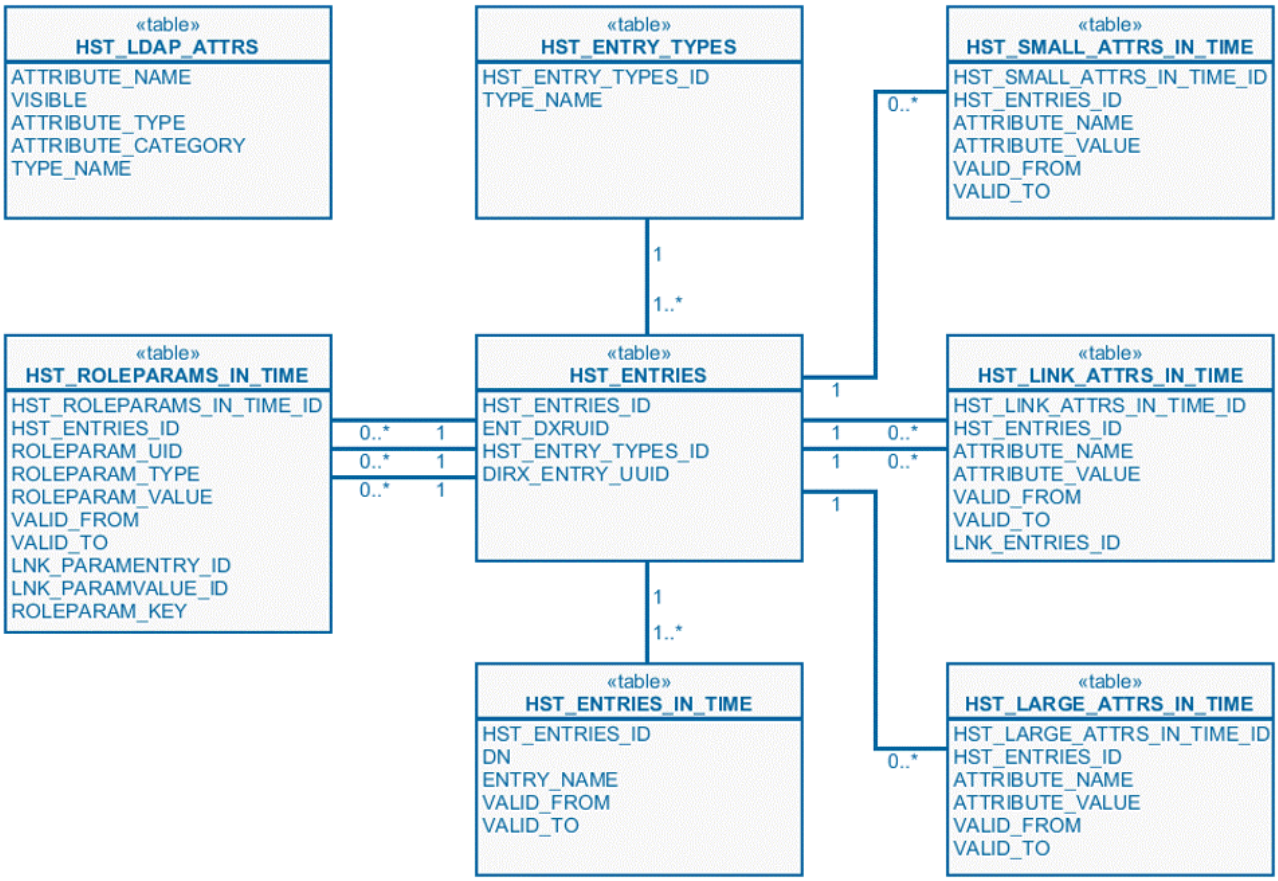

Figure 18. History Database OLTP Schema

The following table describes the table columns in the History Database OLTP schema.

Table Column	Description
HST_ENTRY_TYPES TYPE_NAME	Name of the entry type. The entry type is unique for the whole entry life cycle.
HST_ENTRIES ENT_DXRUID	UID of the entry. The UID is unique for the whole entry life cycle.
HST_ENTRIES DIRX_ENTRY_UUID	DirX Directory entry UID. The UID is calculated by DirX Server and is unique for the whole entry life cycle.
HST_ENTRIES_IN_TIME DN	DN of the entry.
HST_ENTRIES_IN_TIME ENTRY_NAME	Name of the entry.
HST_LDAP_ATTRS ATTRIBUTE_NAME	Name of the attribute.
HST_LDAP_ATTRS VISIBLE	Visibility of the attribute.
HST_LDAP_ATTRS ATTRIBUTE_TYPE	Data type of the attribute.

Table Column	Description
HST_LDAP_ATTRS ATTRIBUTE_CATEGORY	Category of the attribute: SMALL, LARGE, LINK, ROLEPARAM, EXCLUDED, ENTRY.
HST_LDAP_ATTRS TYPE_NAME	Indicates whether the LDAP attribute synchronization is restricted only to a specific entry type.
HST_SMALL_ATTRS_IN_TIME ATTRIBUTE_NAME	Small attribute name.
HST_SMALL_ATTRS_IN_TIME ATTRIBUTE_VALUE	Small attribute value.
HST_LARGE_ATTRS_IN_TIME ATTRIBUTE_NAME	Large attribute name.
HST_LARGE_ATTRS_IN_TIME ATTRIBUTE_VALUE	Large attribute value.
HST_LINK_ATTRS_IN_TIME ATTRIBUTE_NAME	Link attribute name.
HST_LINK_ATTRS_IN_TIME ATTRIBUTE_VALUE	Link attribute value.
HST_ROLEPARAMS_IN_TIME ROLEPARAM_UID	UID of the role parameter.
HST_ROLEPARAMS_IN_TIME ROLEPARAM_TYPE	Type of the role parameter.
HST_ROLEPARAMS_IN_TIME ROLEPARAM_VALUE	Value of the role parameter.
HST_ROLEPARAMS_IN_TIME ROLEPARAM_KEY	Key of the role parameter.
* VALID_FROM	Start of the value validity.
* VALID_TO	End of the value validity.

14.2. About the History Database OLAP Schema

The History Database OLAP schema is designed to enhance the performance of queries on history entries. Materialized or indexed views support relationships between all types of entries and their attributes and also of users and their roles, permissions, groups and accounts, of roles and their junior roles and permissions and of permissions and their groups.

The views depend on the calculation of foreign keys which are derived from link attribute values. DirX Audit Server runs the **History DB update** scheduled job to update the foreign keys of the History OLTP tables.

The following figure illustrates the views of the History Database OLAP schema:

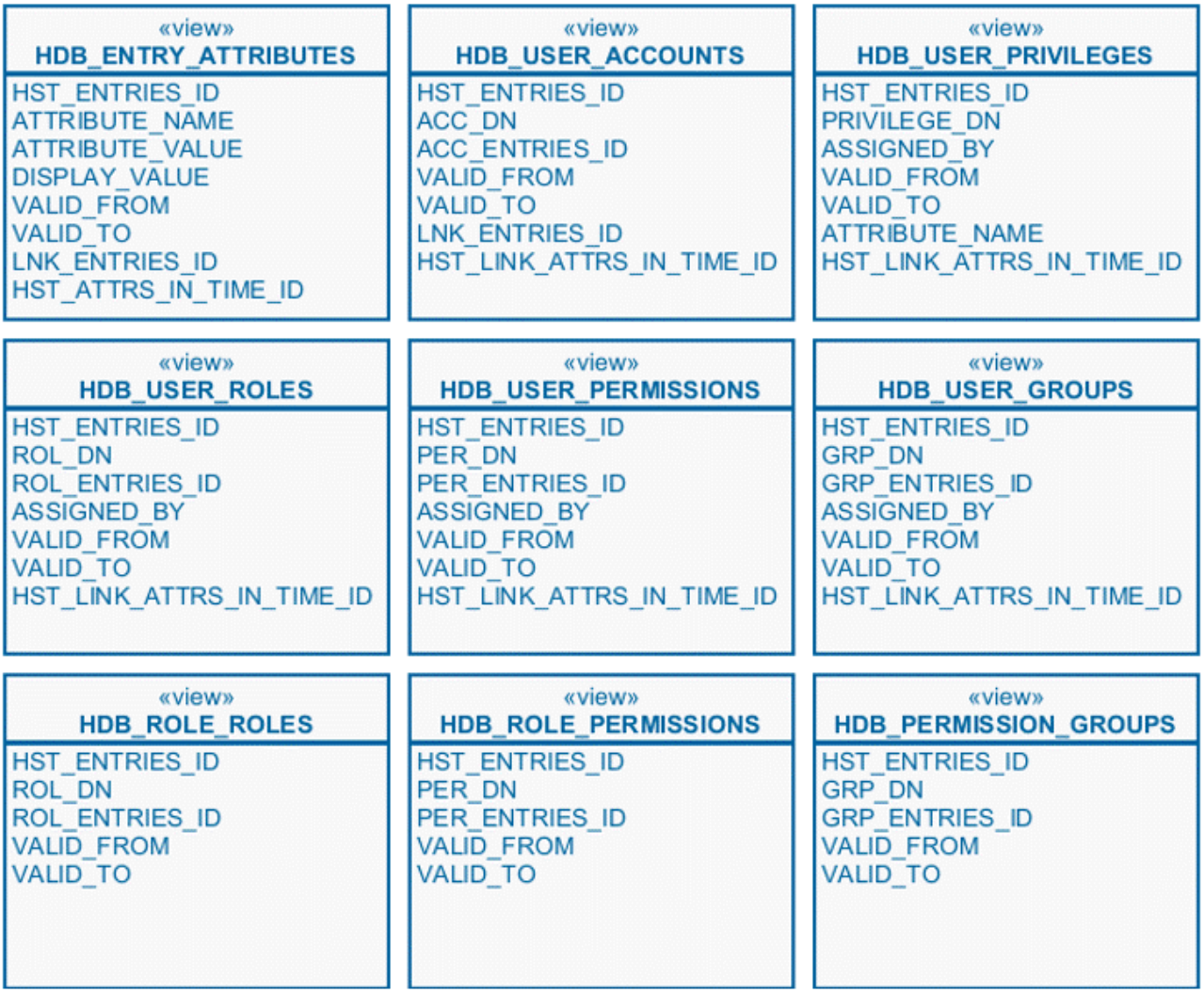

Figure 19. History Database OLAP Schema Views

For some reports, additional History OLAP tables are created and managed. They keep data on request workflows including their subjects, resources and people activities. The content of these OLAP tables is extended with a regular job run by DirX Audit Server.

The following figure illustrates the History Database OLAP schema tables:

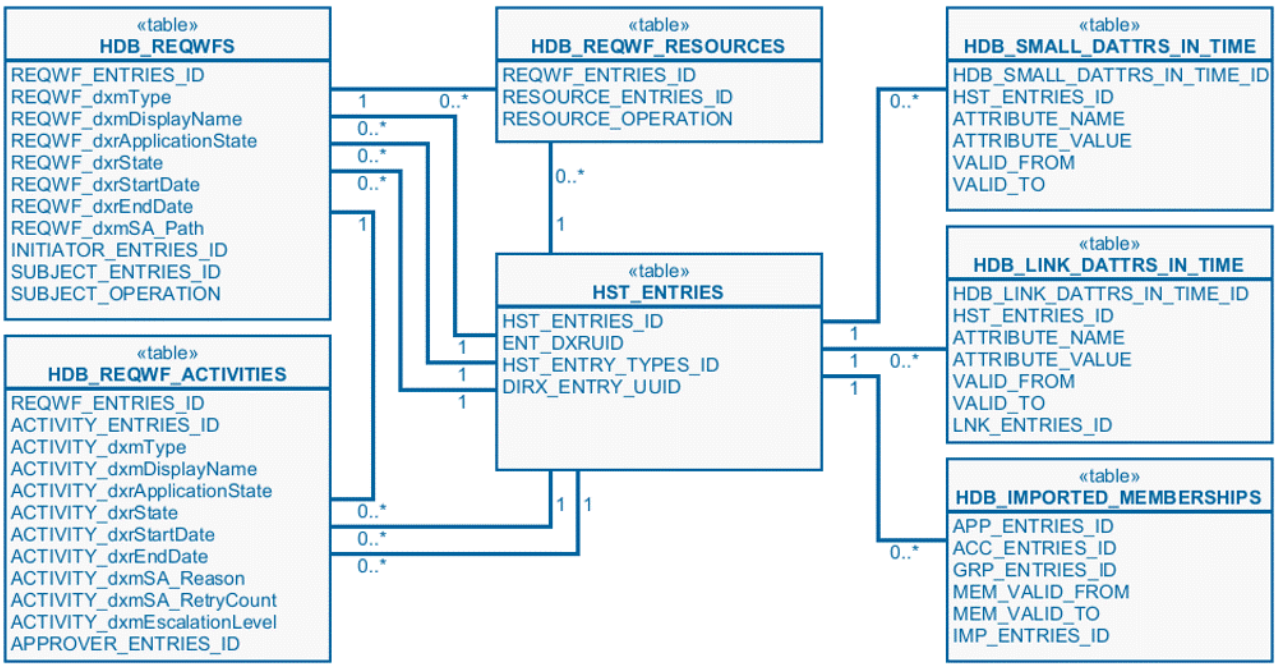

Figure 20. History Database OLAP Schema Tables

The following tables describe views and tables and their columns of the History Database OLAP schema.

Table Column	Description
HDB_ENTRY_ATTRIBUTES ATTRIBUTE_NAME	Attribute name.
HDB_ENTRY_ATTRIBUTES ATTRIBUTE_VALUE	Attribute value.
HDB_USER_ACCOUNTS ACC_DN	Account distinguished name.
HDB_USER_PRIVILEGES PRIVILEGE_DN	Privilege distinguished name.
HDB_USER_PRIVILEGES ASSIGNED_BY	Assignment mode: manual, BO, rule, inherited.
HDB_USER_ROLES ROL_DN	Role distinguished name.
HDB_USER_ROLES ASSIGNED_BY	Assignment mode: manual, BO, rule, inherited.
HDB_USER_PERMISSIONS PER_DN	Permission distinguished name.
HDB_USER_PERMISSIONS ASSIGNED_BY	Assignment mode: manual, BO, rule, inherited.
HDB_USER_GROUPS GRP_DN	Group distinguished name.

Table Column	Description
HDB_USER_GROUPS ASSIGNED_BY	Assignment mode: manual, BO, rule, inherited.
HDB_ROLE_ROLES ROL_DN	Junior role distinguished name.
HDB_ROLE_PERMISSIONS PER_DN	Permission distinguished name.
HDB_PERMISSION_GROUPS GRP_DN	Group distinguished name.

Table Column	Description
HDB_REQWFS REQWF_dxmType	Request workflow type: Request, Certification.
HDB_REQWFS REQWF_dxmDisplayName	Display name of the request workflow.
HDB_REQWFS REQWF_dxrApplicationState	Application state of the request workflow.
HDB_REQWFS REQWF_dxrState	State of the request workflow.
HDB_REQWFS REQWF_dxrStartDate	Start date of the request workflow.
HDB_REQWFS REQWF_dxrEndDate	End date of the request workflow.
HDB_REQWFS REQWF_dxmSA_Path	Path to the request workflow definition.
HDB_REQWFS SUBJECT_OPERATION	Operation over the request workflow subject.
HDB_REQWF_RESOURCES RESOURCE_OPERATION	Operation over the request workflow resource.
HDB_REQWF_ACTIVITIES REQWF_dxmType	Activity type: people.
HDB_REQWF_ACTIVITIES REQWF_dxmDisplayName	Display name of the request workflow activity.
HDB_REQWF_ACTIVITIES REQWF_dxrApplicationState	Application state of the request workflow activity.
HDB_REQWF_ACTIVITIES REQWF_dxrState	State of the request workflow activity.
HDB_REQWF_ACTIVITIES REQWF_dxrStartDate	Start date of the request workflow activity.

Table Column	Description
HDB_REQWF_ACTIVITIES REQWF_dxrEndDate	End date of the request workflow activity.
HDB_REQWF_ACTIVITIES REQWF_dxmSA_Reason	Reason for the request workflow activity resolution.
HDB_REQWF_ACTIVITIES REQWF_dxmSA_RetryCount	Retry count of the request workflow activity.
HDB_REQWF_ACTIVITIES REQWF_dxmEscalationLevel	Escalation level of the request workflow activity.
HDB_SMALL_DATTRS_IN_TIME ATTRIBUTE_NAME	Derived small attribute name.
HDB_SMALL_DATTRS_IN_TIME ATTRIBUTE_VALUE	Derived small attribute value.
HDB_LINK_DATTRS_IN_TIME ATTRIBUTE_NAME	Derived link attribute name.
HDB_LINK_DATTRS_IN_TIME ATTRIBUTE_VALUE	Derived link attribute value.
HDB_IMPORTED_MEMBERSHIPS MEM_VALID_FROM	Start of the membership validity.
HDB_IMPORTED_MEMBERSHIPS MEM_VALID_TO	End of the membership validity.

For some Dashboard components and reports on DirX Identity certification campaigns, additional History OLAP tables are created and managed. They keep data on certification campaigns, certification entries and certification assignment changes and their relations to history entries. The content of these OLAP tables is extended with the regular **History DB update** job run by DirX Audit Server.

The following figure illustrates the History Database OLAP schema tables for certification campaigns:

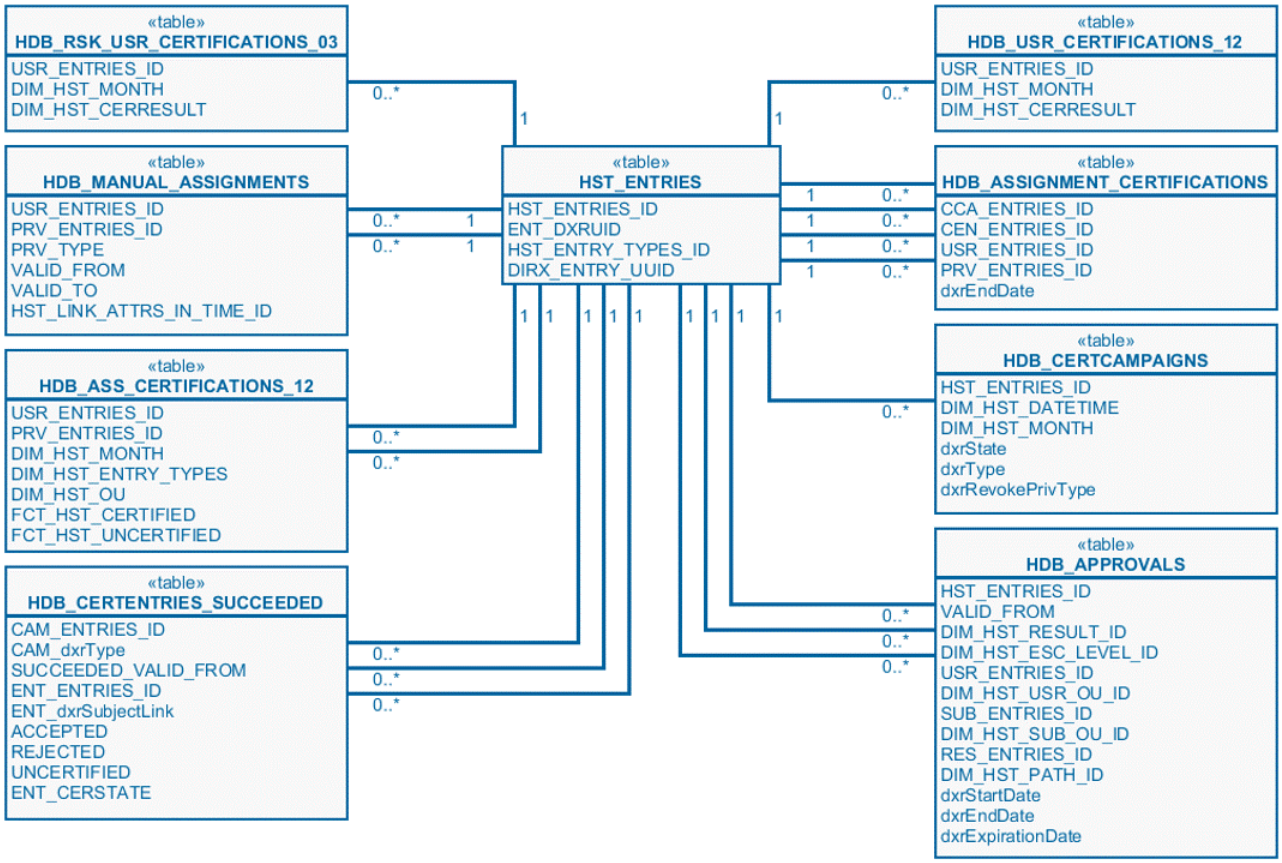

Figure 21. History Database OLAP Schema Tables for Certification Campaigns and Approvals

The following table describes the tables and their columns of the History Database OLAP schema for certification campaigns.

Table Column	Description
HDB_RSK_USR_CERTIFICATIONS_03 DIM_HST_MONTH	Month when a certification is performed.
HDB_RSK_USR_CERTIFICATIONS_03 DIM_HST_CERRESULT	Whether the user is certified fully, partially, planned only or even unplanned.
HDB_MANUAL_ASSIGNMENTS PRV_TYPE	Privilege type: Role, Permission or Group.
HDB_ASS_CERTIFICATIONS_12 DIM_HST_MONTH	Month when an assignment is certified.
HDB_ASS_CERTIFICATIONS_12 DIM_HST_ENTRY_TYPES	Entry type: Role, Permission or Group.
HDB_ASS_CERTIFICATIONS_12 DIM_HST_OU	Organizational unit of a certified user.
HDB_ASS_CERTIFICATIONS_12 FCT_HST_CERTIFIED	Whether the assignment is certified (1) or uncertified (0).
HDB_ASS_CERTIFICATIONS_12 FCT_HST_UNCERTIFIED	Whether the assignment is certified (0) or uncertified (1).

Table Column	Description
HDB_CERTENTRIES_SUCCEEDED CAM_dxrType	Certification campaign type: Privilege certification (RoleToUser) or User certification (UserToRole).
HDB_CERTENTRIES_SUCCEEDED SUCCEEDED_VALID_FROM	When certification succeeded.
HDB_CERTENTRIES_SUCCEEDED ENT_dxrSubjectLink	Reference to the certification entry subject.
HDB_CERTENTRIES_SUCCEEDED ACCEPTED	Number of accepted assignments.
HDB_CERTENTRIES_SUCCEEDED REJECTED	Number of rejected assignments.
HDB_CERTENTRIES_SUCCEEDED UNCERTIFIED	Number of uncertified assignments.
HDB_CERTENTRIES_SUCCEEDED ENT_CERSTATE	Whether the certification entry is certified fully, partially or only planned.
HDB_USR_CERTIFICATIONS_12 DIM_HST_MONTH	Month when a certification is performed.
HDB_USR_CERTIFICATIONS_12 DIM_HST_CERRESULT	Whether the user is certified fully, partially, planned, or unplanned.
HDB_ASSIGNMENT_CERTIFICATIONS dxrEndDate	End date of the assignment certification.
HDB_CERTCAMPAIGNS DIM_HST_DATETIME	Date when a campaign is executed.
HDB_CERTCAMPAIGNS DIM_HST_MONTH	Month when a campaign is executed.
HDB_CERTCAMPAIGNS dxrState	Certification campaign state (SUCCEEDED, RUNNING, PREPARING, FAILED.EXPIRED, FAILED.PREPARE).
HDB_CERTCAMPAIGNS dxrType	Certification campaign type: Privilege certification (RoleToUser) or User certification (UserToRole).
HDB_CERTCAMPAIGNS dxrRevokePrivType	Certification campaign apply changes type: reject, reject-all, review or simulate.
HDB_APPROVALS dxrStartDate	Start date of the approval.
HDB_APPROVALS dxrEndDate	End date of the approval.
HDB_APPROVALS dxrExpirationDate	Expiration date of the approval.

15. Tuning Database Performance

The database performance is highly dependent on the structure of queries you perform from your DirX Audit Manager or DirX Audit Manager Classic and DirX Audit Server and on the audit message types you have collected or the history entries you have synchronized. There is no general recommendation on indexes structure and so specific configuration should be performed individually.

Without any indexing, database performance will most likely be insufficient. Therefore, the DirX Audit installation provides some SQL scripts for index creation. These scripts are in the folder *install_path***/conf/sql/mssql/creation** for SQL Server or *install_path***/conf/sql/oracle/creation** for Oracle Database. These scripts contain the recommended default indexes. You should create additional or different indexes if there are performance problems with executing queries. The Configuration Wizard will create these default indexes if a new empty database is selected. If you use a database that already contains some tables (that is, from a previous version), the indexes are not created or updated automatically. You should follow the instructions in the *DirX Audit Migration Guide* and then check, create or update the indexes manually.

The rest of this chapter provides some useful hints on tuning database performance in your environment.

15.1. Tracing SQL Queries in DirX Audit Manager Classic

If you need to trace performed SQL queries in Audit analysis in DirX Audit Manager Classic, follow these steps:

- Uncomment the following section in the *install_path***\web\audit-manager-classic.war\WEB-INF\classes\log4j2.properties** configuration file:

  ```
  logger.org-hibernate-sql.name = org.hibernate.SQL
  logger.org-hibernate-sql.level = DEBUG
  ```

- Restart the Apache Tomcat service. Now you can search for performed-SQL-query-logged events in the *tomcat_install_path***\logs\dirxaudit-manager** log file. Run the Search operation with your DirX Audit Manager Classic in Audit analysis. The log records look like this:

```
2020-07-01 11:24:50,914 [http-nio-8080-exec-3] DEBUG
org.hibernate.SQL- WITH query AS (select
count(auditevent0_.DAT_AUDITEVENTS_ID) as col_0_0_, ROW_NUMBER() OVER
(ORDER BY CURRENT_TIMESTAMP) as __hibernate_row_nr__ from
DAT_AUDITEVENTS auditevent0_) SELECT * FROM query WHERE
__hibernate_row_nr__ >= ? AND __hibernate_row_nr__ < ?

2020-07-01 11:24:52,789 [http-nio-8080-exec-7] DEBUG
org.hibernate.SQL- WITH query AS (select
auditevent0_.DAT_AUDITEVENTS_ID as DAT1_234_,
auditevent0_.DAT_AUDITMESSAGES_ID as DAT5_234_, auditevent0_.DETAIL
as DETAIL234_, auditevent0_.OP as OP234_, auditevent0_.TYPE as
TYPE234_, ROW_NUMBER() OVER (order by
auditmessa1_.IDENTIFICATION_WHEN DESC) as __hibernate_row_nr__ from
DAT_AUDITEVENTS auditevent0_ inner join DAT_AUDITMESSAGES
auditmessa1_ on
auditevent0_.DAT_AUDITMESSAGES_ID=auditmessa1_.DAT_AUDITMESSAGES_ID )
SELECT * FROM query WHERE __hibernate_row_nr__ >= ? AND
__hibernate_row_nr__ < ?
```

Some queries are available in quasi SQL in the DirX Audit installation folder. You can adapt these queries to SQL and further analyze them.

- Data processing
  *install_path***conf\defaults\sql**
- Report definitions
  *install_path***conf\defaults\report-definitions**

If you are considering optimizing data processing performance, please contact the DirX support unit to get a reference on the SQL queries that relate to the feature you are optimizing.

15.2. Tuning the SQL Server

The SQL Server provides several administration tools that can help you with tuning DirX Audit Database performance:

- SQL Server Management Studio
- SQL Server Profiler
- Database Engine Tuning Advisor

To get advice on statistics and indexes, perform the following steps:

1. Open SQL Server Profiler.
2. Create new trace and log in to your database server. The new trace window is open.
3. Start the selected trace.
4. Open SQL Server Management Studio and log in to your database server.
5. Create a new query based on the statement you caught with the logger. Use your audit messages database for performing the query.
6. Execute the query.
7. When the query is finished, return to the SQL Server Profiler. The performed database activities should be traced. Stop the selected trace and then save the collected data into a trace file.
8. Open the Database Engine Tuning Advisor and then create a new session. Provide the session name, your trace file path as the workload file, the database for analyses and the database to be tuned.
9. Start Analysis.
10. When the analysis is finished, you can view recommendation advised by the tool on the **Recommendations** tab.
11. Get the SQL script for each recommendation you want to apply in the **Definition** field.
12. Run a batch of the SQL statements in the SQL Server Management Studio to apply the recommendations.

15.3. Tuning the Oracle Database

Oracle Database contains several tuning methods accessible in various database tools that can be used to determine the performance problems and perform database tuning. The relevant Oracle Database tools are:

- **Oracle SQL Plus** – the “old fashioned” command line interface that can be used for execution of SQL commands and SQL scripts.
- **Oracle SQL Developer** – a PL/SQL development environment that includes tools for PL/SQL development (editor, compiler, debugger) as well as tools for database administration and tuning (execution plan, autotrace, PL/SQL advisor).
- **Oracle Enterprise Manager** – the Oracle web portal designed for database administration.

For tuning purposes, you need to log in with DBA privileges (for example, user SYSTEM).

Database performance bottlenecks cannot be fully predicted as the performance depends on many factors and exact data distribution.

Response time or high resource needs typically indicate the performance problems.

15.3.1. Detecting High-Load Queries

Typical tuning consists of the following steps:

1. Identifying a high-load query. The Top-SQL page in the Oracle Enterprise Manager contains information about the most resource-consuming SQL statements in the system.
2. Analyzing the query. The main methods here for performance analysis and tuning are SQL execution plans, SQL Advisor reports and application tracing. The next sections provide more information about these methods.
3. Taking corrective action.

15.3.1.1. Generating an Execution Plan

You can obtain an execution plan for any SQL statement even without the execution of the statement. The execution plan depends on database statistics used by the Oracle cost-based optimizer (CBO). The statistics must be up to date and they are typically calculated by a batch process. Statistics can be gathered, calculated, or estimated for object, schema or complete database instances.

An execution plan describes the strategy of the optimizer and often can identify the performance problems, typically full table scans. The execution plan can be obtained from a lot of DB tools; for example, SQL*Plus. Command **set auto-trace on** is the easiest way to turn on execution plan generation. Execution plan is displayed together with optimizer predicates and execution statistics.

The execution plan feature is also available in the Oracle SQL Developer.

15.3.1.2. Using SQL Advisor

SQL Advisor is an Oracle tool that analyzes the given SQL statement and creates a detailed report with recommendations for performance improvements such as query optimization, additional index creation and so on.

SQL Advisor reports can be obtained, for example, from the Oracle SQL Developer.

15.3.1.3. Generating SQL Trace Information

You can generate an application trace for an entire session. The resulting trace file includes a detailed report on SQL statements. Tracing must be enabled or disabled with the **alter session** command. The appropriate trace file is generated on the database server. The **tkprof** utility must be used to format the trace file. The trace file location is stored in the database parameter **dump**.

To obtain the SQL trace information, follow these steps:

In SQL*Plus, perform the following steps:

- Run **`show parameter dump`** – the **`user_dump_dest`** parameter is the location of the trace files.
- Enable tracing with **`alter session set sql_trace=true`**.
- Execute the SQL statement(s).
- Disable tracing with **`alter session set sql_trace=false`**.

  The command sequence could look like this:

```
conn SYS/_password_@_host:port_/orclpdb as SYSDBA;
show parameter dump;
alter session set sql_trace=true;
select _IDENTIFICATION_UID from DXT_DATA.DAT_AUDITMESSAGES_;
alter session set sql_trace=false;
disc;
```

- Run **`disc`** – this action disconnects the current session and closes trace file.

In the database server operating system, perform the following steps:

- Locate the trace and the appropriate ***.trc** file.
- Run the **`tkprof`** utility and create an output text file.
- Check the results.

The information about one statement is organized in a table. The meaning of the sections is as follows:

Parse – translates the SQL statement into an execution plan.

Execute – executes the statement and modifies data (INSERT, UPDATE, DELETE).

Fetch – retrieves the row returned (only for SELECT).

15.4. Updating Database Indexes

Fragmented indexes can cause low performance because additional input and output operations are required to locate data. Depending on the degree of fragmentation, reorganizing or rebuilding the index should be considered. Index reorganization should be applied for a lower degree of fragmentation. The process physically reorganizes nodes of the index. Index rebuild should be applied for a high degree of fragmentation. The process drops the existing index and recreates it.

If you wish to reorganize an index, you can use the following command:

- For SQL Server:
  alter index *index_name* **on** *table_name* **reorganize;**
- For Oracle Database:
  alter index *index_name* **coalesce nologging;**

If you wish to rebuild an index, you can use the following command:

- For SQL Server:
  alter index *index_name* **on** *table_name* **rebuild;**
- For Oracle Database:
  alter index *index_name* **rebuild online nologging;**`

Besides indexes, statistics is another structure that needs to be up to date for optimal performance. The query optimizer uses statistics to prepare a cost-effective execution plan. Statistics helps the query optimizer to estimate number of rows. Outdated statistics may mislead the query optimizer to choose costly operations.

If you wish to update statistics, you can use the following command:

- For SQL Server:
  exec sp_updatestats;
- For Oracle Database:
  exec DBMS_STATS.GATHER_SCHEMA_STATS(null);

For the complete syntax of all commands, see the database server documentation.

A sample quasi SQL script for rebuilding indexes and updating statistics for History DB is located in the DirX Audit installation folder:

*install_path***\conf\defaults\sql\common\adm\rebuild_history_indexes.txt**

16. Dashboard Data in DirX Audit Manager Classic

The Dashboard view is available in DirX Audit Manager Classic and contains components that present charts based on data cubes stored in DirX Audit Database. For more information, see the chapter "Using the Dashboard View" in the *DirX Audit Manager Classic Guide*. The data cubes are built of fact tables containing facts and referencing dimensions. This chapter describes:

- Facts
- Dimensions
- Fact tables
- Component files containing preconfigured dashboard components

16.1. Facts

Facts are always represented as a column of number value in the DirX Audit Database. The following facts are available:

Facts on Audit Events:

- **Total** (FCT_TOTAL) – total number of related audit events.
- **Succeeded** (FCT_SUCCEEDED) – total number of related succeeded audit events. A succeeded audit event has Identification – Outcome field equal to 0.
- **Failed** (FCT_FAILED) – total number of related failed audit events. A failed audit event has Identification – Outcome field not equal to 0.
- **Failed – relative** (FCT_FAILED_RELATIVE) – total relative number of related failed audit events. A failed audit event has Identification – Outcome field not equal to 0.
- **Approved** (FCT_APPROVED) – total number of related approval audit events where a participant approved.
- **Rejected** (FCT_REJECTED) – total number of related rejected audit events where a participant rejected.
- **Created** (FCT_CREATED) – total number of related audit events where an attribute was created.
- **Deleted** (FCT_DELETED) – total number of related audit events where an attribute was deleted.

Facts on History Entries:

- **Total** (FCT_HST_TOTAL) – total number of related history entries.
- **Certified** (FCT_HST_CERTIFIED) – total number of certified entries.
- **Uncertified** (FCT_HST_UNCERTIFIED) – total number of uncertified entries.
- **Without role** (FCT_HST_WOUT_ROLE) – total number of users without a role.

- **Without permission** (FCT_HST_WOUT_PERMISSION) – total number of users without a permission.
- **Without group** (FCT_HST_WOUT_GROUP) – total number of users without a group.
- **Without privilege** (FCT_HST_WOUT_PRIVILEGE) – total number of users without a privilege.
- **Duration** (FCT_HST_DURATION_MI) – total operation duration in minutes.

16.2. Dimensions

Dimensions can be represented either as a column of character string value or as a reference to a table representing an enumerated type in the DirX Audit Database. There are following dimensions available for audit events, history entries and risk data:

Date and time (DIM_DATETIME) – date corresponding to the audit event's Identification – When field.

Date and time (DIM_HST_DATETIME) – date corresponding to a history entry's validity.

Date and time (DIM_HST_GEN_DATETIME) – manually created date using migration scripts. Values are generated as a view of 10,000 days from the current day into the past.

Month (DIM_HST_MONTH) – only for fact tables on history entries. Contains the same content as DIM_HST_DATETIME but is filled only for the last day in a month or for the current day if the month is not yet finished.

Month (DIM_HST_GEN_MONTH) – derived from DIM_HST_GEN_DATETIME only for the last day in a month or for the current day, if the month is not yet finished.

Operation (DIM_OPERATION) – corresponds to the audit event's Identification – Operation field.

Application (DIM_APPLICATION) – digested value representing application, also called target system or connected system in DirX Identity.

Where From – Application (DIM_WHEREFROM_APP) – corresponds to the audit event's Where From – Application field and denotes the application (for example, WebCenter or Manager) with which the action was performed.

Where From – Address (DIM_WHEREFROM_ADDRESS) – corresponds to the audit event's Where From – Address field.

Who – Organizational unit (DIM_WHO_OU) – digested value representing organizational unit of the audit event's initiator (Who).

Who – Organization (DIM_WHO_O) – digested value representing organization of the audit event's initiator (Who).

Who – Country (DIM_WHO_C) – digested value representing country of the audit event's initiator (Who).

Who – Name (DIM_WHO_NAME) – digested value representing name of the audit event's initiator (Who).

What – Organizational unit (DIM_WHAT_OU) – digested value representing organizational unit of the audit event's object (What).

What – Type (DIM_WHAT_TYPE) – corresponds to the audit event's What – Type field denoting the type of the changed object (What). Note that the object can also be an assignment (for example, User-to-Role).

Assignment mode (DIM_ASSIGNMENT_MODE) – digested value representing the assignment mode.

Workflow (DIM_WORKFLOW) – digested value representing the name of an approval workflow related to the audit event.

Activity (DIM_ACTIVITY) – digested value representing the name of an approval activity related to the audit event.

Detail type (DIM_DETAIL_TYPE) – corresponds to the audit event's What – Detail field denoting the detail of the changed object (What).

Password self-serviced or assisted (DIM_PWD_SELF_ASSISTED) – digested value indicating whether the password change was self-serviced or assisted by the helpdesk.

Authentication type (DIM_AUTHENTICATION_TYPE) – corresponds to the Identification – Type field of authentication audit events.

Authentication method (DIM_AUTHN_METHOD) – digested value representing the authentication method.

Authentication method type (DIM_AUTHN_METHOD_TYPE) – digested value representing the authentication method type.

Identification – Source (DIM_SOURCE) – corresponds to the audit event's Identification – Source field.

Identification – **Resource** (DIM_RESOURCE) – corresponds to the resource name of authorization requests.

Identification – Category (DIM_CATEGORY) – corresponds to the Identification – Category field.

DirX Access audit event code (DIM_DXA_CODE) – digested value representing DirX Access audit event's code.

Policy (DIM_POLICY) – digested value representing policy name.

History entry type (DIM_HST_ENTRY_TYPES) – corresponds to the history entry's entry type.

State (DIM_HST_DXRSTATE) – corresponds to the state of the history entry.

Certification campaign type (DIM_HST_DXRTYPE) – corresponds to the history entry's attribute-based type.

Revoke privilege type (DIM_HST_DXRREVOKEPRIVTYPE) – certification campaign settings for the privilege revocation.

User certification campaign result (DIM_HST_CERRESULT) – user certification campaigns results.

Certification campaign lifecycle state (DIM_HST_CAM_LIFECYCLE_STATE) – all, both user and privilege, certification campaigns states.

Orphaned account (DIM_HST_DXTORPHANED) – an indicator of orphaned history accounts.

Target system (DIM_HST_TS) – target systems of history entries.

Target system link (DIM_HST_TSLINK) – corresponds to the application – also called target system or connected system in DirX Identity – of the history entry.

Workflow status (DIM_HST_RESULT) – history workflow result.

Workflow escalation level (DIM_HST_ESC_LEVEL) – history workflow escalation level indicator.

Certification result (DIM_HST_PARTLY_REJECTED) – certification's assignments result.

Organizational Unit (DIM_HST_OU) – corresponds to the organizational unit of the history entry.

Organization (DIM_HST_O) – corresponds to the organization of the history entry.

Location (DIM_HST_L) – corresponds to the location of the history entry.

Risk level (DIM_HST_DXRRSKLEVEL) – corresponds to the risk level of the history entry.

Application (DIM_HST_APPLICATION) – application of the history entry.

Day (DIM_DATE_DAY) – virtual dimension value that is derived from DIM_DATETIME and representing a day.

Month (DIM_DATE_MONTH) – virtual dimension value that is derived from DIM_DATETIME and representing a month.

Year (DIM_DATE_YEAR) – virtual dimension value that is derived from DIM_DATETIME and representing a year.

User organizational unit (DIM_HST_USR_OU) – organizational unit of user in approval workflow entry.

Subject organizational unit (DIM_HST_SUB_OU) – organizational unit of subject in approval workflow entry.

Path (DIM_HST_PATH) – approval workflow entry path.

16.3. Fact Tables

A fact table consists of one or more facts and one or more dimensions. Typically for audit events, there are three facts, for example, total, succeeded and failed or total, accepted and rejected, and several dimensions representing time and other data properties. The fact tables contain aggregated data on audit events or history entries.

16.3.1. Fact Tables on Audit Events

Accounts (FCT_ACCOUNTS) – aggregated data on audit events related to operations over accounts.

Account to Group Memberships (FCT_MEMBERSHIPS) – aggregated data on audit events related to operations over account to group memberships.

Password Changes (FCT_PWD_CHANGES) – aggregated data on audit events related to password changes.

Provisioning Failures (FCT_PROV_FAILURES) – aggregated data on audit events related to failed provisioning operations.

Users (FCT_USERS) – aggregated data on audit events related to operations over users.

User to Privilege Assignments (FCT_USRPRIV_ASSIGNMENTS) – aggregated data on audit events related to user-privilege assignments.

Approvals of Assignments (FCT_APPROV_ASSIGNMENTS) – aggregated data on audit events related to approvals of assignments.

Approvals of Objects (FCT_APPROV_OBJECTS) – aggregated data on audit events related to approvals of operations over objects (entries).

Approvers (FCT_APPROVERS) – aggregated data on audit events related to assignments approvers.

Authentications (FCT_AUTHENTICATIONS) – aggregated data on authentication audit events.

Authorizations (FCT_AUTHORIZATIONS) – aggregated data on authorization audit events originated in DirX Access.

SoD Violations (FCT_SOD_VIOLATIONS) – aggregated data on segregation of duties audit events originated in DirX Identity.

Events (FCT_EVENTS) – aggregated data on audit events of any type.

User changes (FCT_USR_CHANGES) – aggregated data on audit events related to user changes.

16.3.2. Fact Tables on History Entries

History entries (FCT_HST_ENTRIES) – aggregated data on history entries of any type.

SoD violations (FCT_HST_SOD_VIOLATIONS) – aggregated data on violations of SoD policies.

Users (FCT_HST_USERS) – aggregated data on user history entries.

Accounts (FCT_HST_ACCOUNTS) – aggregated data on account history entries.

Approval workflows (FCT_HST_APPROVALS) – aggregated data on approval workflows.

Certification workflows (FCT_HST_CERTIFICATIONS) – aggregated data on certification workflows.

Certification campaigns (FCT_HST_CERTCAMPAIGNS) – aggregated data on certification campaigns.

Recent user certification campaigns (FCT_HST_USR_CERTIFICATIONS_12) – aggregated data on user certifications in the last 12 months.

Recent risk user certification campaigns (FCT_HST_RSK_USR_CERTIFICATIONS_03) – aggregated data on risk user certifications in the last 3 months.

Recent assignment certification campaigns (FCT_HST_ASS_CERTIFICATIONS_12) – aggregated data on assignment certifications in the last 12 months.

Imported group memberships (FCT_HST_IMPORTED_MEMBERSHIPS) – aggregated data on imported group memberships.

16.4. Component Files

DirX Audit is delivered with a set of preconfigured Dashboard components for audit events, history entries and risk data. You can use them directly or as templates, for example in a case when you need to modify a target / connected system name in the dimension filter.

The set of predefined dashboard components for audit events contains a long list of components for DirX Identity and DirX Access audit events. You can also find a generic dashboard component aggregating audit events by the audit source.

The set of predefined dashboard components for history entries contains components for all major DirX Identity entry types including one component that can show aggregated data for all existing entry types in a stacked bar.

The set of predefined dashboard components for risk data contains components for overall user risk.

The dashboard component file name matches the following convention:

{**evn**|**hst**|**rsk**}__*source__facttable__fact1[_fact2]__dimension1[_dimension2][__filterdimension_filtervalue]*.**xml**

where

evn – indicates that audit events are used as the component's data source.

hst – indicates that history entries are used as the component's data source.

rsk – indicates that user risk levels are used as the component's data source.

source – identifies the audited system; dxa – DirX Access, dxi – DirX Identity, any – any data source.

facttable – specifies the fact table's name.

fact – specifies the fact's name.

dimension – specifies the dimension's name.

filterdimension – specifies the name of the dimension to be used for filtering, or more precisely, slicing.

filtervalue – specifies the value to be used for filtering, or more precisely, slicing.

17. Configuring Logging

Sometimes it's necessary to get detailed information about the running application or get additional data for problem analysis. This chapter describes how logging can be configured to get this information.

IMPORTANT: If you enable **DEBUG** level on the top packages a lot of debug information will be produced and such detailed logging can slow down the application significantly. The **TRACE** log level produces even more information. Use this configuration only when needed and always try to set more specific packages (to be logged). Do not use detailed debug logging in production environment for a long time and dispose of produced log information appropriately (log data might contain sensitive information).

17.1. Logging in DirX Audit Manager Classic

The DirX Audit Manager Classic application uses the Log4j 2 logging library for Java.
The configuration file location is:

install_path/**web/audit-manager-classic.war/WEB-INF/classes/log4j2.properties**

This file is in the Java properties format (*key*=*value*).

Log4J has three main components: loggers, appenders and layouts.

To generate **DEBUG** messages into the default log file (*tomcat_install_path*/**logs/dirxaudit-manager.log**) for the DirX Audit Manager Classic application, uncomment the following loggers:

```
logger.solutions-dirx-audit.name = solutions.dirx.audit
logger.solutions-dirx-audit.level = DEBUG
```

If you need more fine-grained logging, you can adjust the packages in the loggers (to a more specific package) or change the log level (for example, to **TRACE**). The set of possible levels contains: **ALL**, **TRACE**, **DEBUG**, **INFO**, **WARN**, **ERROR**, **FATAL** and **OFF**.

Changes after saving the configuration file are applied automatically after a defined time period expires. This time period is defined by the configuration key and value:

```
monitorInterval = 30
```

The value is defined in seconds and is set to 30 seconds as the default.

For more information on Log4j 2, visit http://logging.apache.org/log4j/2.x/.

17.2. Logging in DirX Audit Manager (REST)

The DirX Audit Manager application uses the Ngx-logger logging Angular library.
The configuration file location is:

install_path/web/audit-manager-*tenantID*/plugins/dirx-dxt-app-manager/assets/config/app-config.json

This file is in the JSON format. You can define log level by attribute **common.logLevel**. The level is an integer value that defines the log level for the application. The log level is used to filter the messages that are logged by the application. The lower the number, the more detailed the logging will be. The higher the number, the less detailed the logging will be. The set of possible levels contains: 0 – **ALL**, 1 – **TRACE**, 2 – **DEBUG**, 3 – **INFO**, 4 – **WARN**, 5 – **ERROR**, 6 – **FATAL** and 7 – **OFF**. The default value is 4 – **WARN**. There is an example of the configuration file with the **DEBUG** log level:

```
{
  …
  "common": {
    "logLevel": 2
  }
  …
}
```

The application is logging in the browser console, so you can see the log messages in the browser developer tools console tab.

17.3. Logging in DirX Audit Server

DirX Audit Server uses Log4j 2 (running in the Spring Boot container) for logging.
The configuration file location is:

install_path/server_container/tenants/*tenantID*/conf/log4j2.properties

This file contains configuration in common Log4j 2 properties format.

To generate **DEBUG** messages for the DirX Audit Server application into the default log file, uncomment or set (create/update) the following loggers:

```
logger.solutions-dirx-audit.name = solutions.dirx.audit
logger.solutions-dirx-audit.level = DEBUG
```

If you need more fine-grained logging, you can adjust the packages in the loggers (to a more specific package) or change the log level (for example, to **TRACE**).

DirX Audit Server stores logging messages in the default files:

- *install_path*/**server_container/tenants/***tenantID*/**logs/dirxaudit-server.log**
- *install_path*/**server_container/tenants/***tenantID*/**logs/dirxaudit-server-errors.log**. This file only stores error log messages and thus serves as a backup of error messages, which can be lost in the default file as the result of logging rollover.

Changes after saving the configuration file are applied automatically after a time period defined by the configuration key and value expires. For example:

```
monitorInterval = 10
```

The value is defined in seconds and is set to **10** seconds as the default.

17.4. Logging in DirX Audit Message Broker

DirX Audit Message Broker uses Log4j 2 (running in Apache ActiveMQ container) for logging.
The configuration file location is:

install_path/**message_broker/conf/log4j2.properties**

This file contains configuration in common Log4j 2 properties format.

If you need more fine-grained logging saved into the default log file (*install_path*/**message_broker/data/activemq.log**), you can adjust the log level (for example, to **TRACE**).

Changes after saving the configuration file are not applied automatically: you must restart the DirX Audit Message Broker service to apply the changes.

17.5. Logging in Tools

DirX Audit command-line tools use Java util logging API (integrated in JVM) for logging.

To enable debug logging in command-line tools (located in *install_path*/**tools/**) add a parameter **-debug** to the command.

Examples:

```
install_path/tools/db_fact_population/bin/dxtPopulateFacts -debug
```

```
install_path/tools/db_maintenance/bin/dxtdbtool export -dstdir outdir -debug
```

See the chapter "Using the DirX Audit Tools" in the *DirX Audit Command Line Interface Guide* for detailed tool usage.

The generated log file (the name contains the selected action and timestamp) is located in *install_path*/**tools/***tool_name*/**log/**.

If you need more detailed or more fine-grained logging, you can directly modify the logging configuration file *install_path***/tools/***tool_name***/app/logging.properties**.

To generate logging messages for the specific DirX Audit tool, uncomment or set the respective loggers with the required log level in this file.

This file is a simplified configuration file (in Java properties format), for details see the Java util logging documentation.

18. Managing Cryptographic Material

The DirX Audit installation requires certificates and keys for securing both incoming and outgoing connections. This chapter provides a brief guide to managing the cryptographic material.

18.1. How DirX Audit uses Cryptography

When a party establishes a secure connection, it must be able to build a certificate chain from the known trust anchors and from the certificate(s) presented by the other party. The resulting certificate chain must be valid and trustworthy in order to prove the authenticity of the other party. If the party lacks some certificate to achieve this goal, it can't prove the authenticity of the other party and should abort the connection.

Providing an appropriate set of trusted certificates that act as the trust anchors is therefore paramount for making connections secure. Note that secure connections are assumed by default and the configuration procedure requires following cryptographic material:

- The server key and certificate for the DirX Audit Message Broker and DirX Audit Message Broker admin console, so that the DirX Audit Message Broker can authenticate itself to the JMS clients (for example, the collectors and the DirX Audit plugins) and administrators can use secure connection to the admin console.
- Trusted certificates for the DirX Audit Manager, DirX Audit Manager Classic, DirX Audit Server REST services and jobs when using LDAP or OIDC authentication over a secure channel. These certificates must form a trusted certificate chain to authenticate the configured LDAP Server or OIDC provider.
- Trusted certificates for the DirX Audit Manager Classic and jobs when using the authorization with DirX Access over a secure channel. These certificates must form a trusted certificate chain to authenticate the configured DirX Access Server.
- Trusted certificates for the LDAP and JMS Server Collectors and history synchronization jobs. These certificates must form a trusted certificate chain to authenticate the configured collectors and jobs.

If a service offers a non-secure endpoint, a client of the service can use it. However, secure channels must be preferred when personally identifiable information (PII) is transferred: clients should be configured to connect to secure endpoints and services should offer only secure endpoints to avoid accidental leaks of sensitive information from using a non-secure channel.

18.2. About Java Keystores

There are several formats for representing cryptographic material. DirX Audit uses Java KeyStore (JKS) and requires all the cryptographic material to be supplied in this form.

The JKS format allows storing multiple entries of various types in a single file and therefore can serve as a universal general-purpose cryptographic material container. Entries are referred via **aliases**. A private entry (an entry that contains a private key) can be protected with a password. The entire file can employ password protection as well.

As noted above, a keystore file can contain both certificates, which are public keys with additional information that verifies their origin and conditions for use, and private keys. A keystore file that only contains certificates is often called a truststore to distinguish it from keystores dedicated to hold sensitive data. A truststore is usually used as the source of trust anchors; as such, the file provides no sensitive information (all of the certificates are public), but its content determines which endpoints are trusted and so it must be protected from unauthorized modifications. Keystores that contain private keys are sensitive data in all cases and thus must be kept private.

In DirX Audit, we use the term **keystore** for a keystore file that contains private (server) keys and required certificates – this file is used by servers like DirX Audit Message Broker and the term **truststore** for a keystore file that contains public certificates and the required trust chain, up to the certificate authority (CA) – this file is used by clients like collectors and authenticator and authorization policy enforcement points (PEPs).

18.3. Preparing Cryptographic Material

There are many suitable tools for creating Java keystores. The Java installation itself contains a tool called **keytool** which is sufficient to complete the most common tasks, although this command line tool does not offer the same ease of use as other third-party tools, some of which provide a graphical user interface.

Feel free to use your favorite tool for creating the JKS file for the DirX Audit configuration. This guide shows just a few examples of using **keytool**. See the **keytool** help and documentation for all the details, including outstanding examples of typical tasks (https://docs.oracle.com/en/java/javase/11/tools/keytool.html).

To set up the test environment, you will need to create your own CA and testing certificates and then use the created CA to sign these certificates. The generated certificates will be added to a keystore that will be used by the Configuration Wizard when a keystore is needed. The complete trust certification chain (CA certificate and signed generated certificates) will then be added to a truststore that will be used by the Configuration Wizard when a truststore is needed.

To set up your production environment, you will typically get the CA certificate and generated certificates. You will then only need to add them into the keystores and trustores if they are not already in the required (JKS) format.

The next sections describe these preparation tasks.

18.3.1. Creating a Certificate Authority

In order for a certificate to be trusted, the certificate must be signed by a well-known and accepted CA. However, involving a well-known CA for a testing environment might not be acceptable for many reasons; for example, the costs of the certificates. The solution is to make and use your own CA for use within the testing environment.

The following command creates a keystore file **ca.jks** that contains the entry with the alias **ca**. The entry contains a new RSA private key and a self-issued certificate with the DN set to **CN=CA, O=Demo** and with the validity period of **3,650** days (approximately 10 years). Passwords are entered on prompt and the tool's defaults apply for other settings.

```
keytool -genkeypair -keystore ca.jks -alias ca -keyalg RSA
   -dname CN=CA,O=Demo -ext bc:c -validity 3650
```

We need to export the CA public certificate in the PEM format for later use. The following commands perform the actions required to have a CA certificate in a file:

```
keytool -keystore ca.jks -alias ca -exportcert -rfc > ca.pem
```

18.3.2. Creating and Signing the Server Key, Certificate and Keystore

The following command creates a keystore file **server.jks** that contains the server key. The server identification must be properly set in the certificate and typically matches the hostname of the machine hosting the desired service, for instance:

```
keytool -genkeypair -keystore server.jks -alias server -keyalg RSA
   -dname CN=server.demo.org,O=Demo
```

The server certificate then needs signing by a CA in order to be trusted when the CA is trusted. First, we need to create the certificate request and then to sign this certificate with our CA. The certificate is then exported into PEM format. For the request, we must set the correct hostname of the machine hosting the desired service. In this case, it's **server.demo.org**. The validity is set to **1,825 days** (approximately 5 years). We will use the CA created in the previous step in this example. The following commands perform the actions required to have a CA-signed server certificate in a file:

```
keytool -keystore server.jks -certreq -alias server > server.csr

keytool -keystore ca.jks -gencert -alias ca
   -ext ku:c=dig,keyEncipherment,keyAgreement
   -ext san=dns:server.demo.org
   -validity 1825
   -rfc -infile server.csr > server.pem
```

The **server.pem** file now contains the CA-signed server certificate, which must be imported into the server keystore. To achieve this result, a chain must be composed by chaining the **ca.pem** and **server.pem** files. Use this command on Windows:

```
copy ca.pem+server.pem chain.pem
```

Or this command on a UNIX system:

```
cat ca.pem server.pem > chain.pem
```

The chain for importing is now ready in chain.pem and can be imported with a single command:

```
keytool -keystore server.jks -importcert -alias server
   -file chain.pem
```

The tool presents the CA certificate, if not imported into the keystore, and asks whether it is trusted; if agreed, the input is imported. Finally, **server.jks** contains the server key and certificate signed with the testing CA.

Of course, this scenario can be modified. For instance, if there is an existing CA that can handle the CSR, it is enough to send the CSR file and then import the reply, leaving out some of the steps. It might be necessary to include additional extensions in some of the steps in order to achieve the desired result, especially when dealing with an existing CA that may have specific requirements.

Beware of a minor bug that is currently in **keytool**: error messages are displayed on the standard output instead of the error output, so if a command that redirects the standard output to a file (like the examples just given that store a PEM file), the error message can end up appearing in the file instead of the display. In this case, the user can see no sign of a problem, although the output is invalid. When redirecting the output of **keytool**, check the output. If it contains an error message, try to correct the problem first to get the correct output.

18.3.3. Creating the Server-Client Truststore

While the keystore created in the previous step can be used by the server itself, we need to create a truststore for the clients that will connect to this server that contains all of the public certificates of the server and its trust chain. In our example, it's only the server certificate and the CA certificate as this is the final root one.

To import the CA certificate (stored in the file **ca.pem**):

```
keytool -keystore client.jks -importcert -alias ca -file ca.pem
```

To import the server certificate (stored in the file **server.pem**):

```
keytool -keystore client.jks -importcert -alias server
   -file server.pem
```

18.3.4. Importing Certificates in Different Formats

A more typical scenario assumes that the input for a keystore already exists, but in a different format (for example, PKCS #12 or CER) and must be imported to a Java keystore. The following command examples show how to use **keytool** for these import tasks.

A whole PCKS #12 keystore **my-store.p12** can be imported into a Java keystore **client.jks** using this command:

```
keytool -importkeystore
   -srckeystore my-store.p12 -srcstoretype PKCS12
   -destkeystore client.jks
```

A certificate from file **cert-file.cer** can be imported into a Java keystore **client.jks** under the alias **my-cert** using this command:

```
keytool -importcert
   -alias my-cert -file cert-file.cer
   -keystore client.jks
```

You can use tools like **openssl** to convert other formats.

18.3.5. Importing Certificates into the Java CA Store

It may also be necessary in some cases to import a certificate into the general Java CA store instead of into a custom truststore; for example, when setting up the secure connection to DirX Access Server configured in the DirX Audit authorization settings. You need to import it into the Java truststore **cacerts** file located in *jre_install_path***/lib/security/**.

You can use the following **keytool** command to import the CA certificate **dxaca.pem** into the target JRE truststore **cacerts**:

```
keytool -keystore full_file_path/cacerts -importcert -alias cert.dirxaccess
-file full_file_path/dxaca.pem
```

18.4. Updating Cryptographic Material

You may need to update the cryptographic material, for example to replace expired keys.

If the standard Java KeyStore (JKS) is used, update its content only. No **Configuration Wizard** action is required.

If the original keystore and truststores locations (file names and paths) and passwords are preserved, follow these steps (no **Configuration Wizard** action is required):

- Prepare the new keystore and truststores. Keep the original passwords. See the section "Preparing Truststores and Keystores for SSL Configuration" in the *DirX Audit Installation Guide* for more information.
- Stop all **DirX Audit Server**, **DirX Audit Message Broker** and **Apache Tomcat**, hosting DirX Audit Manager, system services.
- Replace the stores (JKS files). Keep the original locations and passwords.
- Start all aforementioned system services.

If the original keystore and truststores locations or passwords are modified, follow the steps below. Some of them require **Configuration Wizard** to perform core- and tenant-specific actions.

- Prepare the new keystore and truststores. Keep the original passwords. See the section "Preparing Truststores and Keystores for SSL Configuration" in the *DirX Audit Installation Guide* for more information.
- Stop all **DirX Audit Server**, **DirX Audit Message Broker** and **Apache Tomcat**, hosting DirX Audit Manager, system services.
- Replace the stores (JKS files). Keep the original locations and passwords.
- Perform the following cryptographic material related step with **Configuration Wizard** for core configuration, if relevant:
  - Set **Keystore location** and **Keystore password** and **Truststore location** and **Truststore password** in **Message Broker Connectivity**.

- Perform all relevant cryptographic material related steps with **Configuration Wizard** for tenant configuration for each affected tenant.
  - Set **Truststore location** and **Truststore password** in Authentication Configuration.
  - Set **Truststore location** and **Truststore password** in Authorization Configuration when **DirX Access PEP** is selected.
  - Set **Truststore location** and **Truststore password** in Server LDAP Collector for DirX Identity Format.
  - Set **Keystore location** and **Keystore password** and **Truststore location** and **Truststore password** in REST Service Configuration.
  - Set **Truststore location** and **Truststore password** in REST Service Authentication Configuration.
  - Set **Truststore location** and **Truststore password** in Server JMS Collector for DirX Identity Format when **Custom Message Broker** is selected.
  - Set **Truststore location** and **Truststore password** in Server JMS Collector for DirX Access Format when **Custom Message Broker** is selected.
  - Set **Truststore location** and **Truststore password** in Server JMS Collector for DirX Audit Format when **Custom Message Broker** is selected.
  - Set **Truststore location** and **Truststore password** in History Synchronization LDAP Configuration.
- Start all aforementioned system services, if not started automatically by **Configuration Wizard**.

19. Monitoring DirX Audit with JMX

This chapter describes how to monitor DirX Audit with JMX, including how to monitor:

- DirX Audit Server components
- DirX Audit Message Broker

19.1. Monitoring DirX Audit Server Components

DirX Audit Server components contain three collector types and support units for the message processing in collectors. DirX Audit Server can monitor the collectors and selected support units (error storage, persistence unit and splitter) using Java Management Extensions (JMX), a set of specifications used for network and application management. For more information on collectors, see the chapters "Managing DirX Audit Server" and "Configuring DirX Audit Collectors" in this guide. For more information on error handling, see the chapter "Managing DirX Audit Server Error Handling" in this guide.

19.1.1. Managed Beans for DirX Audit Server Monitoring

Managed beans are used to provide access to DirX Audit Server component monitoring. Each tenant application has managed beans for monitoring of the error handling component, the persistence component, and for monitoring of collectors. Each collector is configured for the chosen DirX product.

The monitoring beans are:

- **DxtSrvErrorStorageMBean** – the error storage bean that holds statistics about different types of errors occurring in the processing of records from File, JMS and LDAP collectors.
- **DxtSrvPersistenceMBean** – the monitoring bean for the Persistence unit that holds the number of audit messages persisted in the DirX Audit Database and statistics about the splitter component that splits entries into separate audit messages. It also provides information about the state of the connection to the database.
- **DxtSrvFileCollectorMBean** – the monitoring bean from the File collector that gives access to route statistics. The statistics concentrate on whole exchanges between the File collector and the Persistence unit. An exchange can contain one or more audit messages.
- **DxtSrvJmsCollectorMBean** – the monitoring bean for the JMS collector that gives access to route statistics. The statistics concentrate on whole exchanges between the JMS collector and the Persistence unit. An exchange can contain one or more audit messages.
- **DxtSrvLdapCollectorMBean** – the monitoring bean for the LDAP collector that gives access to route statistics. The statistics concentrate on whole exchanges between the collector and the Persistence unit and on the number of records contained in these exchanges. An exchange can contain one or more audit messages.

19.1.2. DirX Audit Server Component Attributes to Monitor

This section describes some important attributes of DirX Audit Server components that can be monitored.

19.1.2.1. Error Handling Attributes

Error handling attributes that can be monitored include:

- **NumberOfProcessedErrors** – the total number of errors processed by the error storage unit, including the following ones
- **NumberOfProcessedErrorsDuplicates** – the number of duplicates processed by the error storage unit
- **NumberOfProcessedErrorsNonRecoverable** – the number of non-recoverable errors processed by the error storage unit
- **NumberOfProcessedErrorsRecoverable** – the number of recoverable errors processed by the error storage unit

19.1.2.2. Common Statistics for Collectors and the Persistence Unit

Common statistics attributes for File, JMS and LDAP collectors and the Persistence unit that can be monitored include:

- **ExchangesCompleted** – the number of completed exchanges in the collector. In this context, the exchange represents the whole route from the collector to the Persistence unit.
- **ExchangesFailed** – the number of failed exchanges
- **ExchangesInflight** – the number of exchanges currently being processed
- **FailuresHandled** – the number of handled failures; for example, the handling of duplicate messages
- **LastProcessingTime** – the time in milliseconds needed for processing the last exchange
- **MaxProcessingTime** – the maximum processing time from all processed exchanges
- **MeanProcessingTime** – the mean processing time of all processed exchanges
- **MinProcessingTime** – the minimum processing time from all processed exchanges
- **TotalProcessingTime** – the total processing time of all processed exchanges

19.1.2.3. LDAP Collector Attributes

LDAP collector attributes to monitor include:

- **ProcessedEntries** – the number of all processed original audit messages from all processed exchanges
- **RecordsSent** – the number of audit messages sent to the database
- **Connected** – the connection status to the LDAP server
- **LastConnectedTimestamp** – the timestamp when the LDAP collector was last connected to the LDAP server
- **LastConnectionCheckedTimestamp** – the timestamp when the LDAP connection was last checked

19.1.2.4. Persistence Unit and Splitter Attributes

Persistence unit and splitter attributes that can be monitored include:

- **TotalProcessedMsgCount** – the number of messages processed by the Persistence unit that were not processed by the error storage unit
- **Connected** – the connection status to the database
- **LastConnectedTimestamp** – the timestamp when a record was last sent to the database
- **LastConnectionCheckedTimestamp** – the timestamp when a connection status was last checked

19.1.3. Monitoring DirX Audit Server Components with JConsole

Monitoring statistics can be accessed using JConsole, a part of the Java Virtual Machine that provides a graphical user interface for monitoring of components in each DirX Audit Server tenant-specific application. Each tenant has configured separate access to remote monitoring. Connection URL and credentials are needed to access the statistics:

- The server monitoring URL is: **service:jmx:rmi:///jndi/rmi://127.0.0.1:300*xx*/jmxrmi**
- The port, JMX username and password were set with the Tenant Configuration Wizard in the Application Container Configuration step. The port is specific for each tenant. The value is selected by default from the range **30091** – **300*xx***.

On the left side of JConsole for each tenant application is the monitor package name: **com.dirxcloud.audit.management.server**. Each tenant has a group of managed beans corresponding with collectors configured on the tenant and labeled with the tenant identifier in the beginning of each managed bean name. In addition, each tenant has one MXBean that provides access to the individual collector managed beans through attributes and operations.

Here is an example of the name for the DirX Identity File collector MBean in JConsole:

```
1c523d49-bcf9-40b7-973d-93894ab0bb24.DxtSrvLdapCollectorMBean.dxi
```

Each component bean has a group of attributes for monitoring DirX Audit Server components. Some of these attributes are mentioned in the section "DirX Audit Server Component Attributes to Monitor".

19.1.4. Monitoring DirX Audit Server Components with dxt_mgmt_check

The monitoring command-line tool **dxt_mgmt_check** is provided with the DirX Audit installation. It allows administrators to read monitoring managed beans and gives access to statistics for each managed bean. Nagios IT monitoring software can be set up to use this tool to collect statistics.

19.1.4.1. Common Options

Common options include:

-h or **--help** – show this help message and exit.

-p or **--password** *password* – connection credentials – password

-P – connection credentials – password – interactive input

-t or **--tenantid** *tenantid* – tenant ID

-u or **--user** *username* – connection credentials – username

-U or **--url** *url* – connection URL

-v or **--verbose** – be verbose during execution

-V or **--version** – print version information and exit

Options for commands **get_bean_attributes** and **get_value**:

-A or **--attribute** *attribute_name* – attribute name; see the **get_bean_attributes** command for a list of valid attribute names

-B or **--bean** *bean* – bean name, see the **get_beans** command for a list of valid bean names

-T or **--type** *product_type* – DirX product type, see the **get_product_types** command for a list of basic DirX product types

19.1.4.2. Commands

Commands are:

get_bean_attributes – gets all statistics accessible from the bean

get_beans – gets all available beans

get_product_types – gets the DirX product types: **dxi** (DirX Identity), **dxa** (DirX Access), **dxt** (DirX Audit)

get_scopes – gets the available time scopes

get_value – get the value of a specific attribute from a specific bean

Here is an example for getting all available attributes in the File collector MBean for DirX Identity:

```
dxt_mgmt_check
 -p password
 -t 1c523d49-bcf9-40b7-973d-93894ab0bb24
 -u user
 -U service:jmx:rmi:///jndi/rmi://127.0.0.1:30091/jmxrmi
get_bean_attributes
 -B DxtSrvFileCollectorMBean
 -T dxi
```

19.2. Monitoring DirX Audit Message Broker

This section describes how to monitor DirX Audit Message Broker with JConsole and describes some important component attributes to monitor.

19.2.1. Monitoring DirX Audit Message Broker with JConsole

Monitoring statistics for the DirX Audit Message Broker can be accessed using JConsole, the JMX graphical user interface. To access the statistics, you need the connection URL and credentials:

- The broker monitoring URL is: **service:jmx:rmi:///jndi/rmi://127.0.0.1:30699/jmxrmi**
- The JMX username and password were set with the Core Configuration Wizard in the Message Broker System Service step.

On the left side of the JConsole is the monitor package name: **org.apache.activemq**. This package contains attributes common to the whole DirX Audit Message Broker and specific attributes for each queue.

19.2.2. DirX Audit Message Broker Attributes to Monitor

Common attributes for DirX Audit Message Broker:

- **CurrentConnectionsCount**, **TotalConnectionsCount**
- **UptimeMillis**, **MemoryLimit**, **DataDirectory**
- **Queues**, **QueueSubscribers**
- **TotalConsumerCount**, **TotalMessageCount**, **TotalEnqueueCount**, **TotalDequeueCount**
- **MinMessageSize**, **MaxMessageSize**, **AverageMessageSize**

Common attributes for DirX Audit Message Broker health:

- **CurrentStatus**

Attributes for queue:

- **Name**
- QueueSize
- CurrentConnectionsCount
- ProducerCount, ConsumerCount
- EnqueueCount, DequeueCount
- MinEnqueueTime, MaxEnqueueTime, AverageEnqueueTime
- MinMessageSize, MaxMessageSize, AverageMessageSize, StoreMessageSize

For more information on Apache ActiveMQ monitoring with JMX, see its documentation at https://activemq.apache.org/jmx.html.

20. Monitoring DirX Audit Data with Reports

The reports described in this section give an overview of the data in the DirX Audit Data Database (audit messages) and the DirX Audit History database (history entries) and especially on exceptions that deserve administrative attention. Consider running a subset of these reports on a regular basis.

20.1. Reports for Monitoring the DirX Audit Database

The reports described here give an overview of the number of audit events or history entries.

20.1.1. Audit events database status

This report shows charts on the number of audit events produced in the selected time range, grouped by **Source**, **Operation** and **What Type** in total and for a specific year, month and day.

The input parameters are:

- **When** – defines a valid time range for the report. The finish date is the **When** time point (for example, **15/05/2020, 10:00**).
- **Create short report** – generates the report without charts.
- **Total only** – generates the report with only totals by **Source** and the Grand Total. No charts are generated.

20.1.2. History entries database status

This report shows charts on how many history entries have been created in the selected time range, grouped by **Entry type** name in total and for specific year, month and day.

Input parameters are:

- **When** – defines a valid time range for the report. The finish date is the **When** time point (for example, **15/05/2020, 10:00**).
- **Entry types** – generates the report with the specified entry types.
- **Create short report** – generates the report without charts.
- **Total only** – generates the report with the total only.

20.2. Reports for Monitoring the DirX Audit History Database

The reports described here give information on history entries that need special attention.

20.2.1. Missing entry links

This report lists history entries that reference other entries and where at least one referenced entry cannot be determined. Per entry type, it lists the link attribute names and, for two selectable time points, the number of missing links for all link attribute values. The **Difference** column gives the difference between the two time points. A summary line after each entry type shows the relative number trend as a graph and the aggregated numbers for all link attributes.

Input parameters are:

- **When** – generates the report for history entries created within two selectable time points.
- **Value valid after** – generates the report for history entries created after the specified time point to prevent the inclusion of old or irrelevant history entries.
- **Type name** – generates the report for the specified entry type names.
- **Attribute name** – generates the report for the specified link attribute names.
- **Create short report** – generates the report for attributes that have at least one missing entry link.
- **Overview only** – generates the report without a trend chart and detailed information about attributes.
- **Ancient and Fresh** – generates the report with the number of missing entry links created before and after the decisive date in two separate columns.
- **Ancient and Fresh decisive date and when** – specifies the **decisive date** time point and the **When** time point represents the effective date. The column for ancient missing links comprises the period from **Value valid after** to the decisive date, while the column for fresh links comprises the period from the decisive date to the **When** time point.

20.2.2. Missing entry links by target system

This report lists account and group history entries where at least one referenced entry cannot be determined. Per target system, it lists the number of missing account-group membership links out of all links for the selected time range. The **Difference** column gives the difference between the beginning and end of the time range. A summary line after each target system shows the trend as a graph and the aggregated numbers for all link attributes.

Input parameters are:

- **When** – generates the report for history entries created within two selectable time points.
- **Value valid after** – generates the report for history entries created after the specified time point to prevent the inclusion of old or irrelevant history entries.
- **Target system** – generates the report for the specified target systems.
- **Type name** – generates the report for the specified entry type names.
- **Attribute name** – generates the report for the specified link attribute names.
- **Create short report** – generates the report for attributes that have at least one missing entry link.
- **Overview only** – generates the report without a trend chart and detailed information about attributes.
- **Ancient and Fresh** – generates the report with the number of missing entry links created before and after the decisive date in two separate columns.
- **Ancient and Fresh decisive date and when** – specifies the **decisive date** time point and the **When** time point represents the effective date. The column for ancient missing links comprises the period from **Value valid after** to the **decisive date**, while the column for fresh links comprises the period from the **decisive date** to the **When** time point.

20.2.3. Missing entry links with details (CSV format)

This report lists history entries that reference other entries and where at least one referenced entry cannot be determined. It provides details for each entry in CSV format so that it can easily be post-processed by applications like Microsoft Excel. Each line contains the entry type, the entry's **dxrUid**, **dirxEntryUUID** and name, the link attribute name and value and the value's lifetime (VALID_FROM – VALID_TO).

Input parameters are:

- **When** – generates the report for link attribute values created before the specified time point.
- **Value valid after** – generates the report for link attribute values created after the specified time point to prevent the inclusion of old or irrelevant history entries.
- **Type name** – generates the report for the specified entry type names.
- **Attribute name** – generates the report for the specified link attribute names.

20.2.4. Duplicated history entries (CSV format)

This report produces a list of history entries that are considered duplicates in CSV format so that it can easily be post-processed by applications like Microsoft Excel. Each line shows the identifier of the duplicated part (ENTRY, DN, DIRX_ENTRY_UUID), the entry type, then the entry's **dxrUID**, **dirxEntryUUID**, name, DN and lifetime (VALID_FROM – VALID_TO).

Input parameters are:

- **When** – the report contains only those history entries that have been created before this time point.
- **Entry** – the report includes duplicate history entries linked to the same logical entry.
- **DN** – the report includes history entries with multiple records having the identical DN value.
- **DIRX_ENTRY_UUID** – the report includes history entries with duplicate DIRX_ENTRY_UUID values.
- **Entry types** – generates the report for the specified entry types.

DirX Product Suite

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

DirX Identity

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

DirX Directory

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

DirX Access

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

DirX Audit

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

For more information: support.dirx.solutions/about

Legal remarks

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